Design and Development of a Flight Deck Motion Display System

by

Nicholas R. Bourgeois, B.Eng.

Carleton University

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Masters of Applied Science

Ottawa-Carleton Institute for

Mechanical and Aerospace Engineering

Department of

Mechanical and Aerospace Engineering

Carleton University

Ottawa, Ontario

September 30, 2008

The undersigned recommend to

the Faculty of Graduate Studies and Research

acceptance of the thesis

Design and Development of a

Flight Deck Motion Display

submitted by **Nicholas R. Bourgeois, B.Eng.**

in partial fulfillment of the requirements for

the degree of Masters of Applied Science

Dr. R.G. Langlois
Thesis Supervisor

Dr. M.I. Yaras
Chair, Department of
Mechanical and Aerospace Engineering

Carleton University

September 16, 2008

Abstract

Shipboard launch and recovery of maritime helicopters is difficult and dangerous, particularly in elevated sea conditions. Typically, during the aircraft hover and landing phases, a Landing Signals Officer (LSO), among other responsibilities, monitors ship motions from a position close to the flight deck and communicates the state of deck quiescence to the pilot. This thesis presents the design and development of a Flight Deck Motion Display (FDMD) system that has been developed for use by navies to improve the safety and efficiency of helicopter/ship operations. It does this by displaying critical motion parameters using an ergonomically engineered user interface that has been designed to be compatible with data priorities of ship deck and helicopter operators. This user interface features a novel quiescent period indicator, which allows an operator to in one glance, determine the current state of the ship, which motions dominate that state, and combined with the operator's own experience, improve ability to determine whether the flight deck is tending towards or away from quiescence.

The FDMD hardware configuration has been designed to be a four-component system with two sensors and two computers providing operating redundancy and backup. It can also be reconfigured as a two-component system for demonstration and evaluation purposes, or expanded to any number of computers and sensors.

The FDMD software has been meticulously engineered to support multiple hardware and software configurations. Future expansion of the software has been facilitated by its modular design, and stability has been emphasized to ensure reliability at critical times.

A user interface evaluation took place with the cooperation of 12 Wing, Shearwater and the results provided valuable insight into the role the FDMD should play in helicopter operations, as well as the appearance and behaviour of the quiescent period indicator.

Acknowledgements

I would like to thank my supervisor, Prof. Rob. Langlois for his guidance, dedication to the project and his motivational speeches. I would also like to thank Kin Wing Tsui for his support and patience with Virtual Flight Deck Real-Time support and related software contributions.

I'd like to thank General Dynamics Canada and specifically Amanda Simpson for her hard work in project management and promotion of the FDMD within GDC. The joint funding and support from GDC and Ontario Centres of Excellence helped make this research collaboration project a success.

Lastly I'd like to send a big thanks to 12 Wing, Shearwater for helping make the FDMD evaluation a success.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Launch and recovery of shipboard helicopters is a safety critical task which requires an accurate assessment of ship motions. While human detection of orientation is relatively accurate, judgement of vertical acceleration is not. The issue is further complicated due to the fact that flight decks on most non-flight-dedicated military vessels are located at the stern, which causes the flight deck's vertical acceleration to be strongly a function of both the vertical acceleration of the ship's centre of gravity and its pitch angular acceleration. Historically, the approach taken to allow vertical acceleration limits to be expressed as limits on pitch angle has been to impose a very narrow band on acceptable pitch angles. This results in not only a reduced motion envelope for operations, but can be misleading in cases where high flight deck vertical accelerations occur at low pitch angles.

This thesis presents the Flight Deck Motion Display (FDMD) system, which has been developed to increase operational safety and efficiency of helicopter operations in high sea states. This device operates by delivering real-time ship motion information to a flight deck operator along with how those motions compare to predefined limits for specific flight deck operations.

1.1 Background

1.1.1 Helicopter Operations

While the FDMD has been designed to be capable of being used in any flight deck operations, the focused application of the FDMD in this project is in helicopter landings. In Canadian shipboard helicopter landings, standard operational procedures are followed. These involve the use of trafficator lights for visual communication between the deck crew and the pilot, a hauldown system to aid helicopter fine positioning during landing, and securing device to stabilize the helicopter once it is on deck. The primary steps of Canadian landing procedures are described in detail in Shipborne Helicopter Operating Procedures (SHOPS) [1].

Hauldown landings are used during moderate to high sea states and are much more demanding than landings in calm conditions. It is for this type of landing that the FDMD's use is targeted. In general, the following steps are followed:

1. When the helicopter receives clearance it approaches the flight deck and enters a high hover of 15 to 17 feet above the deck.

2. The helicopter positions itself to the side of the Rapid Securing Device (RSD) and lowers a messenger cable. As soon as it is safely possible, flight deck personnel ground the messenger cable and attach a hauldown cable.

3. The hauldown cable retracts to apply a constant tension, but does not prevent the distance between the helicopter and ship deck from changing.

4. The helicopter descends to a lower hover of approximately 5 feet and the pilot notifies the deck crew when ready to land.

5. The Landing Signals Officer (LSO) gives instructions for the helicopter to position

itself over the RSD, and when the deck is steady, orders the helicopter to land.

6. Once the helicopter is on deck the hauldown cable tension is increased and the helicopter is mechanically secured to the ship by the RSD and helicopter securing probe(s). Flight deck personnel may install additional securing restraints to the aircraft.

In the procedure outlined above, the LSO must wait until the deck is steady or 'quiescent' before advising the helicopter to land. If it appears that it will be some time before a quiescent period occurs the helicopter may return to high hover or in some cases be waved off and the landing process repeated. The FDMD is designed to aid the LSO in this decision.

Figure 1.1 shows a photograph from the point of view of an LSO aboard Canada's $Halifax$-class frigates. The console on the left is used to control the RSD and trafficator lights, and the device on the right is a combined roll/pitch indicator.

1.1.2 DRDC Design Specification

The origins of the concept of the FDMD in Canada begin with a design specification written by Colwell in 2004 [2], which he states is the direct consequence of the experience gained by Defence Research and Development Canada (DRDC) in supporting Canadian frigate helicopter/ship flight deck certification trials. This document introduces the basic operation and design of a quiescent period indicator, for quickly identifying whether ship motions are within limits for a particular operation. Some of the justifications for the implementations of a flight deck motion monitoring device can be traced back to previous research performed by Colwell [3] on the correlations between specific ship motions and pilot control inputs as well as successful landing statistics. The conclusions derived from this research are that roll angle amplitude and flight deck vertical acceleration are critical to

Figure 1.1: Photograph of the LSO console aboard Canada's Halifax-class frigates.

successful helicopter operations, that there is little correlation between vertical acceleration and pitch angle amplitude, and that while pitch angle amplitude is not a critical parameter for helicopter landings, pitch angular acceleration is (as it is related to flight deck vertical acceleration).

In [2], Colwell states that the advantages offered by a flight deck monitoring system should reduce the number of helicopter wave offs in operations, and should reduce the number of incidents of high-torque transients, which can be directly related to time-between-failure and required maintenance. Colwell presents the quiescence algorithms and the operation of a two-state and three-state quiescent period indicator. The Flight Deck Motion System (FDMS) consists of a ship motion sensor unit, a data processing computer located in the same compartment as the sensor, and two visual displays for operator use, although there is no discussion on logistically how such a hardware system would operate. Colwell's vision for the graphical implementation of the operations mode of the FDMS consists of two screens shown in Figure 1.2. One screen consists of individual bar graphs for each motion parameter whose height is related to the magnitude of the motion. The second screen consists of a single bar graph, corresponding to the single parameter that is closest to its limits. Colwell defines a quiescent period as a time when all monitored motion parameters are within limits for performing a specific activity. Two rules for changing quiescence status are defined: (1) the state is not quiescent when at least one limit is exceeded, and (2) in order to change state from non-quiescent to quiescent, each motion which has exceeded its limit must experience a subsequent motion peak below its limit.

1.1.3 DND Design Requirements

In 2006 the Canadian Department of National Defence (DND) released a document titled "Flight Deck Motion Display Requirements Specification" [4] which outlines the draft design requirements for a flight deck motion monitoring system. This document has acted

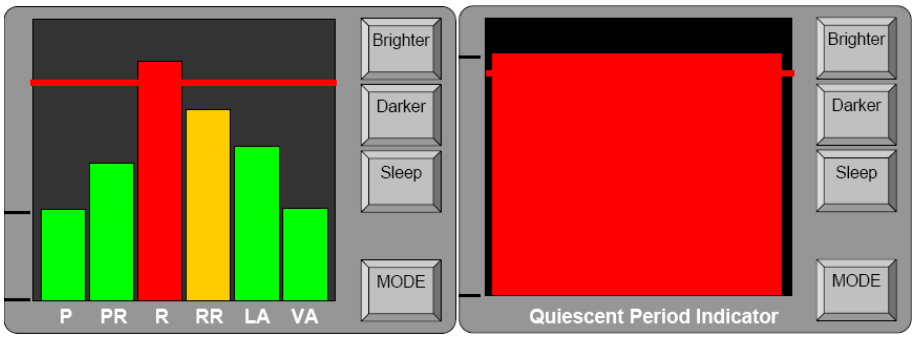

Figure 1.2: Screenshots of FDMS user interface [2].

as a guideline for the design of Carleton's FDMD system. It provides details on the hardware arrangement, hardware ruggedness specifications, software operating modes, and software capabilities for an FDMD system. Some highlights are listed below.

- The system shall be capable of supporting many helicopter-ship operations including: traversing, blade and tail folding/unfolding, start-up, launch, departure, approach, deck landing practice, vertical replenishment, hoisting, in-flight refueling, recovery, shutdown, straightening, and lashing.
- The system envisioned by DND includes a ship motion sensor, a computer sub-system for collecting and processing all data, and a smart display for presenting data in the LSO compartment.
- Any operating system(s) used shall be stable and commercially available, and all custom software shall be written in either C or C++.
- The FDMD system shall consist of five operating modes: startup, deck operations, flight operations, data display, and maintenance and testing.
- The FDMD system response time for real-time data monitoring shall not exceed 250 ms.

- Visual alerts are used to notify the operator of an unusual condition or system failure, and shall include means such as inverse colours, blinking, etc. Auditory alerts are not an effective means of communicating information in the noisy operating environment in which the FDMD will be used.

- All operating modes except for maintenance and testing shall be controlled by software-addressable function keys on the smart display system. Touch screen technology is not suitable for the physical environment in which the smart display subsystem will be used.

Due to the nature of this document being a draft specification and the envisioned goals of the FDMD project beyond the uses DND has described in this document, the design specifications are only loosely based on the contents of it.

1.2 Commercial Ship Monitoring Systems

While there are no known implementations of quiescent period indicator variants available to navies today, there are a number of similar flight deck motion monitoring systems available. The purpose of this review is to assess the capabilities of commercially available systems but not to compare their effectiveness as the technical details of many of them are not publicly available. The following sections will present some of these systems.

1.2.1 Fugro

The Fugro helideck motion monitoring system [5] possesses many of the capabilities that the FDMD will also support. It is designed to present as much information as possible to a flight deck operator in order to judge sea state conditions. A snapshot of its user interface is shown in Figure 1.3.

Figure 1.3: Screenshot of Fugro helideck motion monitoring system [5].

From this screenshot it can be determined that the system has the following capabilities:

- Display of roll/pitch/heave current values and data history;
- Display of maximum previous values of monitored parameters;
- Display of current ship heading, wind direction, and wind speed;
- Periodical updates on meteorological data such as weather, clouds, visibility, air temperature, and air pressure; and
- Capability to be connected to multiple data sources.

Some apparent disadvantages of this system include:

- The screen contains a lot of information, some of which may not be relevant to certain operations.
- In order to include all of this information and have the text remain readable this user interface requires a high display resolution, of at least 1024 x 768 pixels if not more, which increases the size requirement of the display hardware.
- It appears that the only way the software can be interacted with is through the use of a keyboard or mouse.

While vertical velocity and acceleration are not displayed in Figure 1.3, according to the system documentation they are also available. This system is designed to have all of its data sources connected to a central server, which then distributes information across a local area network to individual computers. It is also designed to be connected to a satellite data source for receiving weather data.

1.2.2 Ship Motion Control

Ship Motion Control (SMC) has a product named SMChms, which is defined as "a complete helicopter takeoff and landing system" [6]. Fugro's and SMC's user interfaces are extremely similar and a screenshot of SMChms is displayed in Figure 1.4.

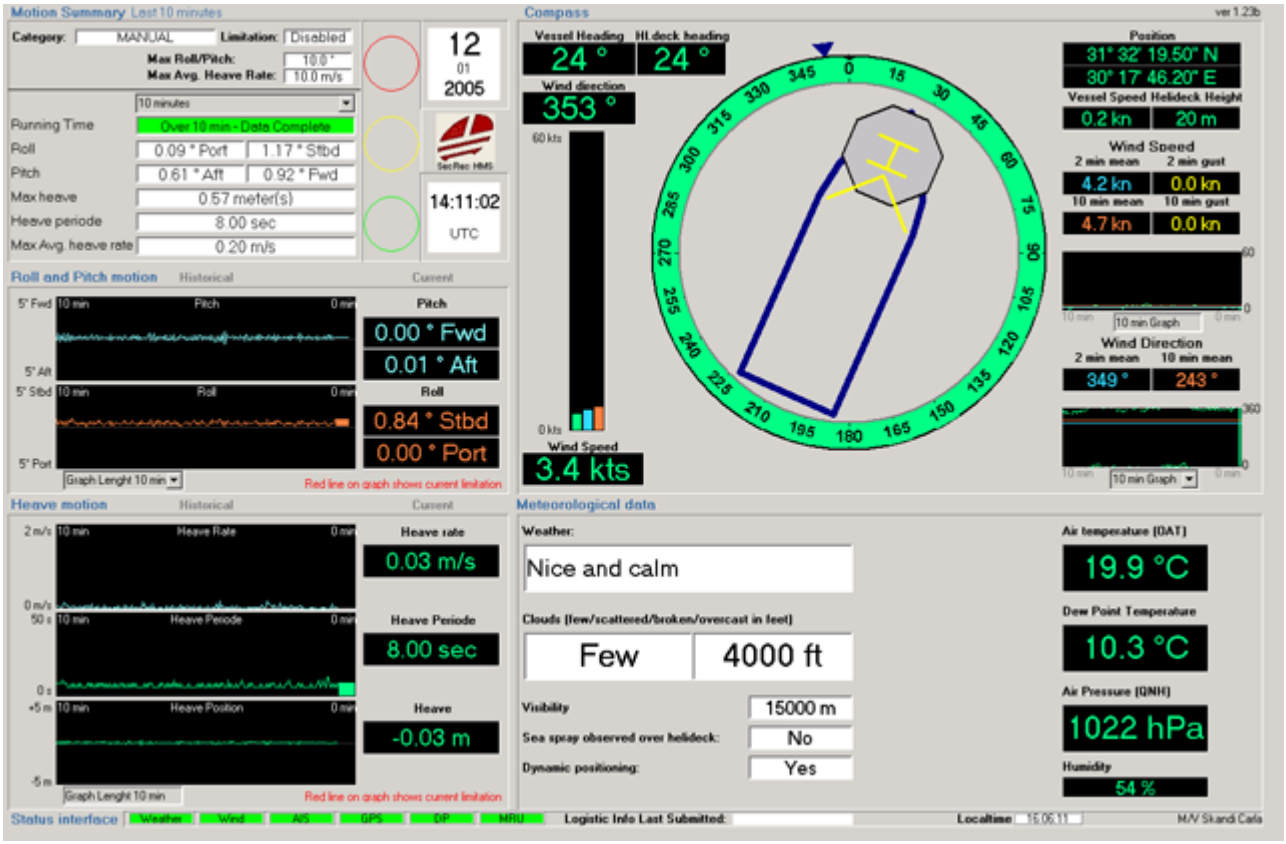

Figure 1.4: Screenshot of Ship Motion Control's SMChms [6].

From this user interface it can be observed that this system has the same capabilities as the Fugro system, and according to the system documentation, also supports:

- Navigation data via GPS;
- Video monitoring of the helideck;
- Full data recording and playback; and

- Automated report generation and transmission.

The system documentation available on SMC's website [6] does not discuss the system hardware setup but does advertise the fact that the inertial sensor used by the system is also a product of the company.

The system appears to suffer from the same disadvantages as the Fugro system. Additionally, there is no apparent implementation of vertical acceleration as an important flight deck motion parameter. Also, the design decision to have the weather information text be the largest on the screen does not appear to have been rationally made, as that information is the least important if monitoring ship motions as it is the least likely to change frequently.

1.2.3 Kongsberg

Kongsberg is a provider of a large number of maritime and defence solutions and services. They have produced a helideck monitoring system that is designed to be used to improve safety in hostile weather conditions [7]. A screenshot of the software is shown in Figure 1.5.

The HMS 100 helideck monitoring system is mainly targeted at offshore floating production and storage vessels, and seismic vessels. The software takes the location of the sensor into account and transforms the measured data to the centre of the helideck. It also includes configuration software with three-dimensional images of the ship to aid in proper configuration of the system. Based on demonstration software available for download, it appears that the main software module communicates with smaller software components that manage communication with a variety of external devices such as data sources and lighting controls. The company's website also indicates that SHOLS and landing period designator versions are available, though it provides no details on what this entails.

From the available screenshot it appears to have been licensed from the same source as the previous user interfaces. It also appears to suffer from attempting to display too

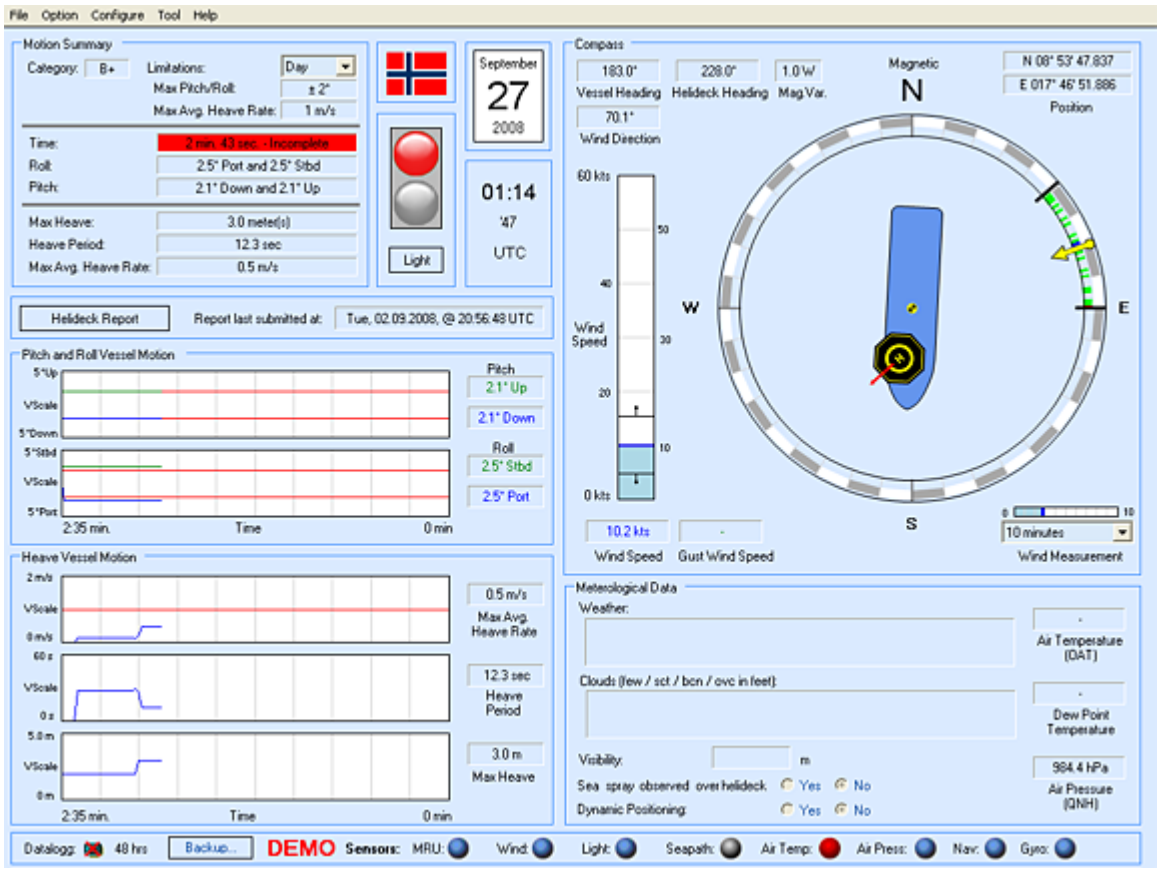

Figure 1.5: Screenshot of the HMS 100 helideck monitoring system [7].

much information on the screen at once.

1.2.4 Miros

Miros Helideck Monitoring Systems (HMS) [8] are designed to give flight deck operators real-time and historical ship motion data and also to provide helicopter crews onshore with information from any helidecks with Miros HMS installed. An image of the Miros HMS user interface is shown in Figure 1.6. In addition to information on helideck motion, the system is also built to provide information from other sources including: weather status, fuel storage, passengers, cargo weights, routing, vessel position and speed, VHF AM frequency, NDB frequency, and ID-code.

Figure 1.6: Screenshot of the Miros HMS [8].

1.2.5 C^2I^2 Systems

Communications Computer Intelligence Integration Systems (C^2I^2) is a company that specializes in real-time systems development, specifically data communications for real-time systems, and is based in South Africa. One of their products is a helicopter take-off and landing system which is designed to aid in all helicopter-ship operations by measuring

and displaying weather conditions and ship's motion data [9]. Images of its user interface are shown in Figure 1.7. Some of the features of this system include:

- Monitoring of ship roll, pitch, heave (vertical speed), and ship speed;
- Monitoring of weather conditions, wind speed and direction, and temperature and pressure;
- Display of radar images;
- Control of flight deck and hangar lights;
- Display of flight deck personnel status;
- Display of video from a flight deck camera; and
- Roll, pitch, and heave limits monitoring.

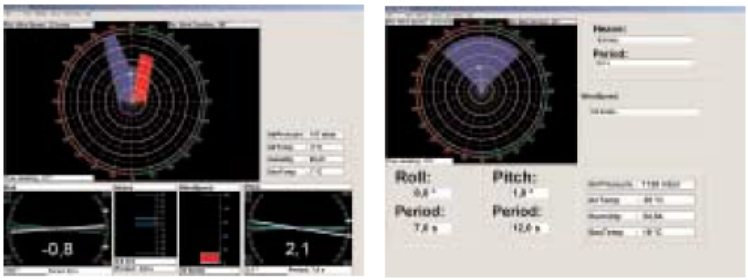

Figure 1.7: Images of the user interface of C^2I^2's helicopter take-off and landing system [9].

While the system brochures state that the system can be used in a stand-alone or ship-integrated configuration, it appears that most of its potential capabilities require inclusion in ship systems, and require specific low-level customization based on the application it is being used for.

1.2.6 Aeronautical and General Instruments Limited

Aeronautical and General Instruments Limited (AGI) has developed a system that performs ship motion monitoring actions that the FDMD has not been designed or implemented to do. Their system displays current wind conditions on a Ships Helicopter Operational Limit diagram, which is a polar plot that shows acceptable relative wind conditions under which helicopter operations can be safely performed. The system is called Ship Helicopter Operational Limits Display System (SHOLDS) [10] and an image of it is shown in Figure 1.8.

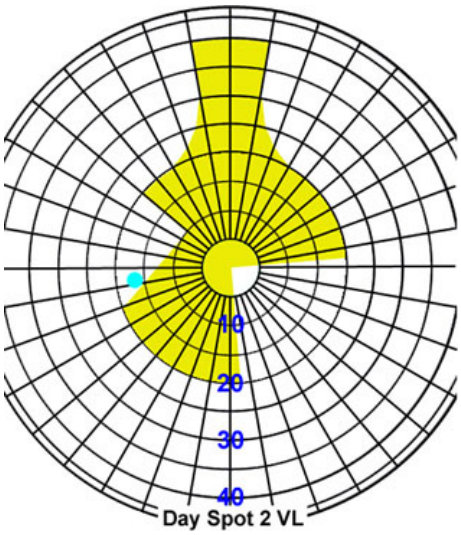

Figure 1.8: Ships Helicopter Operational Limit diagram as part of the SHOLDS system [10]. (Note: colours are inverted for image clarity)

SHOLDS provides a status light which is set to green or red depending on whether conditions are safe or not. Additionally, if the conditions are outside of the acceptable limits, the system can calculate and recommend an alternative ship course and speed.

1.2.7 Prism Defence

Prism Defence has developed an innovative system designed to help manage aviation operations aboard naval vessels. A screenshot of the software's user interface is shown in Figure 1.9.

Figure 1.9: Screenshot of Prism Defence's heliSAFE [11].

From Prism Defence's website [11] it is not clear what exactly the capabilities of its heliSAFE system are. From the screenshot is appears to be able to monitor pitch and roll motions, display radar information, display a camera view of the flight deck, and provide other relevant data on current flight deck conditions.

1.2.8 Siri Marine

The overall focus of Siri Marine's motion monitoring system [12] is different from the helideck monitoring systems presented in the preceding sections. This system is designed to provide real-time information on general ship motions, accelerations, angles, forces, and to generate warnings when the maximum allowable motions or forces on cargoes and structures are reached. The system also records all data to disk for improving future ship operations. These data can also be used, for example, to improve calculations of structural limits and of fatigue lives of various ship components.

A sample user interface screen is displayed in Figure 1.10, though it is difficult to draw any conclusions due to the low resolution of the photograph. The software that has been developed by Siri is called SafetyMax and is designed to help mariners keep better control of their vessels and the cargo on-board, in order to prevent damage. An example of a couple of the SafetyMax modules are: SafetyMax Hull Stress which measures, displays, and logs the deflection and torsion in a vessel, and SafetyMax Tow Monitoring, which transfers the data transmissions from sensors on a towed object wirelessly to a tug, with the status of the tow able to be monitored in real-time on the bridge of the tug.

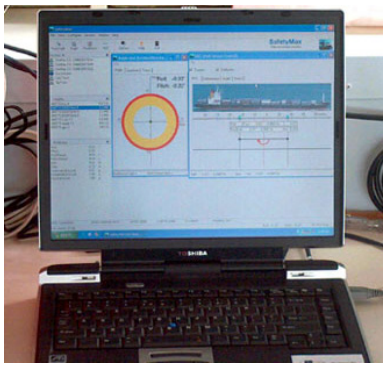

Figure 1.10: Photograph of a computer running Siri Marine's SafetyMax software [13].

1.2.9 HULLMOS

HULLMOS is a system developed by R Rouvari Oy and is a hull motion monitoring system, hull stress monitoring system, and ice impact load monitoring system [14]. The system notifies operators when there is a risk of damage to a hull structure or cargo caused by improper loading or high speed in heavy weather. The system consists of four to six sensors on the main deck, one accelerometer on the bow, and a central unit on the bridge. A screenshot of its user interface is shown in Figure 1.11.

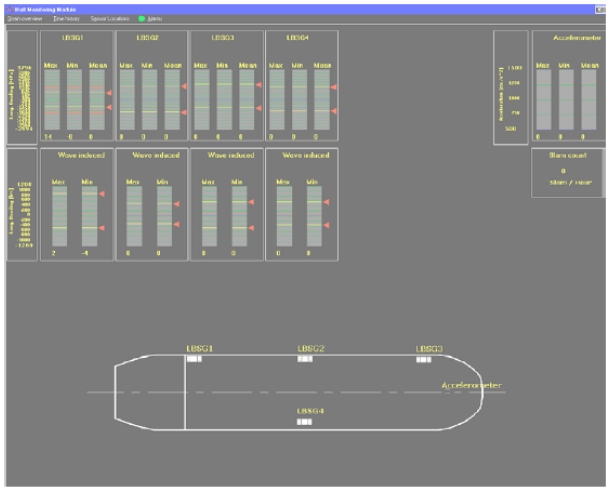

Figure 1.11: Screenshot of HULLMOS' main monitoring screen [15].

1.2.10 Sirehna

Sirehna has designed and developed a complete ship monitoring system including vessel motions, strains, and sea state estimation functionalities [16]. It is based on the HULLMOS system, and adds to it a sea state estimation module which is based on the analysis of ship responses to waves. A diagram depicting their system's operation is shown in Figure 1.12.

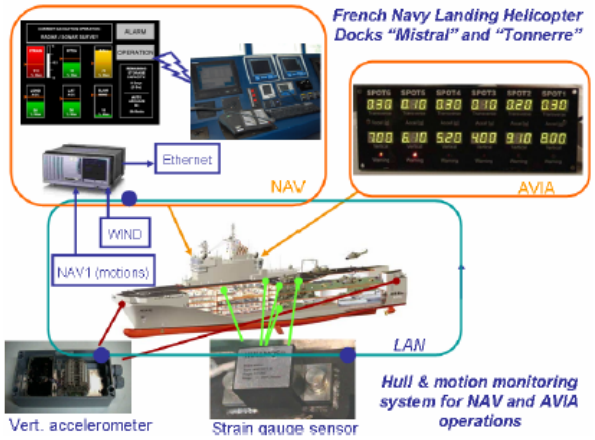

Figure 1.12: Sirehna's complete ship monitoring system [16].

1.2.11 Commercial Systems Comparison

Table 1.1 compares the features of the various flight deck monitoring systems that have been presented in the preceding sections. The symbol '✓' indicates that the system possesses the feature, '×' indicates that it does not, and ' - ' indicates that it is unknown whether the system includes the feature.

1.3 Flight Deck Monitoring Research and Development Activity

Similar to the FDMD project, there are other research projects currently underway involving flight deck monitoring. The following section summarizes some of these efforts.

1.3.1 Landing Period Designator

In the previous sections, many of the systems presented were designed for use in European helicopter-ship operations, where similar to Canada, the flight deck crew monitors ship motions. In the United States, however, it is the responsibility of the pilot to determine when it is safe to land with minor support from the deck crew. In order to improve the efficiency of helicopter landings in that situation, Carico and Ferrier [17] have developed a Landing Period Designator (LPD) system. It consists of a set of lights on top of a ship's hangar which provide information through varying colours/symbols on the quiescence state of the ship. The system has four separate states: green, green-amber, amber, and red. An image of the various system states is shown in Figure 1.13. Red is the only condition that is beyond limits.

This system calculates an 'energy index' (EI) which is a sum of the squares of the angular positions and rates, and lateral/vertical velocities and accelerations of the ship,

Table 1.1: Comparison of motion monitoring systems.

	Fugro	SMC	Kongsberg	Miros	C^2I^2	AGI	Prism	Siri	HULLMOS /Sirehna
Pitch	✓	✓	✓	✓	✓	×	✓	✓	-
Roll	✓	✓	✓	✓	✓	×	✓	✓	-
Vertical position	✓	✓	✓	-	×	×	-	-	-
Vertical velocity	×	×	×	-	✓	×	-	-	-
Vertical acceleration	×	×	×	-	×	×	-	✓	✓
Data histories	✓	✓	✓	✓	-	×	✓	-	-
Weather data	✓	✓	✓	✓	-	×	✓	-	-
Heading	✓	✓	✓	✓	-	✓	✓	-	-
Wind data.	✓	✓	✓	✓	✓	✓	✓	-	-
Min/max motions	✓	✓	✓	-	-	×	-	-	-
Limits	✓	✓	✓	-	-	✓	-	-	-
Limit notifications	-	-	-	-	-	✓	-	✓	-
Multiple data sources	✓	✓	✓	-	-	-	-	-	-
Clear backup features	×	×	✓	×	×	×	×	×	✓
Helideck video monitoring	-	✓	-	-	✓	-	✓	-	-

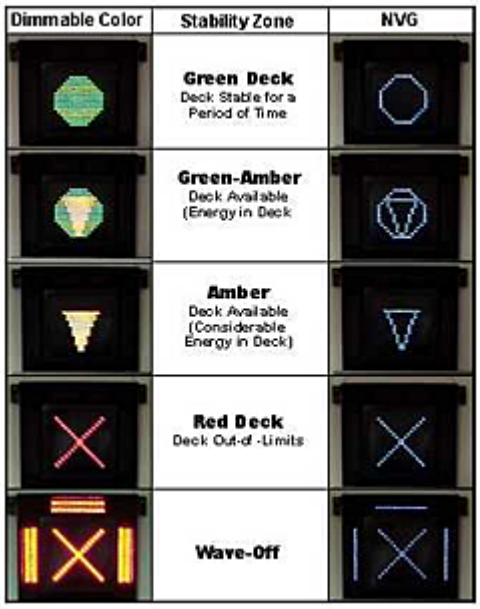

Figure 1.13: LPD display states [17].

each multiplied by an empirically-determined coefficient. This energy index can partially predict future ship motions under the assumption that the ship can only be displaced from a very low energy state to a flight deck out-of-limits state by having a specific amount of energy transmitted to the ship from the sea.

After some pilot testing, Carico and Ferrier came to the conclusion in [17] that the effect of the landing period designator was that it could decrease total hover time by letting pilots land sooner. It also helped to validate the pilot's mental model of the ship's motion, and when accurate, relieved a portion of the pilot's stress of locating or predicting landing periods. On the downside, without any rate or trend information available, the pilots did not know whether the deck was trending towards red or green. This resulted in a tendency to fixate on the LPD display, or to simply ignore it altogether.

In more recent work [18] Carico and Ferrier have improved the LPD display and carried out further trials with helicopter test pilots. Additional lights have been added to the LPD that illuminate during the amber state based on how close the energy index is to amber-

green or red. Images of the improved amber state are shown in Figure 1.14.

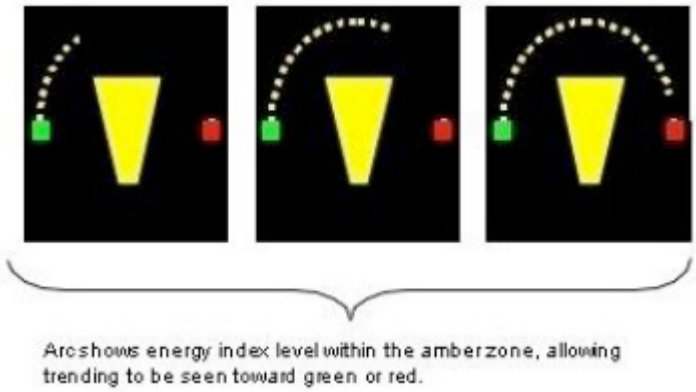

Figure 1.14: Improved LPD with light rates symbology [18].

The most significant results of his tests are that without the LPD most events took place in the amber state, with others being divided among the other states, including red, and that with the addition of the LPD no events took place going into, during, or when leaving a red state. In the end, he is careful to draw specific conclusions from the tests with his general concluding statement being that the LPD was a useful addition to the always-present ship motion cues.

1.3.2 Safety Index

In [19], Gray and MacTaggart present a Safety Index (SI) method. The method begins with a simplified model of the helicopter-deck interface. Measurements of the ship dynamics and environment are continuously read from sensors and introduced to the model. The model is then used to calculate updated forces and moments acting on the ship. These forces are compared to maximum variables of interest such as restraint loads, wheel loads, and tire slip. Each output variable is scaled by the maximum safe value so that each becomes a value between zero and one. The maximum variable value is then taken as the current state of the ship. This value is called the safety index.

Gray ran a number of ship simulations and calculated the corresponding safety index data sets. Notable conclusions were that the maximum roll or pitch did not always coincide with the maximum SI, and that SI values were greatly dependent on the speed of the ship. The purpose of having a single safety index is to give ship operators a single simple value that decisions can be based upon.

Some disadvantages of the SI method include: no way of determining the severity of the particular force being exceeded (ie. landing gear sliding versus failure), the linear scaling of ship variables whose behaviour may not be linear, and the system is generally targetted at situations where the helicopter is on board the ship. Further, the single safety index implies that all operations are limited by the same factor.

1.4 Motivation

Objectively, the primary goal of a ship motion monitoring system designed for helicopter/ship operations is to answer the question "Is it safe to land now?". Such a system should be able to perform this action without predicting future ship motions unless such predictions can be guaranteed to be accurate 100% of the time.

Today's navies are aware of the above statement, and are interested in the potential benefits of a flight deck motion monitoring system. However, such a system must take human factors into account when being engineered, as otherwise it will not be used. Simply put, if a motion monitoring system does not clearly make ones job easier, it will not be used.

The preceding sections have presented that the Canadian Department of National Defence (DND) has expressed interest in installing a flight deck motion monitoring system aboard their helicopter-carrying ships. Many of the available flight deck monitoring systems from around the world have also been presented.

The goal of the Flight Deck Motion Display project at Carleton University is to meet the requirements expected by DND for a flight deck motion monitoring system, to improve upon the deficiencies of currently-available systems specifically for helicopter landings, and to innovate this product such that it is an attractive system to navies around the world.

Observations of the systems presented suggest that they have been engineered to display as much information as possible on one screen, so that flight deck operators can get an overall picture of the complete current condition of the ship and its environment. The system from Kongsberg even advertises in its software startup screen that it is providing "THE FULL PICTURE". Is this however, the best way to achieve the primary goal of a flight deck motion monitoring system?

The motivation of the FDMD project proposes that only the information necessary for specific flight deck operations should be presented to an operator. The visual presentation and the location on the screen of this information also plays an important role in the usefulness of the system.

The primary goal of the Flight Deck Motion Display, is to allow an operator to, in a quick glance, or through their peripheral vision, determine whether the ship's motions are within limits for the current operation or not.

The systems that come the closest to meeting the DND requirements for a FDMD are the ones of similar apperance from Fugro, SMC, Kongsberg, and Miros. Some specific deficiencies in the systems available compared to the DND intended use for them include:

- These systems depend on the user to spend time reading and interpreting numerical values. In addition, the distance an operator can stand away from the monitors is limited by two factors: (1) the size of the display hardware required to display resolutions of at least 1024x768 pixels and (2) the maximum practical size of text on the screen.

- There is no single specific place on the screen an operator should look to determine whether ship motions are within their limits or not. A user either has to check that all of the numerical values are below their limits, or check all of the graphs to ensure that current motions are within the marked limits.

- None of the systems have been designed taking into account that accelerations may be used as a primary motion parameter. They are built around heave position or velocity instead.

- These systems receive information from a number of sources, yet have no clear plan of action in case of failure of an essential component.

- A mouse and likely a keyboard are required to navigate the program outside of the displayed information. The use of these devices is impractical in the applicable at-sea environment.

1.5 Project Objectives

Given the primary goal of the FDMD project, a list of thesis objectives was compiled. They are as follows:

1. Perform a literature survey to locate as many commercial flight deck monitoring systems as possible in order to ensure that the FDMD is competitive and innovative.

2. Research a range of possible computer hardware components and select components appropriate for system demonstrations and testing within the project budget.

3. Select and iterate a hardware configuration for the FDMD that can provide the simplicity of a single sensor and computer, and also be extended to provide redundancy and backup functionality.

4. Select and detail design a network configuration for the FDMD software which supports the selected hardware architecture, and can also be expanded to include other computers and data measurement devices. Special care should be taken to ensure that the arrangement is also simple enough that a minimal amount of time must be spent planning for all possible network behaviours and debugging the software.

5. Design a real-time user interface which meets the design requirements of the DND draft specification while also incorporating a quiescent period indicator as proposed by DRDC.

6. Design and iteratively develop a quiescent period indicator which allows a flight deck operator, in one glance, to determine the current state of the ship, which motion parameters are dominating the ship's movements, and whether the ship's motions are tending to move closer to or farther away from quiescence.

7. Perform a variety of laboratory tests to ensure that the FDMD system is operating correctly in at least simulated ship conditions.

8. Carry out a trial installation of the FDMD or an evaluation with actual flight deck operators in order to ensure meaningful system refinement.

1.6 FDMD Prototype Design Requirements

Based on the draft requirements specification released by DND [4], the deficiencies in currently available systems discussed above, and the goal of the FDMD project to allow an operator to quickly determine a ship's state, a set of design requirements was decided upon for the system's capabilities, hardware properties, and software features. These requirements are presented below.

1.6.1 Prototype Capabilities

The FDMD prototype was required to possess the following capabilities:

- As a primary function, provide flight deck motion information to the LSO by indicating in real-time, the status of flight deck quiescence (as currently defined by DRDC) for flight deck recovery only;
- As a secondary function, support mission planning by displaying relevant historical ship motion data;
- Comprise a complete stand-alone ship motion measurement instrumentation system (with the exception of power) including one sensor, one computer system, and one display;
- Demonstrate limited operation of five operating modes: startup, deck operations, flight operations, data display, and maintenance and test mode;
- Demonstrate FDMD expansion capabilities to include all required flight operations and deck operational tasks; and
- Be able to be installed entirely in an LSO compartment, which will require translation of sensor data to flight deck motion parameters.

1.6.2 Hardware Requirements

The prototype display that would be located in the LSO compartment must:

- possess dimensions not exceeding 12 in. (width) x 9 in. (height) x 3.5 in. (depth);
- have a screen size of at least 8 in. (width) x 6 in. (height); and
- be able to be controlled with programmable bezel keys.

1.6.3 Software Requirements

The system software must satisfy the following requirements:

- Run under a windows-based operating system;
- Be written in C++;
- Be designed such that it is modular, scaleable in hardware interaction, and stable;
- Be written to follow CU/GDC mutually agreed upon coding standards consistent with the available project schedule and budget;
- Provide dedicated display screens for:
  - Connectivity status;
  - Mission planning or statistics;
  - Real-time operator guidance;
  - Data display and archiving; and
  - Configuration and testing;
- Be formally verified and validated consistent with the available project budget and schedule; and
- Provide options for re-configurability through input files.

1.7 Thesis Overview

This chapter provided an overview of the Flight Deck Motion Display project. It gave an overview of flight deck operations, the design specifications and requirements that went into the project, currently available commercial systems used in flight deck operations,

and recent research performed on the subject of ship motion monitoring. The next three chapters describe in detail the design, implementation, and testing of the FDMD. Chapter 2 describes the hardware selected for the FDMD prototype. Chapter 3 describes the software developed for the FDMD. Chapter 4 describes the in-house testing and external user interface evaluations of the FDMD. Chapter 5 discusses some aspects of the FDMD's design, implications of the results of the evaluation, and an outlook on the future of the project.

1.8 Contributions of the Thesis

During the course of conducting the work described in this thesis, the following significant research and development contributions have been made.

1. A comprehensive review of commercial ship deck motion monitoring solutions and systems was conducted and summarized.

2. Hardware and software elements of a versatile flight deck motion monitoring system were designed, implemented, and tested, resulting in a prototype FDMD system uniquely tailored to LSO requirements, and capable of being expanded to other applications.

3. A novel ergonomic quiescent period identification user interface was iteratively developed with input from the operational community. The use of multiple quiescent states, separate limits for entering and exiting quiescence, and physical characteristics of the user interface indicator were explored.

4. A structured pilot/LSO evaluation of the quiescent period indicator was conducted and the results of the evaluation analyzed and summarized.

Chapter 2

Hardware

2.1 Introduction

This chapter provides a detailed description of the FDMD prototype hardware design and implementation. Research into, and selection of, proper hardware for the FDMD prototype system, has played a major role in its development. Being able to present the prototype system using hardware similar to what would be used in the final product is very important for discussing the system with potential operators during system demonstrations. Further, in the event of at-sea demonstrations, equipment ruggedness, safety, and non-interference requirements must be satisfied. This section begins with a description of the hardware architecture design, and follows with details on the selection of an inertial sensor and computer hardware. Additional details on the hardware selection are available in the FDMD Prototype Hardware Report [20].

2.2 Hardware Architecture

The simplest implementation of the FDMD is as a two-component system, with a primary sensor located at the hauldown point of the flight deck, connected to a rugged computer

and display located in the Landing Signals Officer (LSO) compartment. The disadvantages of this system are that if one component fails, then the system is completely unusable, and if the sensor is malfunctioning, there is no way to determine whether there is a fault in its operation (aside from cases where such a fault would be obvious). Thus the only solution that ensures redundancy in the FDMD system is that which consists of more than one computer and more than one sensor.

The design of the system envisioned by DRDC [3] and DND [4] both consist of only a single sensor, connected to a computer system nearby, which is then connected over a long distance to a display of some sort located in the LSO compartment. This system suffers from the same deficiency as the two-component system, with the additional complication of either having to transfer video over a long distance, or to deal with possible latency issues involved in the transfer of data between computers.

The design selected for the FDMD prototype is a four-component system, with two computers and two sensors connected serially. A diagram of this configuration is shown in Figure 2.1. The system consists of an inertial sensor located at the hauldown point which is directly connected to a portable computer or smart display located in the LSO compartment running the FDMD client software. This computer is then connected over a network to another computer running the FDMD server software which is also connected to a backup sensor. The difference between the FDMD client and server software is that the client version is designed to be used on less powerful computing hardware, to only receive data from one sensor source, and to not be required to save data to disk.

The system's primary functionality begins with sensor measurements made at the primary sensor. These measurements are transferred to the smart display where the information necessary for flight deck operations is immediately interpreted and displayed for the operator. The data are then transfered over the network connection to the FDMD server where it is recorded to disk. The direct connection between the primary sensor and

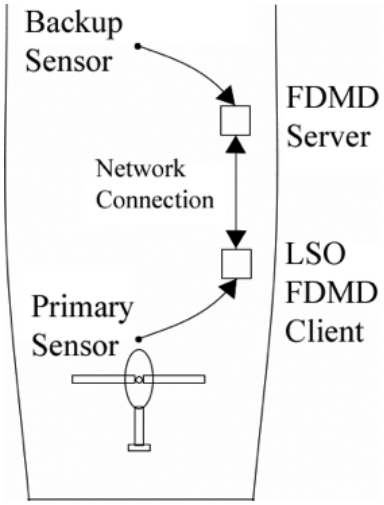

Figure 2.1: Four-component hardware architecture of the FDMD prototype.

the computer system in the LSO compartment ensures that motion data updates occur as quickly as possible.

The system's redundant functionality is provided by the backup sensor connected to the FDMD server computer. Its measurements are compared with the primary sensor's measurements in order to ensure that both sensors are operating correctly. The data from each sensor are transformed to the hauldown cable location before comparison.

Backup functionality is provided by the presence of both a second computer and sensor. By default, backup sensor data are only processed by the FDMD server computer. In case of a primary sensor failure, the system can be enabled to transfer the backup sensor data to the FDMD client computer. In case of a failure of the smart display, the FDMD server computer can still be monitored. It is also possible to record data at the FDMD client computer in case the network connection is not available or if additional recording redundancy is required.

The main advantages of the four-component system include:

- Ideal cable usage with network cables used to cover the longest distances;

- Backup sensor can be of lesser quality than the primary sensor if one wishes to reduce hardware costs;
- Identical systems could potentially be used for primary and backup systems for interchangeability;
- Only a single connection is placed between the primary sensor and the FDMD client, ensuring that data frequency updates are not limited by communication distances;
- Accuracy of the sensor data can be verified;
- Backup data are present in case of primary sensor failure; and
- The failure of the computer system in the LSO compartment does not prevent data from being recorded.

The main disadvantages of the four-component system include:

- The system requires components capable of running two separate FDMD systems;
- Both computer systems require a reasonable amount of computing power in order to process the sensor data; and
- With only two sensors, determining which one is inaccurate can be difficult.

Some additional points about the system design include:

- A keyboard and mouse will not be necessary in the LSO compartment but will be required for the initial configuration of the FDMD and some of the FDMD server's functionality; and
- While the design does not require the smart display to be able to record data to disk, if data recording redundancy is required, then computer hardware can be chosen that will allow this functionality to be included.

2.2.1 Hardware Installation Considerations

Figure 2.2 shows the LSO console on the HMCS Montreal which has recently been updated by General Dynamics Canada. Marked with a white square is a potential location for the FDMD to be installed.

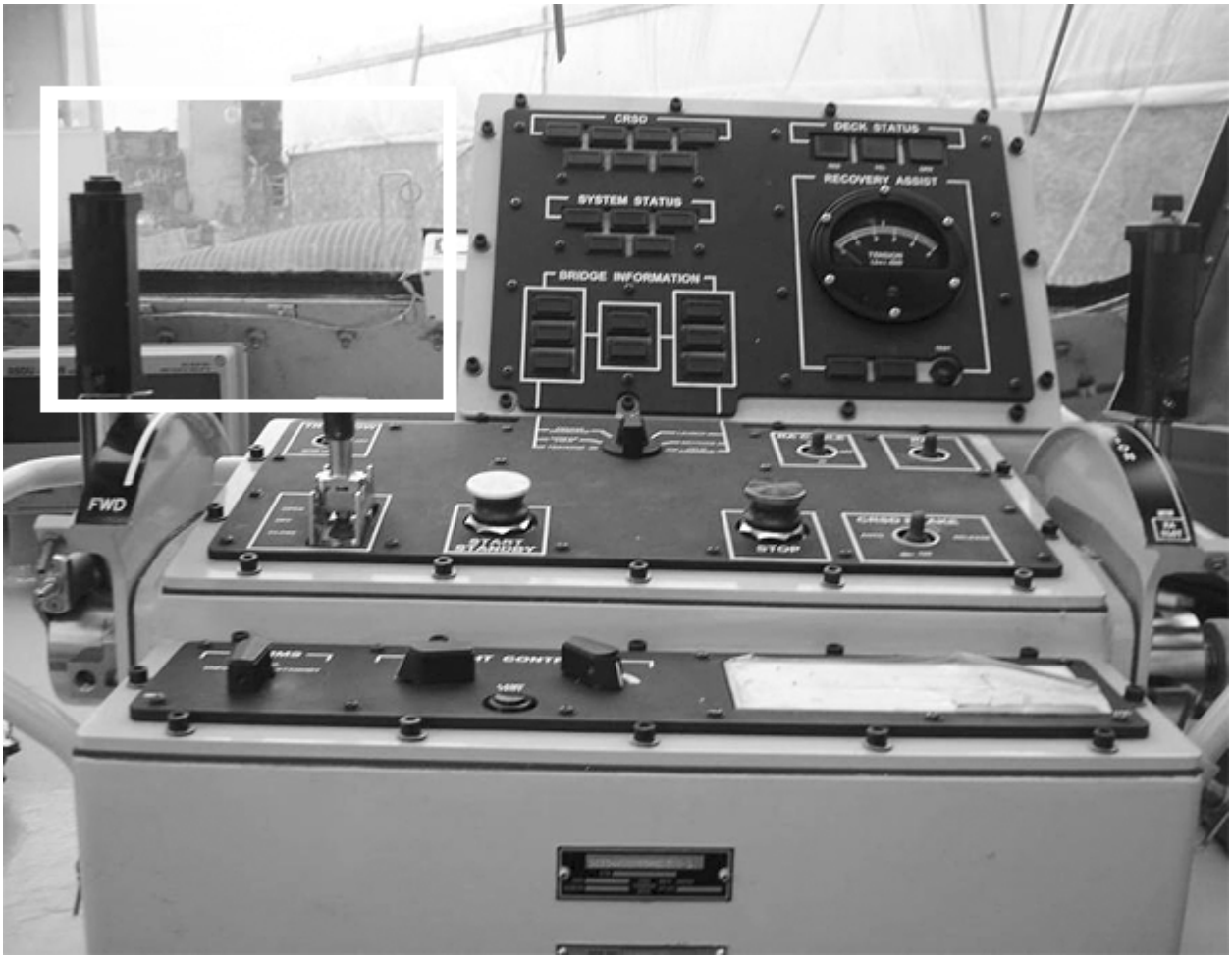

Figure 2.2: Updated LSO console aboard the HMCS Montreal.

2.3 Inertial Measurement Unit

The hardware design of the FDMD requires at least two inertial sensors. Most inertial sensors contain at least accelerometers for measuring linear accelerations and gyros for measuring angular velocities. In order for all of the FDMD's functionality to be available it is necessary to measure or derive roll angle, pitch angle, all angular velocities, all angular accelerations and all linear accelerations.

There were few strict requirements for the primary inertial sensor of the FDMD. In early design it was specified to be a "high quality" 6-dof sensor package. The DND draft design specification [4] requires an inertial sensor which measures accelerations to full-scale range of not less than ±1g and to an accuracy of ±0.002g or better, velocities to a full-scale range of not less than ±10°/sec and to an accuracy of ±0.2°/sec, and angular positions to a full-scale range of ±30° roll, ±10° pitch, and to an accuracy of ±0.6° or better.

2.3.1 High-Quality Sensor Package

In order to reduce the price of the primary sensor, a high-quality non-rugged sensor package was selected[1]. Crossbow's Attitude and Heading Reference System (AHRS) AHRS400MB was selected, which combines the functions of a vertical gyro and directional gyro in order to provide measurements of roll, pitch and azimuth angles. An image of the front of the sensor and its cable connector is shown in Figure 2.3.

The AHRS is designed to be used in applications such as platform stabilization, UAV control and avionics. The 'MB' designation identifies that this model is specifically designed for use in marine-based operations. This means that applying power to the sensor at sea where it will immediately be exposed to motion will not disrupt its startup process. The AHRS uses solid-state micro-machined (MEMS) angular rate and linear acceleration

[1] A rugged sensor meets military requirements for durability and electromagnetic interference.

Figure 2.3: Crossbow AHRS400MB-200 inertial sensor.

sensors to measure 3-axis accelerations and 3-axis angular rates. A 3-axis magnetometer also allows the AHRS to measure true heading. A Kalman filter algorithm corrects for dynamic errors and drift in measurements and calculations. The most comprehensive operating mode of the AHRS provides the following measurements:

- Roll, pitch, and heading angles;
- Roll, pitch, and yaw angular rates;
- X-axis, Y-axis, and Z-axis accelerations;
- X-axis, Y-axis, and Z-axis magnetic fields;
- Temperature sensor voltage; and
- Relative elapsed time.

The AHRS400MB communicates with a computer over an RS-232 serial connection. A software program called Gyroview is provided by Crossbow for displaying and recording the sensor data on a computer. For use with the FDMD, custom C++ software has been

written to communicate with the sensor. A summary of some of the capabilities of the inertial sensor is shown in Table 2.1. Specifications given as equalities are worst-case measurements.

Table 2.1: Summary of AHRS400MB-200 technical specifications.

Specification	Value
Performance	
Update Rate (Hz)	> 50
Stabilized Startup Time (sec)	60
Attitude	
Range: Roll (°)	$\pm$ 180
Range: Pitch (°)	$\pm$ 90
Static Accuracy (°)	$< \pm$ 0.75
Dynamic Accuracy (°)	$\pm$ 2.5
Heading	
Range (°)	$\pm$ 180
Static Accuracy (°)	$< \pm$ 2
Dynamic Accuracy (° rms)	$\pm$ 4
Angular Rate	
Range: Roll, Pitch, Yaw (°/sec)	$\pm$ 200
Bias: Roll, Pitch, Yaw (°/sec)	$< \pm$ 0.05
Acceleration	
Range: X/Y/Z (g)	$\pm$ 4
Bias: X/Y/Z (g)	$< \pm$ 0.012
Environment	
Operating Temperature (°C)	-40 to +71
Non-Operating Vibration (g rms)	6
Non-Operating Shock (g)	1000
Electrical	
Input Voltage (VDC)	9 to 30
Power Consumption [at 12 VDC] (W)	< 4
Physical Properties	
Size (in)	3.0 x 3.75 x 4.1
Weight (lbs)	< 1.7

2.4 Computer Hardware

An overall goal in prototype hardware selection for the project has been to purchase equipment that is as rugged as possible, as the end product will be used in military applications. The DND draft specification [4] consists of requirements from the following documents:

- MIL-STD-461E, requirements for the control of electromagnetic interference characteristics of subsystems and equipment;
- D-03-003-007/SG-000, specification for design and test criteria for shock resistant equipment in naval ships;
- D-03-003-019/SG-000, standard for vibration resistant equipment;
- MIL-STD-464A, interface standard, electromagnetic environmental effects requirements for systems;
- MIL-STD-3009, military standard 2009, lighting, aircraft, night vision imaging system compatibility; and
- MIL-STD-1787B, aircraft display symbology.

Attempts have been made to match the specifications in these documents to the extent possible within the constraints of the project's budget, as well as the limited availability of some of the documents themselves. The following sections present the hardware that has been selected for the FDMD prototype.

2.4.1 Hardware Requirements

In order to remain within the project's budget it was decided to ensure that the equipment located in the LSO compartment would be as rugged as possible, so that during demon-

strations (which only require a sensor and single computer) computer equipment as similar as possible to the final product would be able to be presented. Hardware requirements for the "smart display" located in the LSO compartment were derived from the MHP draft specification and from the hardware architecture of the four-component system. The physical requirements for the smart display were presented in Section 1.6.2, and it also must:

- have an ethernet port for communicating with another computer;
- have at least one USB port dedicated to the connection of a keyboard/mouse;
- have either an RS-232 serial port or a USB port for a connection to an inertial sensor; and
- have sufficient processing ability to run the FDMD client software.

Less rugged requirements are necessary for the FDMD server computer as it would be located in the interior of the ship, requires more processing power than the FDMD client computer, and would not be used in as many FDMD demonstrations. The server PC must:

- have an ethernet port for communicating with an FDMD client computer;
- have at least one USB port for connecting a keyboard/mouse;
- have either a serial port or an additional USB port for receiving data from an inertial sensor;
- be able to output to a VGA monitor; and
- have sufficient processing ability to run the FDMD server software.

2.4.2 Hardware Availability

A variety of companies were contacted in the search for computer hardware that fulfilled the above requirements. The results of this search can be divided into five categories.

1. Industrial PCs - this category consists of smart display PCs that are either not weather resistant at all or only resistant from the front and designed to be installed in a rack. Prices typically range from $2000 to $6000.

2. Environment resistant industrial PCs - this category consists of sealed environment-protected PCs that would likely be able to fulfill most of the ruggedness requirements for the LSO compartment PC. Prices typically range from $3000 to $12,000.

3. Industrial rugged tablet PCs - this category consists of tablet PCs which are typically designed for a variety of industrial practices such as use in factories or in vehicles. They range in cost from $3000 to $10,000.

4. Marine grade PCs - this category includes PCs and displays designed to be installed on seagoing vessels, for personal or commercial uses. They typically are constructed with corrosion-resistant electronics but may or may not be splash-proof. They range in cost from $2000 to $8000.

5. Military grade PCs - this category consists of displays and computers built to military ruggedness standards. Prices typically range from $15,000 to $40,000 for the computer and $10,000 to $15,000 for the display.

It was determined that in order to acquire a computer system that satisfied all of the requirements for the LSO compartment smart display computer that it would be necessary to (1) select a military grade display and (2) spend over $20,000 on the computer and display. Many of the industrial computers investigated provided sufficient environmental

ruggedness for the prototype, but only military models possessed both the ruggedness and the bezel keys instead of a touchscreen option. The following sections present the hardware chosen for the FDMD in order to satisfy the compromise between ruggedness and cost.

2.4.3 FDMD Client Computer Hardware

Military-grade hardware for all components of the FDMD prototype was beyond the available budget for the project. In order to compromise on the matter, a rugged display and computer were selected for the hardware components located in the LSO compartment, and marine-grade components were selected for the FDMD server role.

10.4" AMLCD Color Video Display

A rugged computer display was purchased from General Dynamics Canada and a schematic of it is shown in Figure 2.4. The unit has an 8.3" (211 mm) by 6.2" (157 mm) viewing area, supports up to a resolution of 800 x 600 pixels and can be viewed from ± 50 degrees on either side and from 45 degrees above and 20 degrees below. Its dimensions are 9.45" (height) x 11.26" (width) x 1.66" (depth) and it weighs a maximum of 8.25 lbs. In addition to the VGA video display it also supports two channels for displaying video from cameras. It has a touchscreen that communicates over USB and sixteen bezel keys that communicate over an RS-422 serial connection.

This display requires a 28 VDC power source to operate. The FDMD prototype has been set up to be powered from a variable output laboratory power supply.

Xplore iX104C3 Rugged Tablet PC

The Xplore rugged tablet PC that was selected to run the FDMD client software provides flexibility in its use in FDMD demonstrations and implementations. The advantage of a touchscreen client FDMD computer is that it adds the option of having the FDMD be

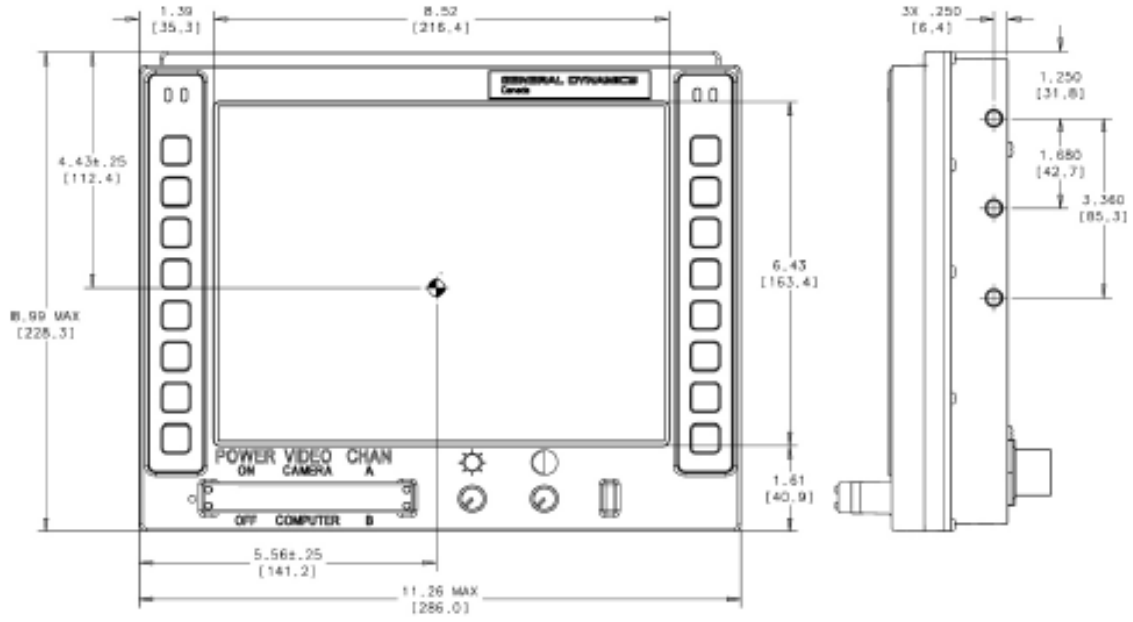

Figure 2.4: Front and side view of rugged display unit purchased from General Dynamics Canada.

controllable using the touchscreen in addition to or instead of the GDC display in case it is not available or impractical to demonstrate. An image of the computer is shown in Figure 2.5. Technical specifications are listed in Table 2.2. Some of its key features are listed below.

- A triple-layer magnesium housing with customized gaskets and a bumper system enables it to survive in dangerous work environments that may expose it to vibration, extreme temperatures, moisture, dust, or drops to concrete.
- It includes Xplore's custom and award-winning display technology, AllVue, which allows the display to be easily viewed in both direct sunlight and office lighting environments.
- The touchscreen features Xplore's dual-touch technology, which allows the touchscreen to be manipulated both with fingers and the included stylus.

- The system is completely mobile, and can provide up to several hours of use with a lithium ion battery.
- The tablet has been third party tested and warranted to meet US military standard MIL-STD-810F.

Figure 2.5: Xplore iX104C3 rugged tablet PC.

2.4.4 FDMD Server Computer Hardware

It was decided that the FDMD server computer should be at least partially resistant to environmental exposure so a marine-grade computer and display were selected.

Argonaut Avalon Mini-PC

The Argonaut Avalon is a marine grade mini-computer. It has a shock resistant mounting and corrosion resistant electronics. The major advantages of this PC are its heavy-duty pentium 4 processor and a large selection of I/O ports. An image of the computer is

Table 2.2: Xplore iX104C3 tablet PC technical specifications.

Processor speed	1.4 GHz Pentium M
RAM	1.0 GB
10/100 RJ45 ethernet	Yes
15-pin VGA	Yes, up to 1024 x 768
RS232 serial ports	0
USB Ports	2
Audio	1 headphone, 1 mic
Power requirements	100-240V AC
Hard disk	80 GB
Dimensions	11.20" x 8.25" x 1.60"
Weight	4.9 lbs
Operating temperature range (°C)	-20 to 60
Approximate price	$3000

shown in Figure 2.6 and its technical specifications are listed in Table 2.3. The hardware purchased for the FDMD includes a "pro" upgrade which maximizes the system's RAM, hard drive space and optical drive capabilities.

Argonaut Tflex G212PST LCD Monitor

A touchscreen-enabled splash-resistant LCD display was purchased from Argonaut in order to simplify hardware interoperability between the FDMD server computer and display. An image of it is shown in Figure 2.7. The model possesses a 12" LCD screen that supports up to a resolution of 1024x768 pixels and has the touchscreen added as a separate upgrade. Some of its features include:

- Sunlight readable transflective LCD;
- Wide side angle visibility;
- SunTouch sunlight readable touch screen; and
- Internal corrosion resistant electronics.

Figure 2.6: Argonaut Avalon mini-computer, front and rear views.

Table 2.3: Argonaut Avalon mini-PC technical specifications.

Processor speed	2.8 Ghz
RAM	2 GB
10/100 RJ45 ethernet	Yes
15-pin VGA	Yes
RS232 serial ports	1
Parallel ports	1
USB ports	3
Firewire ports	3
Audio	1 headphone, 1 mic
Power requirements	100-240V AC
Hard disk	160 GB
Optical drive	DVD-R/RW
Dimensions	5.83” x 10” x 3.3”
Weight	6.5 lbs
Operating temperature range (°C)	5 to 35
Approximate price	$2500 with pro upgrade

Figure 2.7: Argonaut Tflex sunlight readable touchscreen LCD monitor.

2.5 Summary

This section has presented the prototype hardware that is used in the FDMD prototype. In summary, these components include:

- Crossbow AHRS400-MB inertial sensor to act as primary motion sensor.
- General Dynamics Canada AMLCD rugged display for ideal user-computer interface in the LSO compartment.
- Xplore rugged tablet PC to run the client version of the FDMD in the LSO compartment.
- Argonaut marine-grade mini PC to run the server version of the FDMD.
- Argonaut Tflex LCD touchscreen splash-resistant monitor for displaying the FDMD server.

Chapter 3

Software

3.1 Introduction

This chapter describes in detail the software designed and developed for the FDMD. It features five operating modes: communications, data management, operations, statistics, and test/configuration. Each of those operating modes, has a corresponding software module and user interface in the FDMD. Each module is designed to be independent of the others, in order to provide flexibility in future development of the software.

This chapter documents the design and operation of each module, and how the modules are interconnected. Specific details on class structure, data processing algorithms, communication protocols, and other related topics are provided. Additional details on the software are available in the FDMD Prototype Software Report [21].

3.1.1 Software Overview

The FDMD software is divided into five modules plus a set of base classes. Each module has its own user interface, and in general is designed to be able to operate independently of the other modules. The five modules as shown in Figure 3.1 are: communications, data

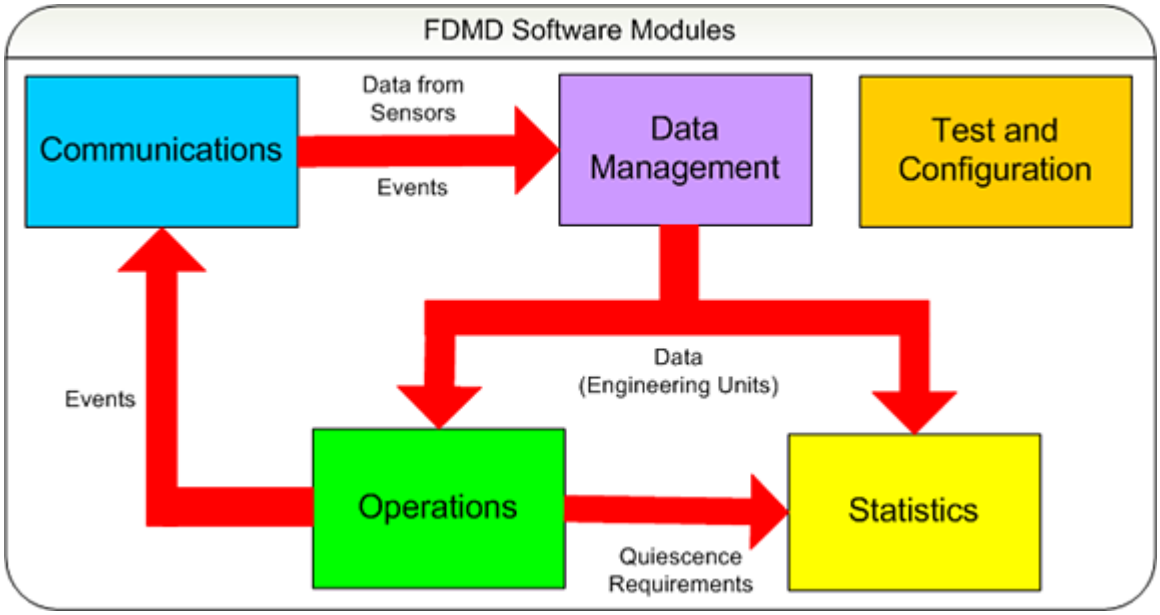

Figure 3.1: FDMD software modules.

management, operations, statistics, and test/configuration.

A short description of each module is given below.

- Communications - manages all of the data input and output that involves serial and network connections to other devices. Also monitors the status of all hardware components in the system.
- Data Management - manages all data recording and playback, conversion of raw data to engineering units, and comparison of data from different sources.
- Operations - all real-time data monitoring functionality is managed by this module. Also includes and performs all quiescent status calculations.
- Statistics - responsible for calculating various useful flight deck statistics.
- Test and Configuration - provides an interface for configuring the FDMD, and more detailed system monitoring services than the communications module.

Each FDMD module is responsible for loading and managing the data files related to its function. Each module also has its own user interface screen in the FDMD. The general flow of information in the system is that data arrive at the communications module, are transfered to the data management module for conversion to engineering units, are then transferred to the operations module for quiescence status updates and real-time display, and are simultaneously transferred to the statistics module to update the statistics calculations. Data events are generated by an operator using the operations module, which are transferred to the communications module for transfer to other locations, and then passed to data management for recording. The following sections detail the structure and functionality of each module, and how they operate together to perform the primary functions of the FDMD.

3.1.2 Design Considerations

The FDMD software is designed to be more than a program that simply displays the motion measurements made by an inertial sensor to a flight deck operator. It has been specially engineered with emphasis on the software being modular, scalable, and stable.

The FDMD can operate with only one of its modules operating, but base functionality is provided by the combination of communications, data management, and operations. The FDMD is designed to be modular so that on top of these three operating modes, additional ones can easily be added, and integrated such that they do not interfere with the operation of existing modules, and do not require any changes to be made to their current operation. For example, additional data processing operating modes could be added and would simply need to be connected to the mechanism that transfers data from the data management module to the operations module. The statistics module is an example of this. Alternatively, a new module could be added for gathering motion data from new types of sources, and as long as the data were stored in the correct format, it

could be passed directly to the data management module.

The FDMD is designed to be scalable in such a way that it can easily be expanded to include additional FDMD clients/servers or additional sensors. Various configurations were considered which changed the way that FDMD computers communicated with each other and distributed motion data between themselves. The design selected is the simplest one that allowed the flexibility of any number of measurement devices and FDMD consoles to be able to communicate with a single FDMD.

In the design and implementation of the FDMD software, stability was emphasized over performance and convenience. Some of this was provided by the communication methods of the Qt C++ libraries, which are introduced in Section 3.1.4. Careful attention was also paid to which software objects were to be dynamically created in memory, as this can often cause programs to crash.

3.1.3 FDMD User Interaction

A sample screenshot from the FDMD is provided in Figure 3.2. In the image, black squares with labels are arranged along the sides of the screen. These squares are buttons used to control the FDMD, and are referred to in this report as 'software buttons'.

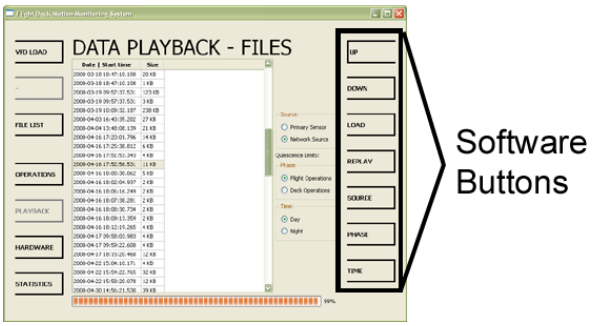

Figure 3.2: Screenshot of the FDMD with right-side software buttons emphasized.

There are three possible methods available for activating the FDMD's software buttons:

1. clicking them with a mouse;

2. touching them with a finger or stylus when using a touchscreen monitor; and

3. pressing a bezel key on the AMLCD military-grade display.

A photograph of the AMLCD display is shown in Figure 3.3. It features eight programmable buttons on each side of the display, seven of which can be labelled by positioning corresponding software buttons next to them. Signals are transferred from the display to a computer over an RS-422 serial connection. Interface code written in C++ was not available from the manufacturer, so the software that processes messages from the display is part of the FDMD.

Figure 3.3: Military display supported by the FDMD.

3.1.4 Qt Framework

The FDMD prototype uses the Qt framework classes created by Trolltech [22]. "Qt is a cross-platform application framework for desktop and embedded development. It includes an intuitive API and a rich C++ class library, integrated tools for GUI development and internationalization, and support for Java and C++ development" [23]. In the FDMD, Qt classes are used for almost everything, from user interfaces to storage classes, to robust

communication between software components using Qt's unique signals and slots functionality. There are commercial and open source licenses available for Qt development. For the FDMD project, a commercial license was purchased.

Signals and slots are used for communication between Qt objects, and are used in the FDMD. Signals and slots are functions defined in classes and can be connected to each other if they share the same argument data types. A Qt macro is used to connect a signal to a slot, so that when a signal in one object is 'emitted' with the data to be transfered, the function in the object designated as the slot is called. Signals can also be connected to other signals.

Multiple signals can be connected to single slots, and single signals can be connected to multiple slots. The major benefit of using signals and slots is that if the communication fails, for example in a case where the receiving object has been deleted, no software error occurs. This makes communicating with signals and slots very stable. Additional details on the Qt framework and signals and slots can be found in the extensive Qt documentation that is available directly from Trolltech [3].

3.1.5 Development Environment

The FDMD is written in C++ and has been developed using Microsoft Visual Studio 2005 in the Windows XP SP2 operating system. The Qt libraries are used in addition to the Windows win32 programming libraries. Experience confirms that Qt integrates well into Visual Studio.

3.1.6 Motion Parameter Identification

In the FDMD configuration files, each motion data parameter has a unique text identifier. In the FDMD software, each one also has a unique numerical index. These names and indices are listed for reference in Table 3.1.

Table 3.1: Motion parameter identifiers and indices used in the FDMD and its configuration files.

Name	Description and Units	Index
xAccel	X-axis acceleration [G]	0
yAccel	Y-axis acceleration [G]	1
zAccel	Z-axis acceleration [G]	2
rollRate	Roll angular velocity [deg/s]	3
pitchRate	Pitch angular velocity [deg/s]	4
yawRate	Yaw angular velocity [deg/s]	5
rollAngle	Roll angle [deg]	6
pitchAngle	Pitch angle [deg]	7
yawAngle	Yaw angle [deg]	8
temperature	Inertial sensor temperature [deg C]	9
time	Inertial sensor time [counts]	10
timeTag	Inertial sensor time tag [counts]	11
rollAccel	Roll angular acceleration [deg/s^2]	12
pitchAccel	Pitch angular acceleration [deg/s^2]	13
yawAccel	Yaw angular acceleration [deg/s^2]	14

3.1.7 Coordinate Systems

The coordinate systems of the ship and inertial sensor used in the FDMD are shown in Figure 3.4.

The ship axes are the standard ones used widely in ship-helicopter dynamics with the origin located at the centre of rotation[1] of the ship. The sensor axes correspond to the sensor having its Y-axis aligned with the ship's Y-axis and the origin placed at the centre of gravity of the inertial sensor. Technically, the inertial sensor could be rotated so that its X and Z-axes are also aligned with the ship's axes. However, this requires a coordinate transformation of orientation measurements so there is no benefit in doing so.

[1]The centre of rotation is defined as the intersection of a vertical line passing through the ship centre of mass and the water plane

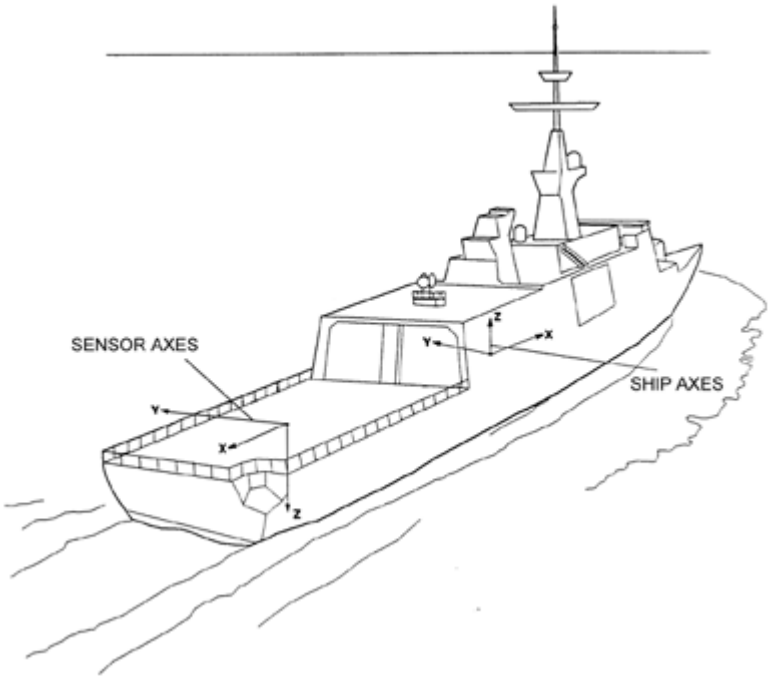

Figure 3.4: FDMD coordinate systems.

3.2 Motion Data Storage Classes

Outside of the functionality provided by each of the FDMD's modules, are a set of 'base' classes which provide functionality available to all parts of the FDMD. The structure of a few of those classes is described here.

3.2.1 BaseDataPacket

The BaseDataPacket class is the simplest motion data storage class in the FDMD. Any data received from an external source is first placed into a BaseDataPacket object.

The BaseDataPacket class has three main storage items: (1) source and destination information, (2) a timestamp, and (3) raw data in bytes. The timestamp is separate from the raw data because it is assigned to the packet after it has arrived at the FDMD. Raw data are stored in 16-bit integers, which means that each data value takes up two bytes.

Data are typically accessed by assigning a quint16 pointer to the memory address of the start of the raw data, which allows each datum to be accessed by using the quint16 pointer as an array.

BaseDataPacket is used for data communication between different FDMD instances, between the communication and data management modules, and for saving data to disk. Saving data to disk slightly modifies the format by placing the timestamp into the raw data storage area in order to simplify read/write operations.

3.2.2 BaseMotionData

The BaseMotionData class stores motion data in engineering units for a single instant in time. The motion parameters stored correspond to the contents of Table 3.1, from index 0 to index 11. This class has been customized to match the data output of a Crossbow inertial sensor, and includes all of the data measurements required by the FDMD operations module.

3.2.3 BaseQpiDataPoint

The BaseQpiDataPoint class is used to represent the quiescent status of one or more motion parameters at an instant in time. This class is used differently from the data storage classes in that there is not a QPI data point for each regular data point. QPI data points are in general only generated when there is a change in quiescence or when a peak in the motion data takes place. The benefit of this is that it reduces the amount of data that need to be processed by related display elements and overall makes the program run faster. The exception to this is the monitoring of acceleration parameters. In order to achieve continuous movement in the quiescent period indicator for vertical acceleration, it is necessary to continuously generate quiescent status data points.

3.2.4 Lightweight Classes

Lightweight classes contain "Lw" in their class name and all have a corresponding class with the same name that is not lightweight. For example, there are classes named BaseDataPacket and BaseLwDataPacket. Lightweight classes are exactly the same as their non-lightweight counterparts except they are not subclasses of QObject. Lightweight classes are used when large amounts of data need to be loaded into memory. The absence of the QObject components reduces memory requirements and processing time. The advantage of having data classes be subclasses of QObject is that it simplifies the transfer of multiple data types between the same software objects.

3.3 Definitions

The following terms are used when discussing the FDMD software.

- FDMD Instance - A computer running either the client or server version of the FDMD software.
- FDMD Network - The collection of computers and sensors which make up the FDMD system.
- Virtual Flight Deck Real-Time (VFD-RT or VFD) - A ship motion simulation program, which is able to provide simulated sensor data to the FDMD over a TCP connection.

3.4 Communications Module

The communications module manages all of the data input and output for the FDMD. The FDMD can receive data from serial and network sources, and can output data over a

network to other FDMD instances.

3.4.1 Qt TCP Classes

The Qt networking module consists of two main Transmission Control Protocol (TCP) classes, QTcpServer and QTcpSocket. QTcpServer is used by creating an instance of it, and then configuring it to listen for a connection. The QTcpSocket class is created to connect to a listening TCP server. When a QTcpSocket connects to a QTcpServer, the QTcpServer creates a QTcpSocket pointer, and then uses that pointer to send information to the remote QTcpSocket. Whenever the QTcpSocket receives data, it emits a signal that can be connected to a function that processes whatever data have arrived.

In summary, a QTcpServer listens for a connection, and when it receives one, it can then transmit data through it. The QTcpSocket class is created on the receiving end, and receives data after it has connected to a TCP server. Further details on the implementation of these classes is available in Qt's documentation [24].

3.4.2 Data Sources

The software foundation of the communications module consists of the operation of independent "data sources". The data source classes are BaseDataSource, CcDataSource, CcNetworkDataSource, and CcCOMDataSource. The relationship between them is shown in Figure 3.5. The fundamental design of the FDMD network is that it can consist of any number of FDMD instances and data sources, but each FDMD instance must have knowledge of all data sources and FDMD instances in the network. The purpose of the data source classes is to give a software representation to each of those hardware components, computers and sensors.

All data sources are either instances of CcNetworkDataSource or CcCOMDataSource. The data management module is given access to data sources at the BaseDataSource

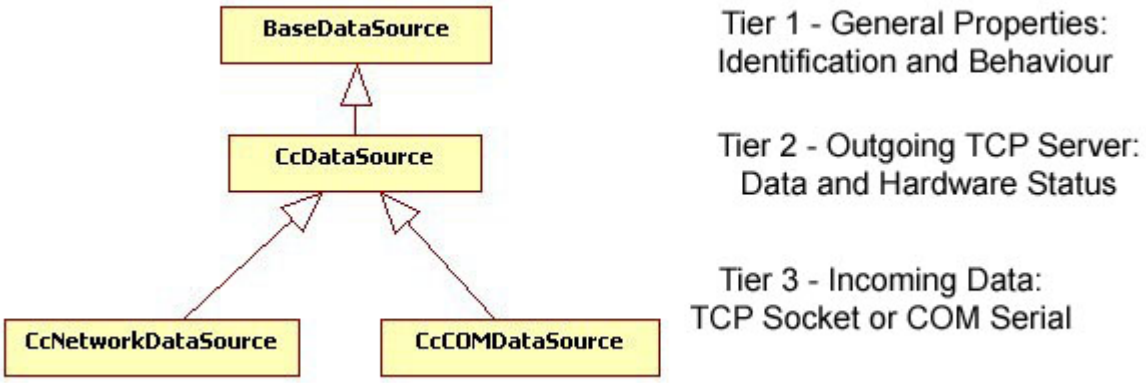

Figure 3.5: Data sources inheritance relationship.

level and user interface elements are given access to data sources at the CcDataSource level. Thus information and functionality are divided among the classes based on what information the data management and user interface classes require. Functionality is also divided among the classes based on shared functionality between CcNetworkDataSource and CcCOMDataSource. In general, the following trend is followed:

- BaseDataSource - This class consists of all of the general information that the data source needs to manage data, and that the FDMD needs to know, in order to handle the data.

- CcDataSource - This class contains all of the information and functionality needed to host a data or event server. Other FDMD programs can connect to a data server to receive data from whichever input that data source is also connected to.

- CcNetworkDataSource - This class contains all of the information and functionality needed to connect to an FDMD data or event server over a TCP connection. It can also connect to a VFD.

- CcCOMDataSource - This class contains all of the information and functionality

needed to connect to a serial data source, and is specifically tailored to communicating with the CrossBow AHRS400MB-200 inertial sensor.

Each data source operates independently, and only interacts with the communications module when transmitting data to the data management module. A data source has one data input and at most one data output connection. Since each data source contains code from both tiers 2 and 3, each data source is capable of connecting to a local serial or network input data source, and of hosting a data server that other data sources can connect to. If a connection is made to a data source's data server, then data are passed to the data server whenever it is received.

The FDMD prototype system contains four data sources: a primary data source, an FDMD client, an FDMD server, and a backup data source. Thus the FDMD contains four instances of the data source class, and has a configuration file for each data source. The difference between the FDMD client and the FDMD server are slight differences in each of those configuration files. Each of the FDMD client and server know about the existence of each of the components in the FDMD network. Figure 3.6 shows the data sources for the FDMD server and client and how they are connected to each other.

As shown in the Figure, each FDMD instance contains four data source objects corresponding to the two sensors and two computers. There are two types of communication links: (1) data communication for transfer of motion data between FDMD instances and (2) device status communication between FDMD instances. The first type of communication takes place between sensor data sources, and the latter takes place between client and server FDMD data sources. In the figure, the straight lines are data communication, and the diagonal lines are device status communication.

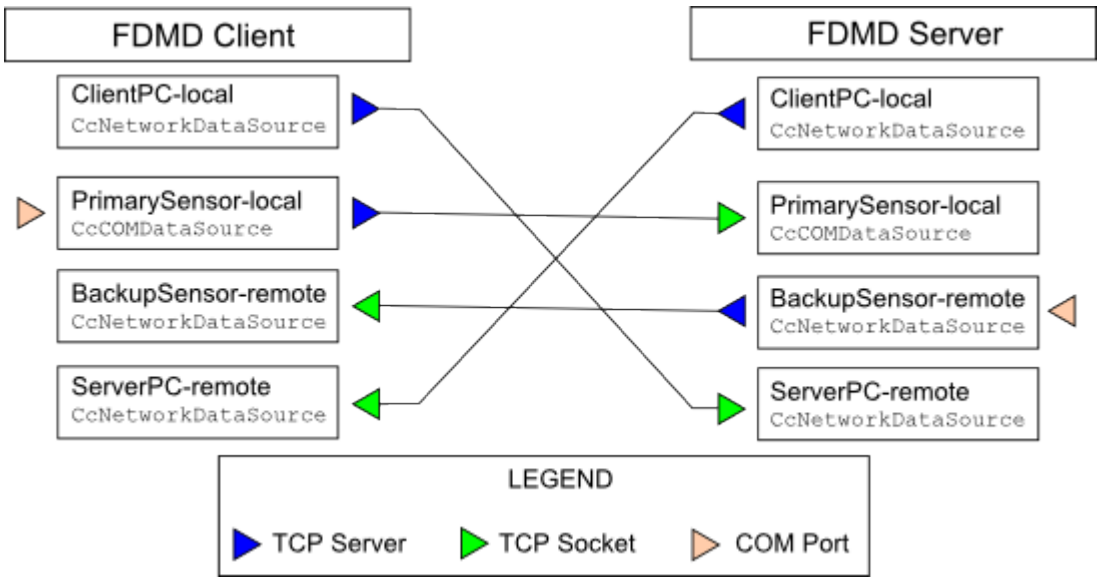

Figure 3.6: FDMD network communication paths.

3.4.3 Signals and Slots

Qt signals and slots are used to transfer information between software components in the FDMD. Each data source has a transmitData(QObject*) signal which is connected to the signal CcNetworkDataSource::transmitData(QObject*), which is connected to the slot DmDataManager::receiveData(QObject*). The data manager object then continues the processing of the data.

Each data source has functions CcDataSource::forwardDataPacket(QObject*) and CcDataSource::forwardDeviceStatusPacket(BaseSystemStatus) that are used to transfer data externally, and are called whenever data are received. These communication paths are discussed in more detail in Section 3.8.4.

3.4.4 Serial Data Processing

The Qt framework does not contain any functionality for communicating over a serial connection. An open source class that provides a simple interface for using the win32 serial

classes was selected for setting up the serial connection. All of the connection monitoring and data processing procedures were custom designed and built for use with the FDMD.

The CcCOMDataSource class creates an independent processing thread for receiving and processing data from a serial port. The thread is a subclass of QThread, and is named CcSerialDataMonitor. Serial communication is the only part of the FDMD that uses multithreading, and the reason for this is because it allows the serial processes to block while waiting for new data.

Signals and slots are used for communication between the serial thread and the data source. The signal CcSerialDataMonitor::transmitData(QObject*) is connected to the slot CcCOMDataSource::processData(QObject*), which also calls CcDataSource:: forwardDataPacket(QObject*) if the data need to be transfered over the network. The slot then emits the signal CcCOMDataSource:: transmitData(QObject*) (which is connected to the signal of the same name in the communications module as discussed above).

The creation and basic operating loop of a CcSerialDataMonitor is shown in Figure 3.7. The function QThread::run() is called in step 5, and is the function that needs to be rewritten when creating a thread based on QThread. The function CcSerialDataMonitor::run() loops continuously as long as the program is operating, and spends all of its time either in step 6, connectToSerial(), when not connected to a data source, or in step 8, monitorSerial(), when connected to a data source. These two functions will now be described in more detail.

A diagram outlining the operation of the function CcSerialDataMonitor:: connectToSerial() is shown in Figure 3.8. At this point, two things should be noted. The first is that the base class being used in the FDMD to control serial connections is called CSerial, which is free software and is available under the terms of the GNU Lesser General Public License. The second point to note is that the CcSerialDataMonitor class has been designed to detect any sort of fault in its connection, and immediately enter a loop for reconnection

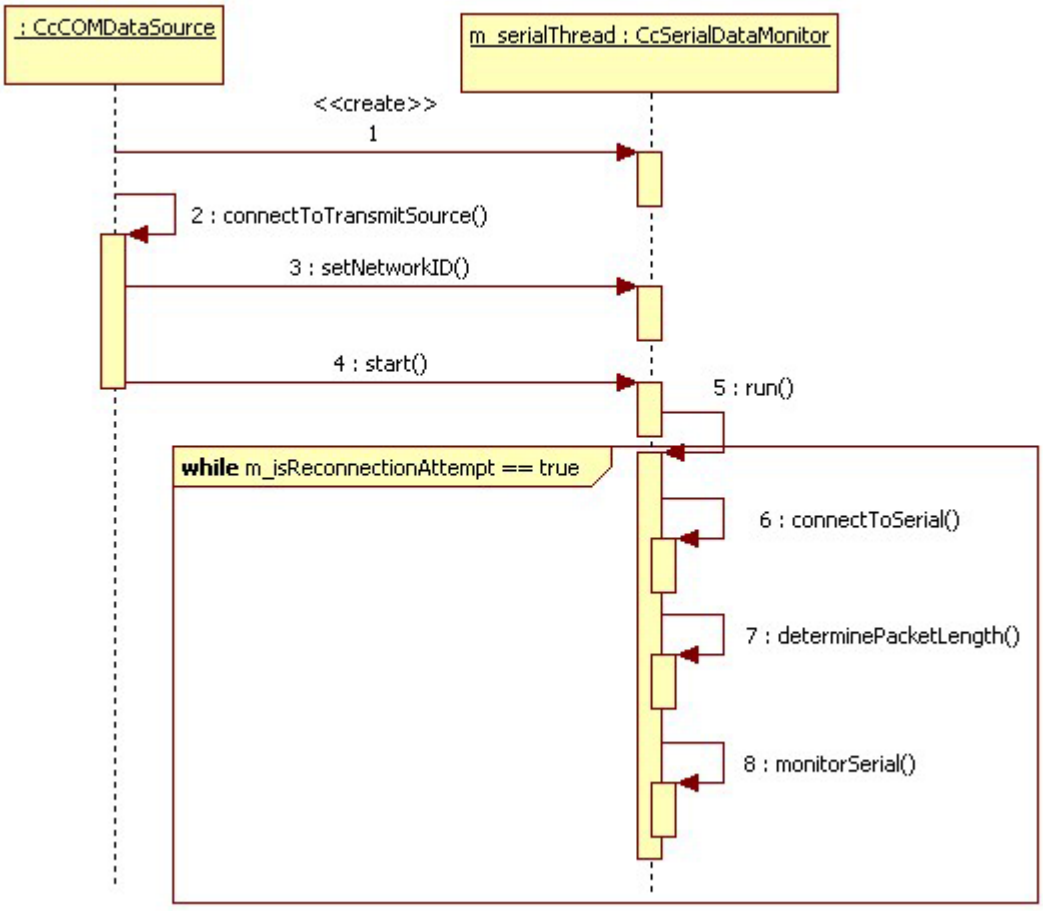

Figure 3.7: Creation and basic operation of a CcSerialDataMonitor thread.

upon fault detection. In Figure 3.8, steps 11, 13, 15, and 17 are used to configure the serial connection. After the connection has been established, the function CcSerialDataMonitor::pingSerial() is called in step 19 in order to ensure that the connection is operating correctly. If it is, the program continues to step 8 in Figure 3.7. If not, the connection function begins again. Whenever a connection needs to be (re)established, the property m_isReconnectionAttempt is set to *true*.

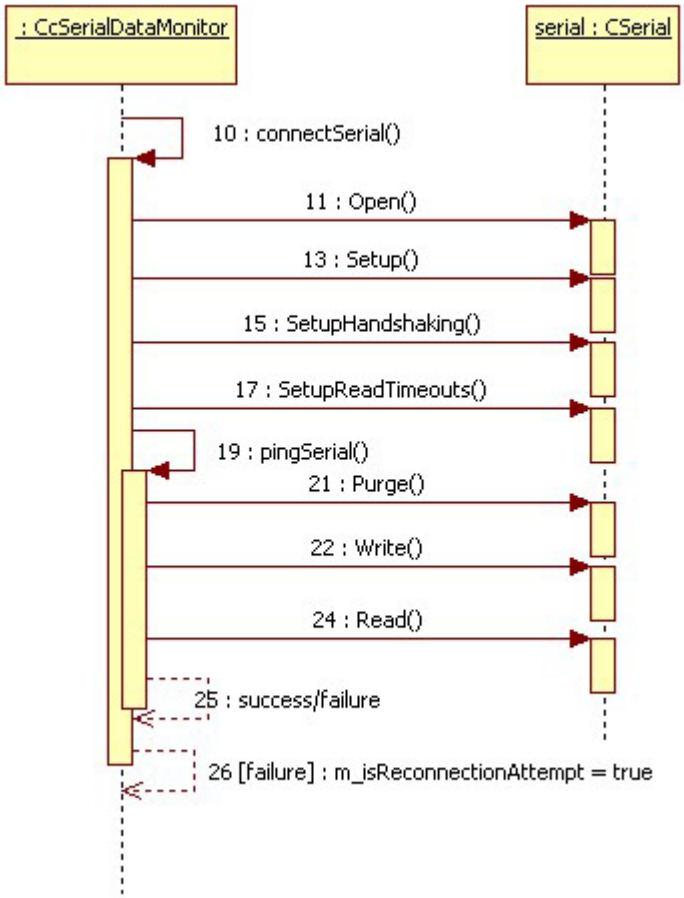

Figure 3.8: Operation of CcSerialDataMonitor when establishing a connection to a serial data source.

Figure 3.9 shows the function CcSerialDataMonitor::monitorSerial(), which is the main data receiving/processing loop. As long as the variable isMonitoring is equal to *true*, the

function continuously checks a buffer to see if a data packet has arrived, processes any data packets waiting in the buffer, and then transmits those data to the data source object the thread is connected to. If a communication fault is detected, isMonitoring will be set to $false$, causing the program to leave the data reading/processing loop, and m_isReconnectionAttempt is set to *true*, to cause the thread to begin the process of re-connecting to the data source.

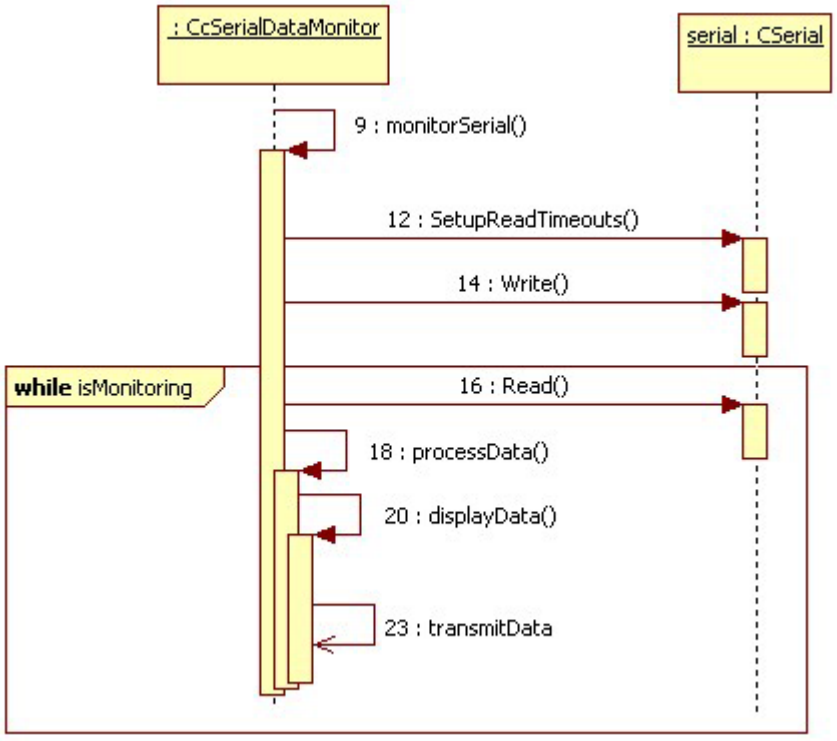

Figure 3.9: Main data receiving and processing loop of CcSerialDataMonitor.

AHRS400MB-200 Sensor Communication

Communication with the AHRS400MB-200 inertial sensor is performed by sending single character messages to the sensor and then waiting for a response. The exact steps that are followed to establish a data communication with the sensor are listed below.

1. The serial port connection is opened using the CSerial class.
2. The character 'P' is sent to the sensor. This places it in polled data mode. No response from the sensor is expected.
3. The entire serial buffer is read.
4. The serial buffer is purged using CSerial::Purge().
5. The character 'R' is sent to the sensor. This pings the sensor.
6. If the sensor replies with anything but 'H', the serial connection is closed and the process goes back to step 1.
7. The character 'a' is sent to the sensor. This puts it in angle mode. The sensor replies with 'A'.
8. The serial buffer is purged using CSerial::Purge().
9. The character 'G' is sent to the sensor three times in a row. This requests three data packets.
10. If three complete data packets are not received, the serial connection is closed and the process goes back to step 1.
11. The character 'C' is sent to the sensor. This places it in continuous data mode.
12. Data processing begins.

Data packets in angle mode are 30 bytes in length. The first byte is always 0xFF. The last byte is always a checksum. The checksum is calculated using the following process:

1. Sum all packet contents except the header and checksum.

2. Divide the sum by 256.

3. The remainder should equal the checksum.

When the sensor is running in continuous mode, the header is often not the only byte containing 0xFF. Therefore it is important to verify the checksum of every data packet. In the FDMD prototype, if synchronization with the data is lost, the data received are discarded while the software resynchronizes. In the final FDMD, these data should either be set aside and processed when synchronization is regained, or simply saved to disk and flagged with an event marker.

3.4.5 Communications User Interface

A screenshot of the communications user interface is shown in Figure 3.10.

The top half of the screen shows a graphic representation of each of the hardware components in the FDMD prototype system. The status of each component is indicated by its colour: green, orange, or red. This user interface element is designed to be as clear and concise as possible. Further details on the hardware of the system are shown in the bottom half of the screen. The window on the left side displays text updates on network communications. The window on the right displays detailed status messages from each of the hardware components in the FDMD network. Originally the status text for each component was displayed in the top half of the screen, but the logistics of always making the text fit in the space available were impractical to implement. Also, the window in the bottom right of the screen directly relates to the order in which data sources are stored, unlike the element at the top which is designed to be aesthetic. The software buttons displayed along the left side of the screen perform the following functions:

- CONNECT - Causes each data source to enter its connection loop. In the final FDMD system, this function will occur automatically when the system starts up.

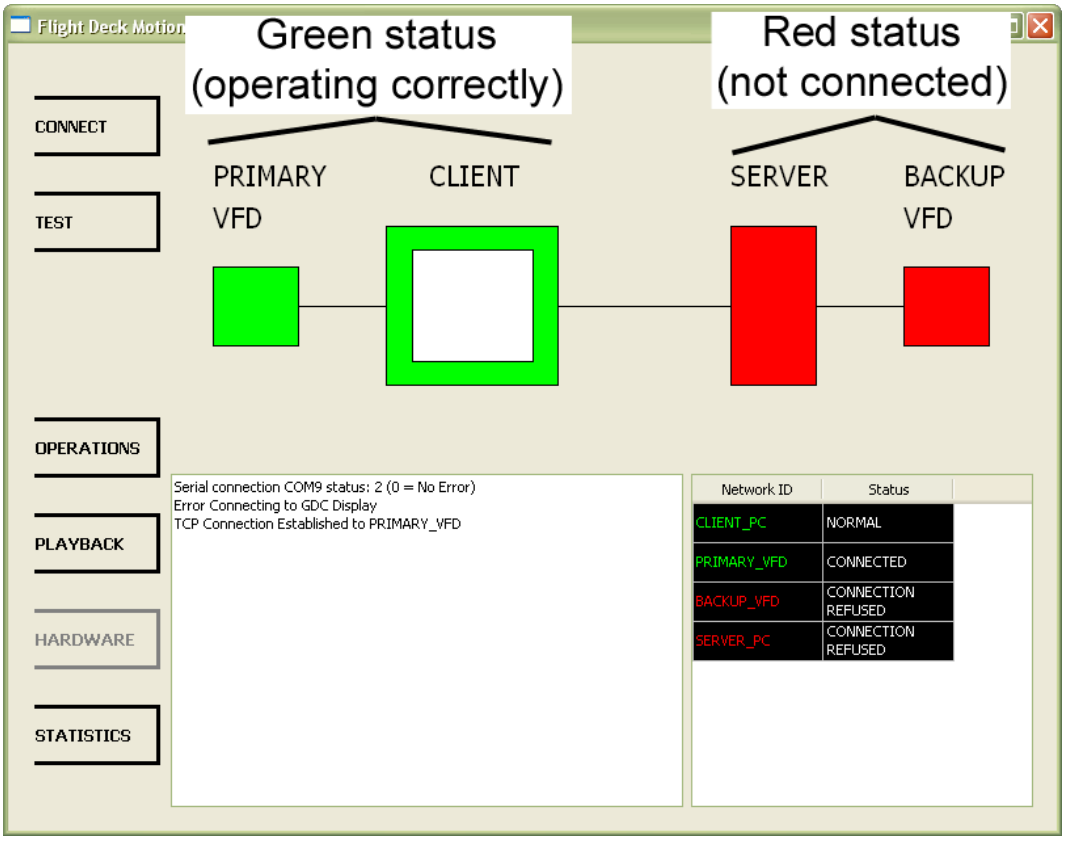

Figure 3.10: Screenshot of the communication user interface.

- TEST - Goes to the testing and configuration mode.
- OPERATIONS - Goes to the operations mode.
- PLAYBACK - Goes to the data management mode.
- STATISTICS - Goes to the statistics mode.

3.5 Data Management Module

The data management module is responsible for any action involving manipulation of raw motion data or interaction with hard disk data files. When the FDMD is monitoring ship motion in real-time, and specifically when connected to the Crossbow inertial sensor, the data management module carries out the following operations:

1. Identifies that a data packet is the object that has arrived;
2. Records the data packet to disk;
3. Multiplies the raw data by conversion factors to obtain values in engineering units;
4. Subtracts the gravity vector from linear accelerations;
5. Transforms the motion data to the coordinate frame of the ship;
6. Transforms the data from the sensor location to the location of the hauldown cable;
7. Transmits the data to the operations module;
8. Compares data from primary and backup sources when updates have been received from each; and
9. Deletes the data packet object when finished.

The sequence diagram of this process is shown in Figure 3.11.

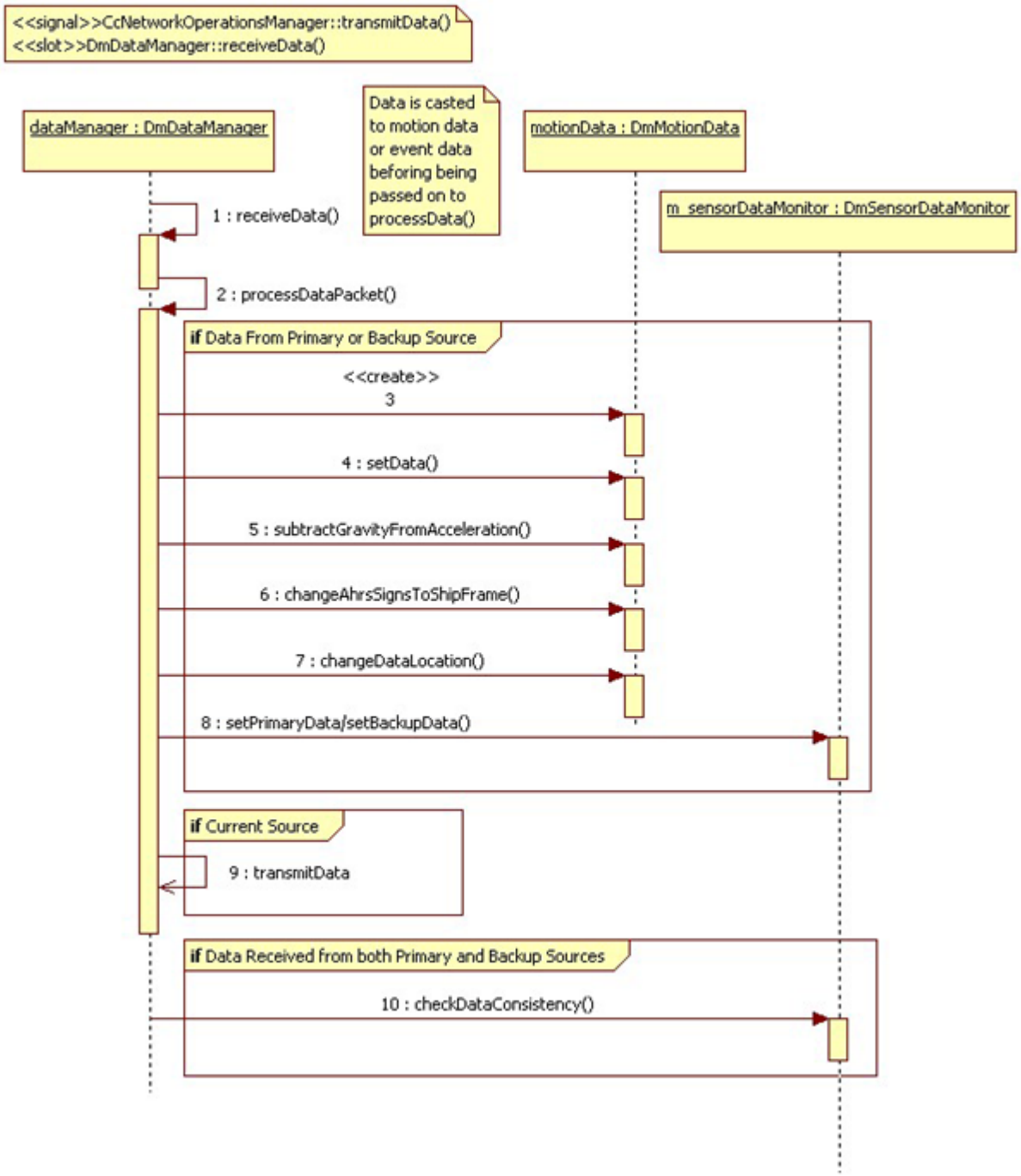

Figure 3.11: Sequence diagram of the main data processing function in the data management module.

3.5.1 Data Storage - DmMotionData

The DmMotionData class inherits functionality from the BaseMotionData class presented in Section 3.2.2, and adds attributes and functions that incorporate functionality required by the data management module. Specifically, the class adds storage for calculated values of angular accelerations, enhanced functions for setting engineering data values from data packets, and functions for post-processing the data once they have been converted to engineering units.

The DmMotionData class also adds the functionality to convert data from raw data packets to its own storage format of data in engineering units. It does this with the aid of another class, DmDataPacketDecoder, that exists to convert data from raw to engineering units, and is discussed in detail in the following section.

3.5.2 Conversion of Data to Engineering Values

The three components needed in the data conversion process are (1) a text file with conversions defined, (2) storage classes for storing the contents of the conversions text file, and (3) a class that performs the conversions defined in the text file.

The DmDataPacketDecoder class contains all of the storage classes for a set of conversions and the functionality for performing those conversions on one raw data value at a time. A set of conversions exists for the data received from a single data source. The class diagrams of DmDataPacketDecoder, DmDataConversion, DmCofficientList, and the relationships between them are shown in Figure 3.12. Each DmDataPacketDecoder contains a list of DmDataConversion objects, each corresponding to a single motion parameter. Each DmDataConversion contains a DmCoefficientList, which contains all of the numerical values required for a single engineering conversion.

When observing the relationship between DmDataConversion and DmCoefficientList

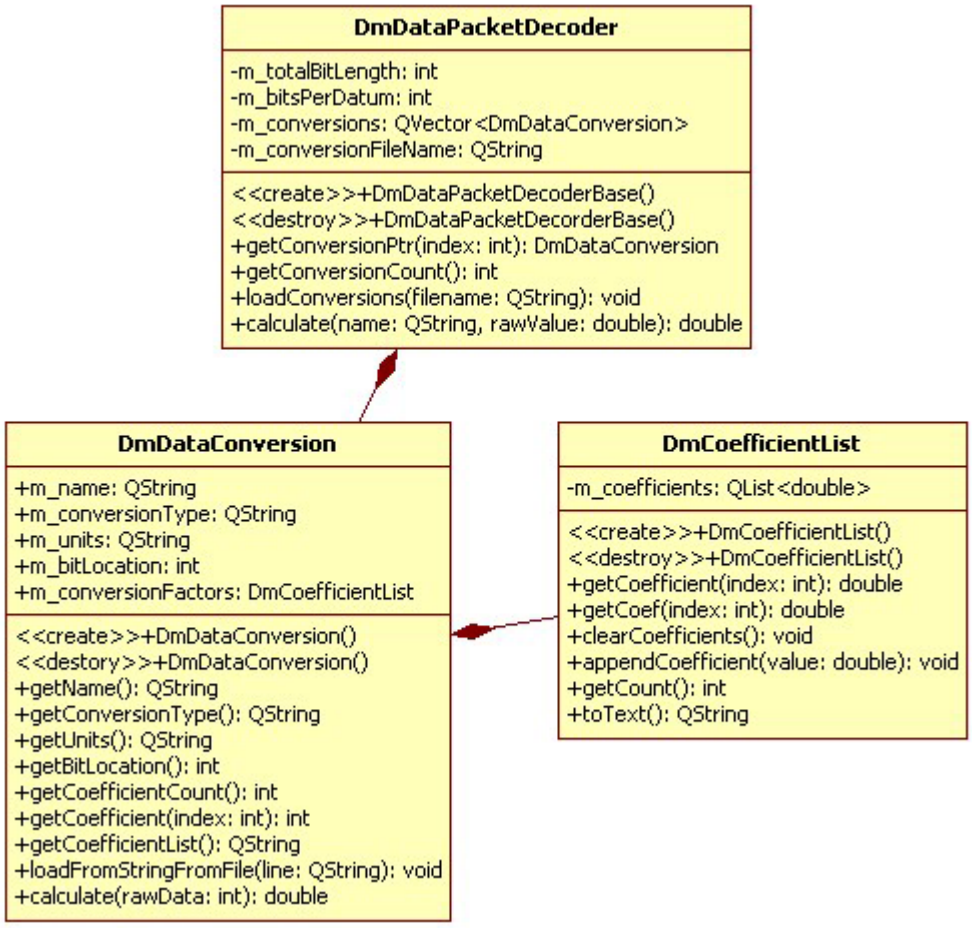

Figure 3.12: DmDataPacketDecoder and related storage objects class diagrams.

in Figure 3.12, one might wonder why these are not combined into a single class. The main reason for this is that in the original design of the classes it was not known how complex the data conversions could be. Having the coefficient list as a separate class simplifies the process of having to expand the DmDataConversion class in case more complicated conversion types need to be implemented in the future. For example, if a data conversion was not continuous, and different coefficients were required for different raw data ranges, then the current class structure would be easier to adapt to that type of conversion.

To use the data decoder class the first step is to create a DmDataPacketDecoder object and call DmDataPacketDecoder::loadConversions(QString filename). Once loaded, conversions are accessed by calling DmDataPacketDecoder::calculate(QString name, double rawValue) where *name* is the name of the motion parameter and $rawValue$ is the raw data value. The return value from the function is the value in engineering units.

As DmDataPacketDecoder::calculate(QString,double) is the only function used to convert data to engineering units, the data decoder class has no knowledge of the format of the classes that are used to store the data. Thus the function that uses a data decoder must be located elsewhere. As discussed in Section 3.5.1, the DmMotionData class contains this functionality.

The Crossbow inertial sensor and VFD transfer raw data as 16-bit unsigned integers in 2's complement format. The 2's complement format is a common method for dealing with the problem of storing negative numbers in unsigned integers. It uses the first half of the integer storage for positive values, and the second half of the integer storage for negative values. Therefore, in this case, if the unsigned raw data is larger than 0xFFFF / 2, then the value is negative. In order to calculate the negative value, 0xFFFF is subtracted from the unsigned raw data.

Engineering Conversions File Format

This section describes the file format used to store engineering conversions.

The locations of conversion files are specified in the configuration files for each data source. In the FDMD prototype, the conversion files used are named conv_xbow.txt and conv_vfd.txt. The first line of the conversion file is the number of conversions in the file, and the remaining lines are the conversions, one on each line. A screenshot of the top portion of conv_xbow.txt is displayed in Figure 3.13.

```
11
% Columns are separated by one or more tabs
% NAME      UNITS       2BYTE CONVERSION  COEFFICIENTS
xAccel      g           7     XBOW        4,1.5,0
yAccel      g           8     XBOW        4,1.5,0
```

Figure 3.13: Screenshot of a portion of the conversions file for the Crossbow inertial sensor.

Each conversion line consists of five pieces of information, each separated by one or more tabs. The five pieces of information are as follows.

1. Name - each motion parameter has a unique string identifier. Valid entries are listed in Table 3.1.

2. Units - a string containing the units of the parameter, for display purposes only.

3. Location - raw data are stored as an array of bytes. Each data value is stored as a 16-bit integer (2 bytes). Basically, the location entry is the location of the data if the array of bytes was treated as a 16-bit unsigned integer array. In the code, the Qt class quint16 is used.

4. Conversion - this is the conversion type. Valid entries are NONE, XBOW, XBOWTEMP, and NORMAL.

5. Coefficients - this is a comma-separated list of the numerical coefficients corresponding to the conversion type, in the order of a, b, c, d, etc.

The formulas for each conversion type are as follows:

- XBOW $= raw * (a * b)/(2^{15} - 1) + c$
- XBOWTEMP $= a * (b/c * raw - d)$
- NORMAL $= a + b * raw$
- NONE $= raw$

3.5.3 Data Management User Interface

The user interface for the data management module consists of two screens, one for displaying playback data and one for selecting files for playback. The first screen is shown in Figure 3.14.

At the top centre of the screen the date of the playback file is displayed. Below that is the nearest event marker, where the time and type of the marker are displayed. The centre of the screen contains the three graphs of roll angle, pitch angle, and vertical acceleration. The line in the centre of each graph is a cursor that marks the current time in the data. The nearest event displayed is the one nearest to this time, and the data value displayed on the left side of the graph in blue is the data value recorded at that time. Below the graphs is a quiescence history bar, and below that is a bar that gives information on time. The start and end times are the start and end of the data file, and the time in the middle is the current time. The black bar is representative of the amount of data currently displayed compared to the total time in the data file. At the bottom of the screen are status indicators for the primary and backup data sources, and along the sides

Figure 3.14: Data management user interface for displaying playback data.

of the screen are software buttons for operating the FDMD. The function of each button is listed below.

- + (Zoom in) - Halves the visible time range, centred around the current time.
- - (Zoom out) - Doubles the visible time range, centred around the current time.
- FILE LIST - Switches the user interface to the file list screen.
- OPERATIONS - Goes to the operations mode.
- HARDWARE -Goes to the communications mode.
- STATISTICS - Goes to the statistics mode.
- << (Back) - Moves the cursor back in time. Can be held down for continuous browsing.
- >> (Forward) - Moves the cursor forward in time. Can be held down for continuous browsing.
- REPLAY - When playback data are loaded, it simulates real-time FDMD operation by transmitting the data to the operations mode.

The second user interface screen of the data management module is displayed in Figure 3.15.

The title at the top of the screen, "DATA PLAYBACK - FILES", clearly identifies the purpose of this screen. The list of files is displayed in the middle. Each file is listed by its date, recording start time, and file size in kilobytes. The file to be loaded is highlighted. The "Source" radio buttons at the side indicate from which data source the current file list was recorded. The "Phase" and "Time" radio buttons indicate which quiescence settings will be used when loading playback data. Note that quiescence data are not recorded to

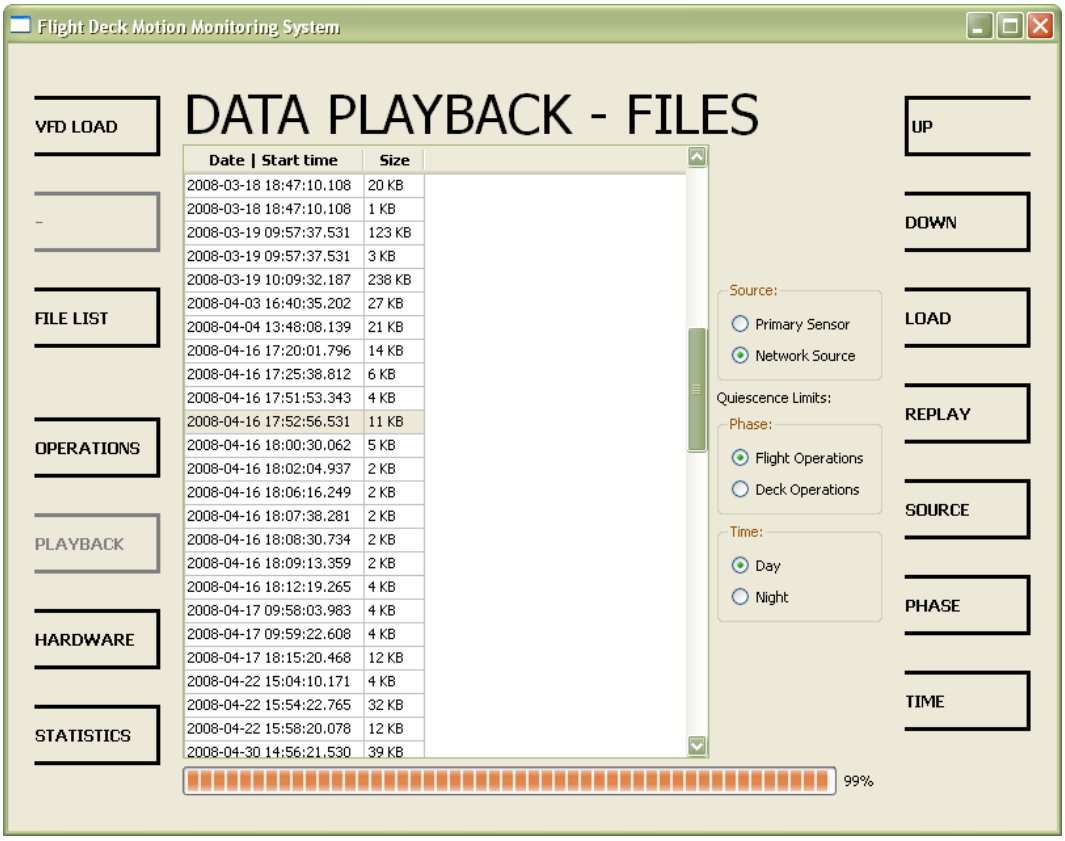

Figure 3.15: Data management user interface for selecting a playback file.

disk, and need to be recalculated when files are loaded. The progress bar at the bottom updates when data are being loaded. The function of each software button is listed below.

- VFD LOAD - Allows a user to directly load a VFD data file into the FDMD.
- FILE LIST - Switches the user interface back to the graphs view.
- UP - Moves the file selection cursor up one file.
- DOWN - Moves the file selection cursor down one file.
- LOAD - Loads the selected file into the FDMD.
- SOURCE - Changes the file list between the primary and backup data source.
- PHASE - Changes the phase quiescence settings that will be used when the data are loaded.
- TIME - Changes the time of day quiescence settings that will be used when the data are loaded.

3.6 Operations Module

The purpose of the operations mode is to display real-time motion data. It is responsible for loading all of the quiescence limits settings, applying them to incoming data, and for processing any events generated by the user. OpOperationsManager is the main operations module class. The current implementation of the operations user interface displays roll, pitch, and vertical acceleration data.

3.6.1 Class Building Blocks

The OpParameter class is the fundamental class of the operations mode. One instance of the class exists for each motion parameter being monitored. Each OpParameter object contains all of the identification and display information for the parameter, all of its quiescence limits thresholds, an array of the most recently received data for that parameter, and the functionality needed to calculate quiescent status updates whenever new data are added. All of the parameters being monitored in the operations module are managed by the OpShipDeck class, and there is one OpShipDeck instance in the OpOperationsManager class. A simple figure which shows how data are processed by the operations mode is shown in Figure 3.16. After step 4 in the figure, signals and slots are used to transfer data updates to the user interface. OpParameter objects are directly connected to corresponding user interface elements.

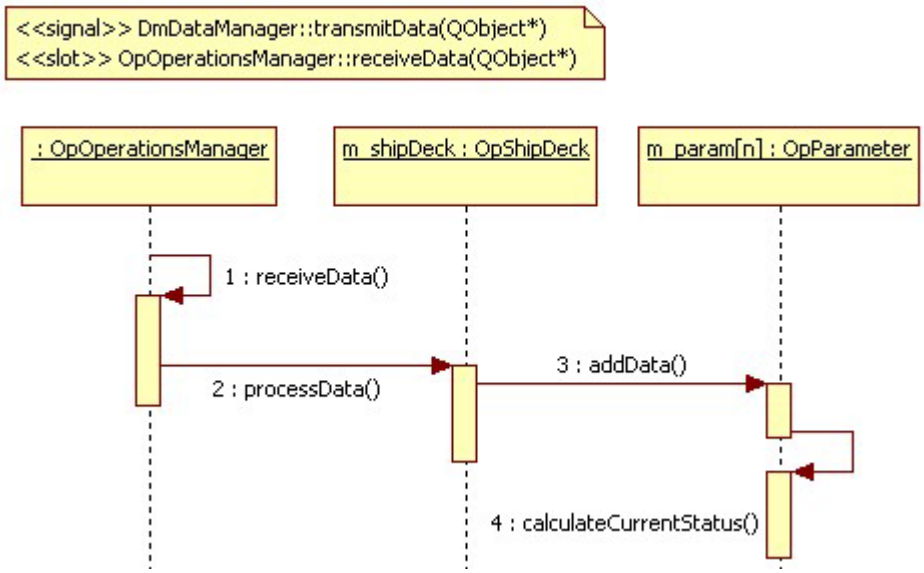

Figure 3.16: Sequence diagram of data processed by the operations module.

3.6.2 Quiescence Limits Thresholds

Each motion parameter has a set of quiescence limits thresholds corresponding to different phases, times of day, and whether the motion is entering or exiting quiescence. Currently there are eight different combinations, and there is a separate configuration file for each. The list of parameters being monitored is generated from the first of these files, qpi0.ini.

Quiescence Thresholds File Format

A screenshot of a quiescence thresholds configuration file is shown in Figure 3.17. Note that only representative (not actual) quiescence limits have been used in this document.

```
DAY-FLIGHT-EXIT
rollAngle "ROLL" DEG 7 4 3 -3 -4 -7 1
pitchAngle "PITCH" DEG 5 4 3 -3 -4 -5 1
zAccel "VA" G 0.80 0.60 0.40 -0.40 -0.60 -0.80 0
```

Figure 3.17: Screenshot of a quiescent period thresholds file.

The first line defines under which conditions the quiescent limits should be used. In this case the limits should be used when time=DAY, phase=FLIGHT, and direction=EXIT. Each of the following lines contains three strings followed by seven numerical values, all separated by spaces. The first string is the name identifier for the motion parameter. Each motion parameter has a unique name, in this case being rollAngle, pitchAngle, and zAccel. The full list of possible parameter names can be found in Table 3.1. The next string is the text to be displayed in user interface elements. The third string is the units to be displayed in user interface elements. The first six numerical values are the quiescent limits for that parameter. The first and last numbers are the maximum and minimum values to be shown on graphs and gauges. The second and fifth numerical values are the 'red limits' and the third and fourth values are the 'yellow limits'. When a motion parameter exceeds its yellow limits, its quiescence status becomes yellow. When its red limit is exceeded, its

quiescence status becomes red. The final numerical value on each line is the number of data peaks that must be experienced before a quiescence status change can take place from a higher to a lower quiescent state. For roll and pitch this value is one, and for vertical acceleration this value is zero.

3.6.3 Quiescence Status Monitoring

Whenever new data arrives, each parameter updates its quiescence status. Quiescence status changes can take place whenever data passes over a limit or when data experiences a peak. A simple slope calculation method is used to detect changes in slope direction. In order to avoid the detection of false peaks due to noise in the data, a moving average calculation of ten data points is used. It is not the most accurate method of calculating slope, but this is a situation where performance is more important that accuracy. The function OpParameter::calculateCurrentStatus() contains the quiescence status updating algorithm, which carries out the following steps:

1. Calculate the current slope of the data by using the following steps:
   (a) Add together the last 10 data points in one sum, and the 10 data points before the current data point in another sum.
   (b) Perform the same sums for the times of the data points.
   (c) Find the average data values and average time steps for the last 10 data points, and the 10 data points before the current point using the sums calculated in the previous two steps.
   (d) Use the two average data points to calculate the averaged slope of the last 10 data points.
2. Determine the current quiescent status based on the relationship between the data and the quiescence limits.

3. If the number of peaks needed to change quiescence is zero, change status to the current quiescence status and exit.

4. If the current status is greater than the last status, change to the current status and exit. (red > yellow > green)

5. If the new calculated slope and the slope calculated during the last function iteration have different signs (ie. a peak) perform the following checks:

   (a) If the last quiescent status was red and the current status is:

      i. green, then change to green;

      ii. yellow, then change to yellow;

      iii. red, then change to red.

   (b) If the last quiescent status was yellow and the current status is:

      i. green, then change to green;

      ii. yellow, then change to yellow;

   (c) If the last quiescent status was green and the current status is green, then change to green.

6. Save the new slope calculation for the next data point arrival.

At first changing to a quiescent status that is not different from the last quiescent status appears redundant, but the more important use of OpParameter::changeStatus() is to record a quiescence data point, and in this case record one whenever a peak is detected in the data, regardless of whether it is a quiescence status change or not.

The function OpParameter::changeStatus(QPStatusKind newStatus, double avgValue, double avgTime) performs the following actions:

1. updates the current status to the new status;

2. records a quiescence data point using OpParameter::recordQpiDataPoint(); and

3. emits the signal OpParameter::statusChanged(QObject*).

The function OpParameter::recordQpiDataPoint(double avgValue, double avgTime) peforms the following actions:

1. creates a new BaseQpiDataPoint object;

2. derives a percent value for the data point by calling OpMotionParameterThresholds::calcFractionOfLimit2(double);

3. assigns the remaining properties of the quiescence data point; and

4. removes any stored quiescence data points that are beyond the maximum storage amount.

Quiescence Enter and Exit Limits

In the FDMD's configuration files, separate quiescence limits can be defined for entering and exiting quiescence. This means that when a parameter's quiescent status is red, the 'enter' limits are used because the parameter needs to enter quiescence, and when the quiescent state is green or yellow, the 'exit' limits are used because the parameter is going to exit quiescence. An image showing the enter and exit quiescence entries for roll angle is shown in Figure 3.18. Recall that the actual limits are not used in this document. Currently roll angle is the only parameter that uses different enter and exit limits. As shown in the figure, the limits needed to exit quiescence are $\pm$ 4, and the limits needed to enter quiescence after leaving it are $\pm$ 3. This limits configuration means that the parameter will spend more time in green or yellow quiescence during a quiescent period, and that after it has exceeded its limits, it must hit a lower peak value in order to re-enter quiescence, thus increasing the probability that it will stay in green quiescence.

```
DAY-FLIGHT-ENTER
rollAngle "ROLL" DEG 7 3 3 -3 -3 -7 1
DAY-FLIGHT-EXIT
rollAngle "ROLL" DEG 7 4 3 -3 -4 -7 1
```

Figure 3.18: Enter and exit quiescence entries for roll from qpi0.ini and qpi1.ini.

3.6.4 Overall Quiescence Monitoring

The operations module needs to calculate the overall quiescent status of all parameters for the quiescent period indicator and quiescence history user interface elements. This is done using an OpQpiParameter, which is a subclass of OpParameter. The class is used by registering all of the currently monitored OpParameter objects with it. Whenever new data arrive and each of the regular parameters are updated, this class' update function is called as well. This function goes through each of the registered OpParameter objects and determines which quiescence status should be used as the overall status.

3.6.5 Operations User Interface

The operations user interface is one of the most important parts of the FDMD project. It has gone through many iterations and a screenshot of its current implementation is shown in Figure 3.19.

There are three types of user interface elements on this screen: data history graphs, instantaneous value indicators, and a quiescent period indicator. The graphs display a two-minute data history with quiescence limits marked, and an overall quiescence history below all of the graphs. The instantaneous indicators display a black line that moves across a coloured gauge, and a ship animation that reflects the current state of the ship. The angles shown by the ship animations are a fraction (one half and one third) of the actual ship motion. This was implemented in order to avoid situations where it appeared that the ship was sinking, as the scales on the graphs are not the same as reality (ie. to scale,

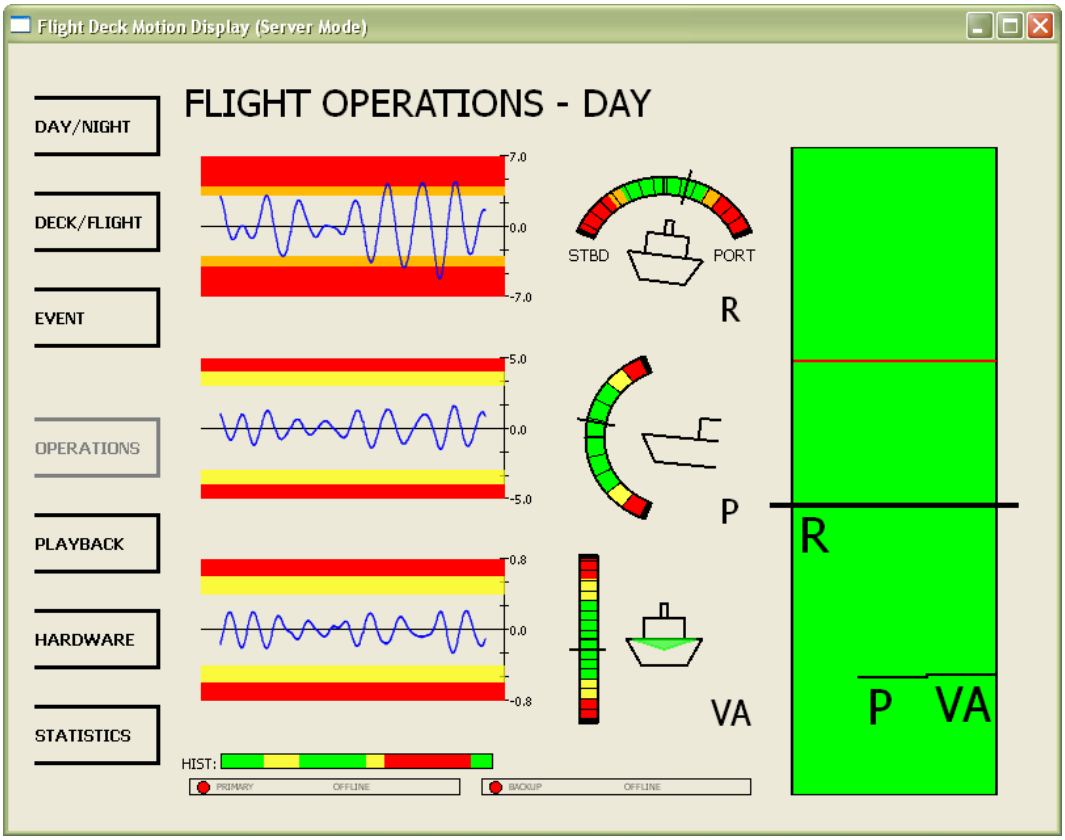

Figure 3.19: Screenshot of the operations user interface monitoring real-time motion data.

$\pm 5^{\circ}$ on the pitch indicator would be so small the value would be unreadable). The pitch and roll indicators are instances of a user interface element designed to display angles, and the vertical acceleration indicator is an element designed to display accelerations. The accelerations indicator operates the same as the pitch and roll indicators, but instead of the ship moving, a transparent arrow moves up and down, following the movement of the gauge marker. The colour of the arrow matches the quiescent status of the acceleration parameter. For this indicator ship motion was avoided in an attempt to convey that vertical acceleration is different from monitoring pitch angle, vertical position, or vertical velocity.

The quiescent period indicator (QPI) is the bar shown on the right side of the screen. The colour of the bar matches the current overall quiescence of the ship. Inside the QPI are bars marking the relative magnitudes of roll, pitch, and vertical acceleration, scaled by their maximum safe limit. The roll and pitch bars indicate the most recent peak values experienced, while the vertical acceleration bar marks its instantaneous value. The exception to this rule is that if roll or pitch exceed their limits, then immediately their height would be moved up into the red.

At the bottom of the operations user interface are two indicators for the current status of the primary and backup data sources. Along the left side of the screen are software buttons for operating the FDMD. The buttons are placed on the left side of the screen, as that would be the most accessible side for using them in an LSO compartment. The function of each button is listed below.

- DAY/NIGHT - Switches the quiescence limits between day and night values, and records a limits change event marker.
- DECK/PHASE - Switches the quiescence limits between flight and deck operations values, and records a limits change event marker.

- EVENT - Records an event marker.
- PLAYBACK - Goes to the data management mode.
- HARDWARE - Goes to the communications mode.
- STATISTICS - Goes to the statistics mode.

By using the DAY/NIGHT and DECK/PHASE buttons listed above, the displayed quiescence limits can be changed and all quiescence calculations from then on use the new limits. The data graphs are immediately updated with the new limits. Quiescence history data however is not recalculated.

3.7 User Interface Design

This section describes the custom user interface elements designed and implemented for the FDMD.

3.7.1 User Interface Elements

One of the most impressive features of the FDMD user interface is that aside from the software buttons, every user interface element is designed to scale with the screen size. This has been done by storing all internal dimension of each element as fractions of the width and height of the element.

In general, all of the user interface elements in the FDMD operate through similar mechanics. Every user interface element is a subclass of QWidget, and every class that has had its visual display customized has had its QWidget::paint() function rewritten. This paint function is called automatically whenever a screen redraw is required, and can be called manually by calling QWidget::update() by a function within the class itself. All

of the user interface elements that change over time have a function for receiving updates, such as receiveUpdate(QObject*). For user interface elements that display motion data, typically a data point is passed to the function and the element is redrawn. For user interface elements that monitor hardware status, calling the function receiveDeviceStatus() only causes the element to redraw itself, as these elements contain pointers to the data sources they are monitoring.

3.7.2 Communications

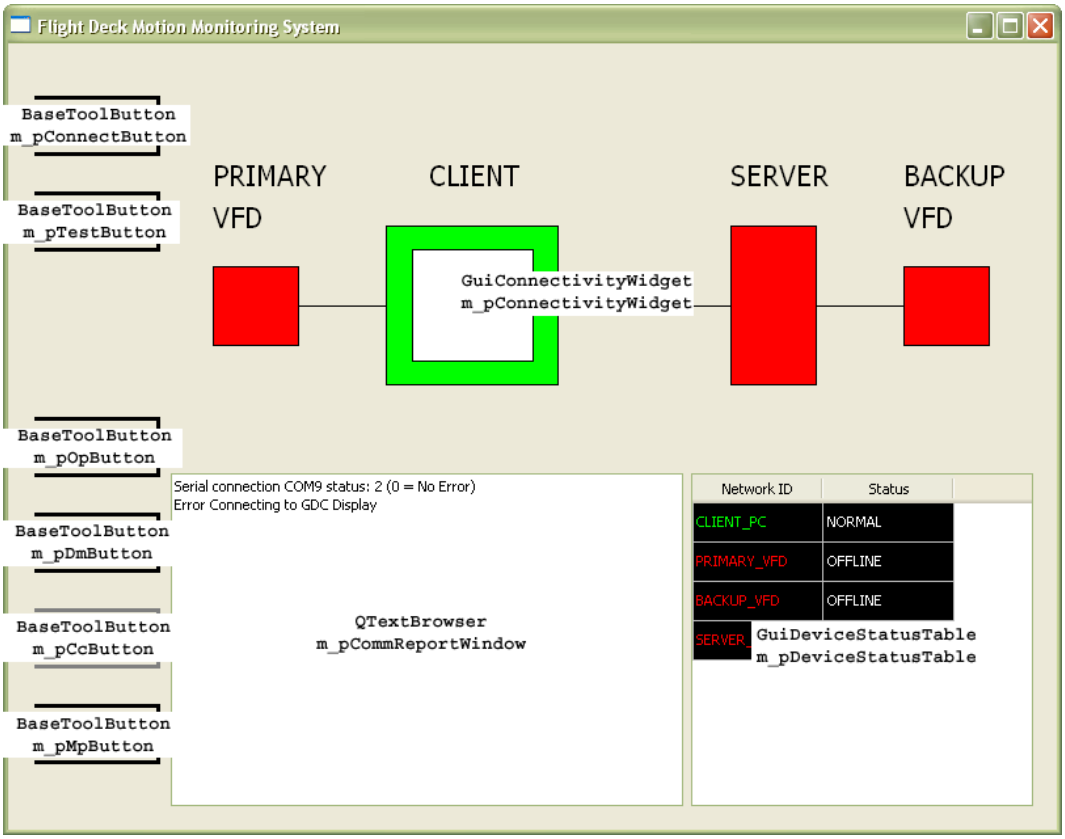

Figure 3.20: Diagram of the user interface classes that make up the communications user interface.

An image of the communications user interface overlaid with the names of its user interface classes is shown in Figure 3.20. The communications user interface only consists

of three classes aside from its software buttons. The GuiConnectivityWidget and GuiDeviceStatusTable elements operate in a similar fashion: each is initialized with a QList of CcDataSource objects, and whenever their receiveDeviceStatus(BaseSystemStatus) function is called they automatically update themselves.

All of the software buttons in the FDMD use the BaseToolButton class, which is a subclass of QToolButton. Its purpose is to provide an alternate user interface appearance to the standard appearance of buttons in Qt or Windows.

3.7.3 Data Management

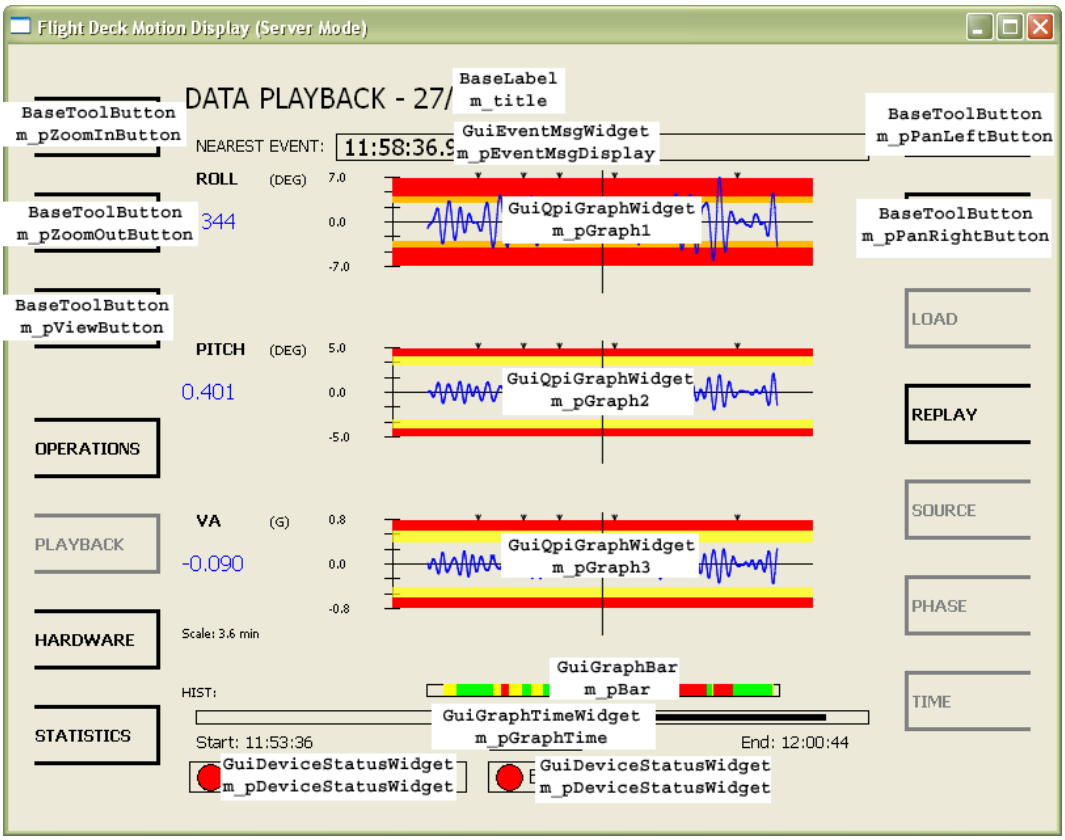

Figure 3.21: Diagram of the user interface classes that make up the data management user interface.

An image of the data display of the data management user interface overlaid with the

names of its user interface classes is shown in Figure 3.21. The data management and operations module's user interfaces each have a title at the top of the screen which is an instance of the BaseLabel object. BaseLabel is a subclass of QLabel, and being subclassed provides full control over how the title is drawn. (In this case used for dynamic resizing.) GuiEventMsgWidget is a simple element that displays the nearest event marker in the data. GuiGraphTimeWidget displays information on the time location of the displayed data, and GuiDeviceStatusWidget displays the current device status of the primary and backup sensors. GuiDeviceStatusWidget is initialized with a pointer to a data source, and updates its display as required.

Graphs

In the FDMD software, there is a single graph user interface element that can be configured for different roles. In data management the graphs are configured to have adjustable current times (marked by a cursor) and time ranges, to display name and units labels, to have their Y-axis on the left side of the graph, and to display the current numerical value corresponding to the location of the cursor. The graphs also have the option to display individual quiescent period histories though most of the time it is disabled to reduce screen clutter.

The major difference between the graphs in the data management module and the operations module are that the graphs in the latter operate in real-time mode. This means that the current time or location of the cursor is always the most recent data point so that as data arrive the graph continuously moves to make space for them. The operations module graphs also display less information and have their numerical axes on the right side of the data instead of the left.

The GuiGraphBar class displays the overall quiescence history of all of the parameters, and is a subclass of GuiQpiGraphWidget with its graph hidden instead of the quiescence

history. This class is configured to receive updates from a OpQpiParameter instead of an OpParameter like the rest of the graphs. It is connected to its OpQpiParameter the same way as the other graph classes are connected to an OpParamter, as OpQpiParameter is a subclass of OpParameter.

All of the motion data related user interface elements need to display the same time range all the time. This is done using the GuiGraphManager class. GuiGraphManager stores pointers to all of the GuiGraphWidget, GuiGraphTimeWidget, and GuiEventMsgWidget classes, and coordinates their behaviour. It does this by providing the functions setCurrentTime(), panLeft(), panRight(), zoomIn(), and zoomOut(), which are the actions used to navigate data in the FDMD. The GuiGraphManager is directly connected to the user interface and related controls in the data management module, though it has no user interface representation itself.

3.7.4 Operations

An image of the real-time data display of the operations user interface overlaid with the names of its user interface classes is shown in Figure 3.22. Similar to the data playback UI, three graphs are shown along with a single quiescence history bar. The operations user interface has the addition of instantaneous value indicators and a quiescent period indicator.

The classes GuiShipPositionWidget and GuiShipAccelWidget use the same function names as GuiQpiGraphWidget and are connected to data updates the same way. Each of them has a setupFromObject(QObject*) function for initializing the element, and an addData(BaseDataPoint&) function for receiving data updates. Unlike the graph elements, only the most recent data point needs to be stored locally. The GuiShipAccelWidget has an additional slot addQpiData(BaseQpiDataPoint&) as it also requires quiescence data in order to set the colour of its arrow.

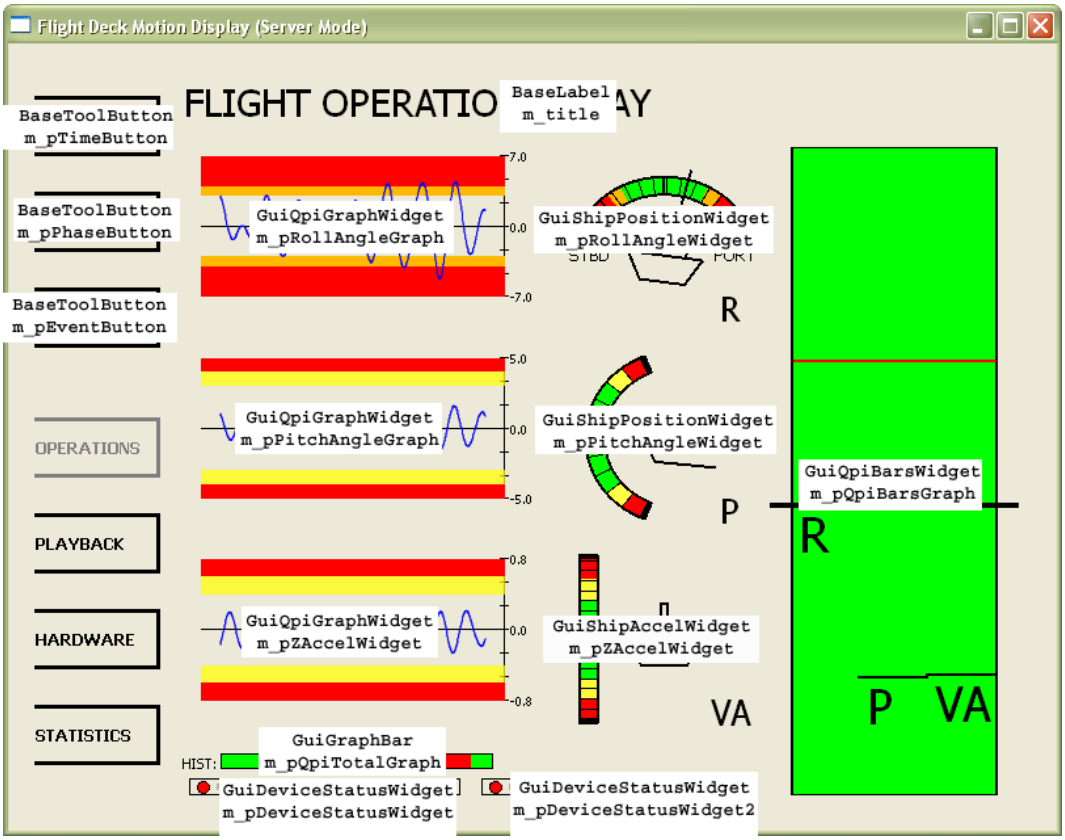

Figure 3.22: Diagram of the user interface classes that make up the operations user interface.

3.7.5 Quiescent Period Indicator

Images of the quiescent period indicator (QPI) at a green and red state are shown in Figure 3.23. The QPI operates by monitoring roll angle, pitch angle, and vertical acceleration. The DND draft specification [4] includes details on the monitoring of only roll angle and vertical acceleration, but pitch angle is included as well in order to aid operators in the transition between basing flight deck quiescence on roll and pitch, to roll and vertical acceleration.

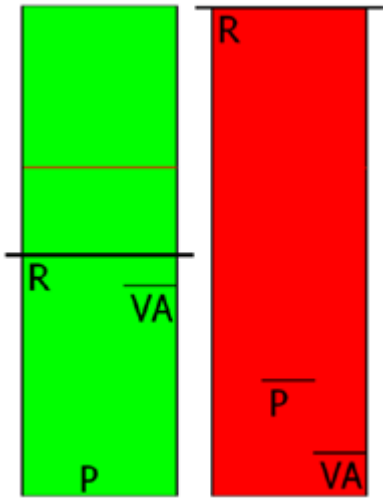

Figure 3.23: Quiescent period indicator green state (left) and red state (right).

The height of each individual bar is the value of the parameter divided by its limit. Therefore the transition from quiescent to non-quiescent is equal to 1 for each parameter. The top of the bar represents 1.3. If a parameter exceeds its limit by more than 130% then the bar and label remain at the top of the QPI.

The height of each of the bars only updates when their quiescence status changes. This results in discontinuous motion within the indicator but is actually convenient for the intended use of the QPI. If instantaneous values are used for the heights of each of the bars, there can be situations that arise where the peak motions of the parameters are very

large, but if they are simultaneously passing through zero an incorrect impression of the ship's state can be conveyed. Having the height of each bar represent the previous peak experienced by each parameter conveys useful information when an operator only has a few seconds to look at it.

The QPI can also be configured so that the height of only the vertical acceleration indicator changes with instantaneous values. This can be useful as the quiescence definition for accelerations does not require a subsequent peak below limits to enter quiescence. In this case, if an operator has a little bit of time to monitor the QPI, they can get an impression on how quickly the flight deck's vertical acceleration is changing.

3.8 Communication Between Modules

3.8.1 Data Types

There are three types of information that are transfered between FDMD software modules:

1. motion data;
2. event data; and
3. device status data.

Motion data travel in the form of BaseDataPacket or DmMotionData. A datum is always received at the communications module, placed into a BaseDataPacket, transmitted to the data management module which transforms it into a DmMotionData, and then transmitted to the operations and statistics modules.

Event data travel in the form of BaseEventMarker or BaseLimitsChangeEvent. A datum is always generated in the operations module, placed into one of its container classes, and transferred to the communications module. The communications module

transfers the data to other FDMD instances over a network if required, and then passes the data on to the data management module, where it is recorded to disk.

A device status datum can originate from any module and its storage class is BaseSystemStatus. In the FDMD prototype, device status data are not recorded to disk, and are only used to update device status user interface elements. For the most part, device status updates are generated by data sources, although in testing of the FDMD, device status messages are generated in the data management and operations modules for monitoring internal data processing rates.

3.8.2 FDMD Initialization

As FDMD modules are generally not aware of each other, there is a single overriding class which provides the top-level FDMD user interface, creates instances of each of the FDMD modules, and connects them so data can be passed between them. This class is called BaseFdmdUi. BaseFdmdUi is a subclass of QMainWindow, is the highest level graphical user interface class of the FDMD, and is created in main.cpp (which is the highest-level code file).

BaseFdmdUi uses a QStackedWidget to display all of the FDMD operating modes in the same screen area, and contains overall Qt slots for changing to each of these operating modes. The class constructor creates all of the operating modules, connects their signals and slots, and also creates the thread that monitors input from the AMLCD display. That thread is directly connected to BaseFdmdUi, which distributes key press events to the currently displayed operating mode.

3.8.3 FDMD Module Class Design

Each of the FDMD module classes share the same base functionality and are structured similarly. A class hierarchy diagram of the module/user interface classes is shown in

Figure 3.24.

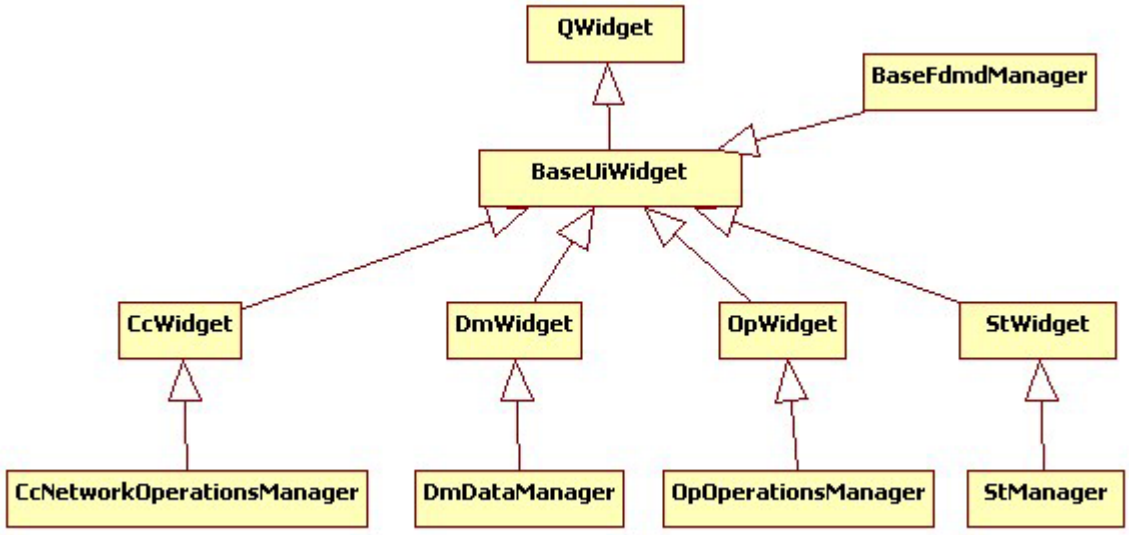

Figure 3.24: Class hierarchy diagram of the main module classes in the FDMD.

Each module inherits its base user interface functionality from QWidget. The first layer, BaseUiWidget, contains all of the functionality that is common for all modules. This means that it contains all of the signals and slots functions for communicating the FDMD's data types between modules. A snapshot of the code that defines these signals and slots is shown in Figure 3.25.

```
public slots:
    virtual void receiveData(QObject*);
    virtual void receiveKeyPress(int keycode);
    virtual void receiveDeviceStatus(BaseSystemStatus);

signals:
    void transmitData(QObject*);
    void forwardData(QObject*);
    void transmitDeviceStatus(BaseSystemStatus);
    void forwardDeviceStatus(BaseSystemStatus);
```

Figure 3.25: Shared signals and slots between FDMD modules.

The functions receiveData() and receiveDeviceStatus() are for receiving data, event, and device status messages. The function receiveKeyPress() is for receiving key press events from BaseFdmdUi. There are two types of communications signals. The 'transmit' signals are for transferring messages between internal FDMD components. The 'forward' signals are for transferring messages to external FDMD instances. This convention is used throughout the FDMD's communication components. Details on how these signals and slots are connected are discussed in the following section.

The next layer of the module class hierarchy is the one that includes CcWidget, DmWidget, OpWidget, and StWidget. BaseFdmdManager is a special case and will be discussed separately. In this layer, the user interfaces of each operating mode are defined. While Qt supports visual user interface creation in order to facilitate user interface construction, in this case it was decided that it would be better to manually code all of the FDMD's user interfaces due to the large number of custom classes (which is supported in Qt's designer, but not implemented well).

The next layer is the one that includes CcNetworkOperationsManager, DmDataManager, OpOperationsManager, and StManager. This layer is where the functionality of each operating mode is implemented, where data storage and processing classes are defined, and where they are connected to the user interface as needed. This is where the receiveData() and receiveDeviceStatus() functions are defined as well.

In general, each of the FDMD modules and their related classes are independent and unknown to the other modules. The minor exception to this is the data management's data playback functionality which is discussed in Section 3.8.5. The major exception to this rule is the implementation of BaseFdmdManager which is the test and configuration module class. In the FDMD prototype, this class does not divide its user interface and functionality into separate classes, and also has knowledge of all other classes. This allows BaseFdmdManager to provide full configuration services for all modules, and monitoring

services for all modules. In the FDMD prototype its only functionality is to change the FDMD's monitoring between primary and backup data sources, and to provide internal data rate monitoring. In the final version of the FDMD, when the module will be set up to provide numerous user interfaces for configuring the FDMD, it will likely be expanded into multiple class layers for ease of development and management.

3.8.4 Module Signals and Slots

Qt signals and slots are used to communicate data, event, and device status messages between modules. A diagram which illustrates data and events communication paths is shown in Figure 3.26. Included for reference and continuity is a generic data source object.

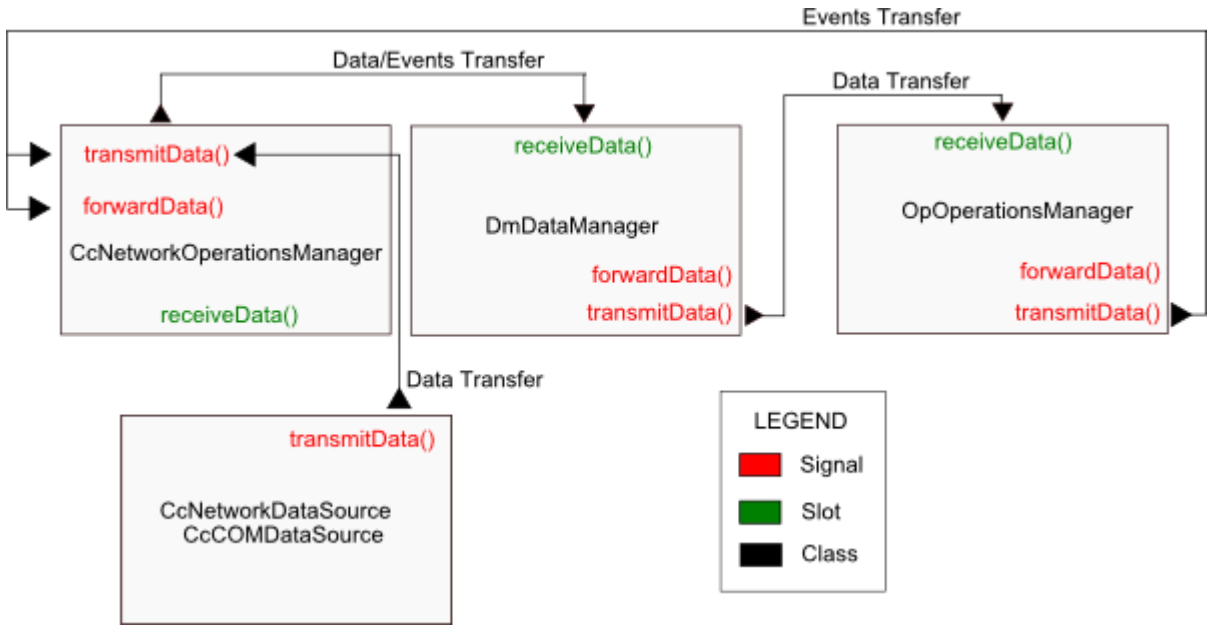

Figure 3.26: Signals and slots connections between FDMD modules for communication of data and events.

Figure 3.26 illustrates how data and events flow in one direction, from data sources to the data manager and operations, and how events flow in the other direction, from operations to communications and data management. A diagram that shows the FDMD

device status communication paths is shown in Figure 3.27. This figure illustrates how device status messages are transmitted from data sources to communications, and then to other program modules. Also shown are the signals and slots used to communicate internal data rates from data management and operations to the test/configuration class BaseFdmdManager.

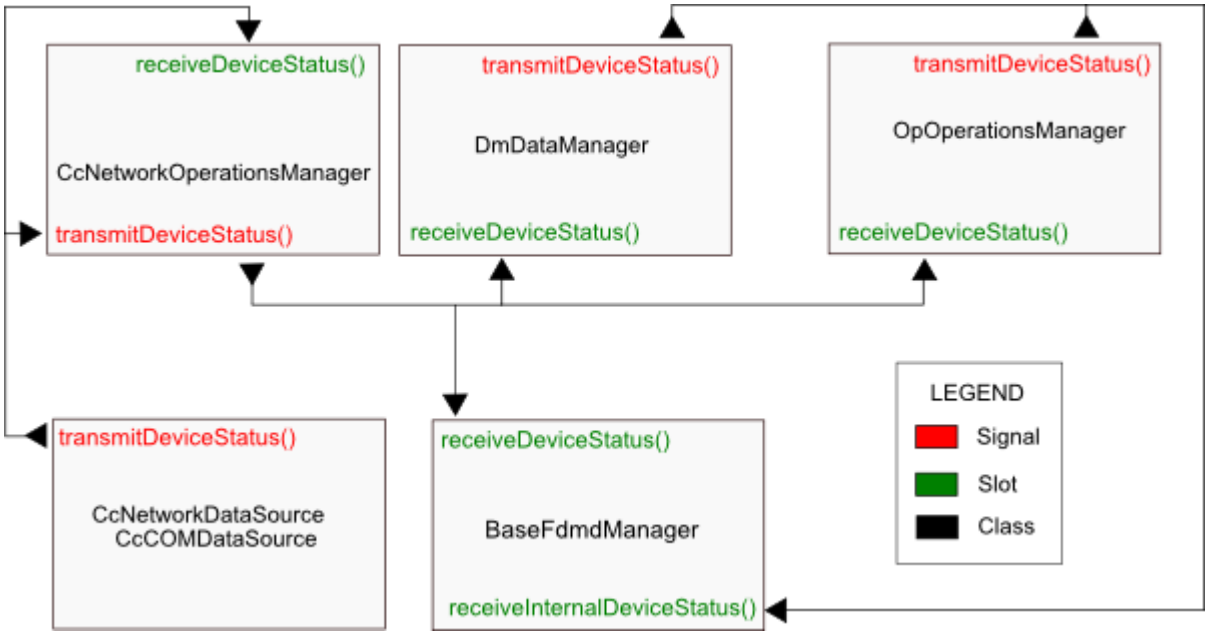

Figure 3.27: Signals and slots connections between FDMD modules for communication of device status messages.

3.8.5 Data Playback

The FDMD data management module is capable of loading and displaying recorded data with full quiescence information. This combines the functionality of the data management and operations modules in such a way that violates their existence as independent modules, although in this case data management has knowledge of the operations classes, but not vice versa. In the original design of the FDMD, the data playback functionality did not include quiescence information, and thus this conflict did not exist. In practice however,

the data playback functionality is not useful without the quiescence data included.

The method used for generating the quiescence information for parameters and then plotting the recorded data is very similar to the real-time calculation and plotting method that the operations module uses. After a set of playback data has been loaded, it can also be timed and passed on to the operations module to replay the data set as it would have appeared on that screen originally. This data playback functionality between modules was also not included in the original design of the FDMD, but works well because of the design of the FDMD nonetheless. This real-time data playback feature has also proved to be extremely useful and valuable in development, testing, and demonstration of the FDMD. The function DmDataManager::loadPlaybackFile(QString filepath) is used to load data into the data management module and the steps it uses are as follows.

1. If a playback file is already loaded, delete it from memory.
2. Load the file specified by $filepath$ into memory.
3. Read the date from the data file and update the screen title.
4. If a valid pointer to the operations module is not available, exit.
5. Create new OpParameter objects and name them according to the parameters to be graphed.
6. Create a new OpQpiParameter and register the new OpParameter objects with it.
7. Connect the OpParameter and OpQpiParameter objects to data management's graph user interface elements.
8. Clear all data drawn in data management's graph user interface elements.
9. Get the thresholds information for the OpParameter objects from the operations module, and set up the data management graphs with that information.

10. Get a pointer to the data decoder for the playback file.

11. For each raw data point in the playback file perform the following actions:

    (a) Update the data management progress bar.

    (b) Create a BaseLwDataPacket and use the data decoder to convert the data to engineering units.

    (c) Transform the data to the hauldown cable location.

    (d) Add the data to each of the OpParameter objects. This will automatically:

        i. Update the quiescent status of the parameter.

        ii. Transmit the data to the corresponding graph.

    (e) Update the total parameter. This will update the quiescence history bar automatically.

12. For each event data point, add the data to each of the graphs.

13. Zoom out to the minimum/maximum times on each graph.

14. Disconnect the OpParameter and OpQpiParameter objects from the graphs.

15. Delete the OpParameter and OpQpiParameter objects.

Real-time data playback can only be done after a data file has been loaded and plotted in the data management mode. It is done by creating an instance of a DmDataPlaybackTimer object, providing the data file and a data decoder to the object, and by connecting the playback timer object's signal transmitData(QObject*) to the communication module's slot, receiveData(QObject*). The Qt class QTimer is used to keep track of elapsed time.

3.9 Additional Modules

The operation of the two additional FDMD modules is summarized in this section. The test and configuration mode was one of the base operating modes specified in the design requirements of the FDMD. The statistics mode is intended to convey that the FDMD can potentially be used for flight planning, although its potential is limited without wind speed and direction data.

3.9.1 Statistics

The statistics mode of the FDMD displays real-time statistics on flight deck motion. A screenshot of the current user interface is shown in Figure 3.28.

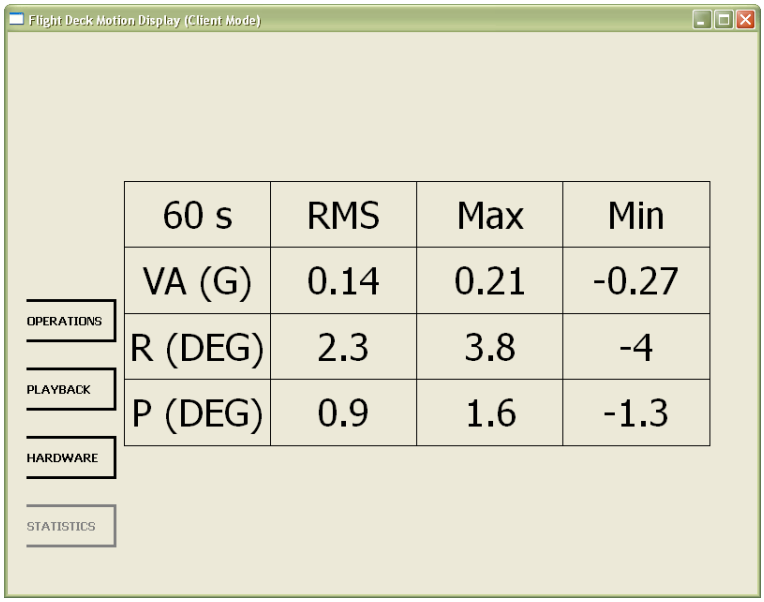

Figure 3.28: Screenshot of the statistics user interface.

The current implementation of the statistics user interface only displays root mean square (RMS), minimum and maximum values of roll angle, pitch angle, and vertical acceleration. Information on lengths of quiescent periods could also be included. The

"60 s" in the top left corner of the table indicates what time range of data is being used to generate the statistics. In this case, the last sixty seconds of data have been used. Originally this module was meant to provide flight planning functionality, but this turned out to be difficult without additional environmental information such as wind speed, wind direction, and ship heading.

3.9.2 Test and Configuration

The current implementation of the test and configuration module provides a minimum amount of functionality for testing and configuration. A screenshot of the user interface is shown in Figure 3.29.

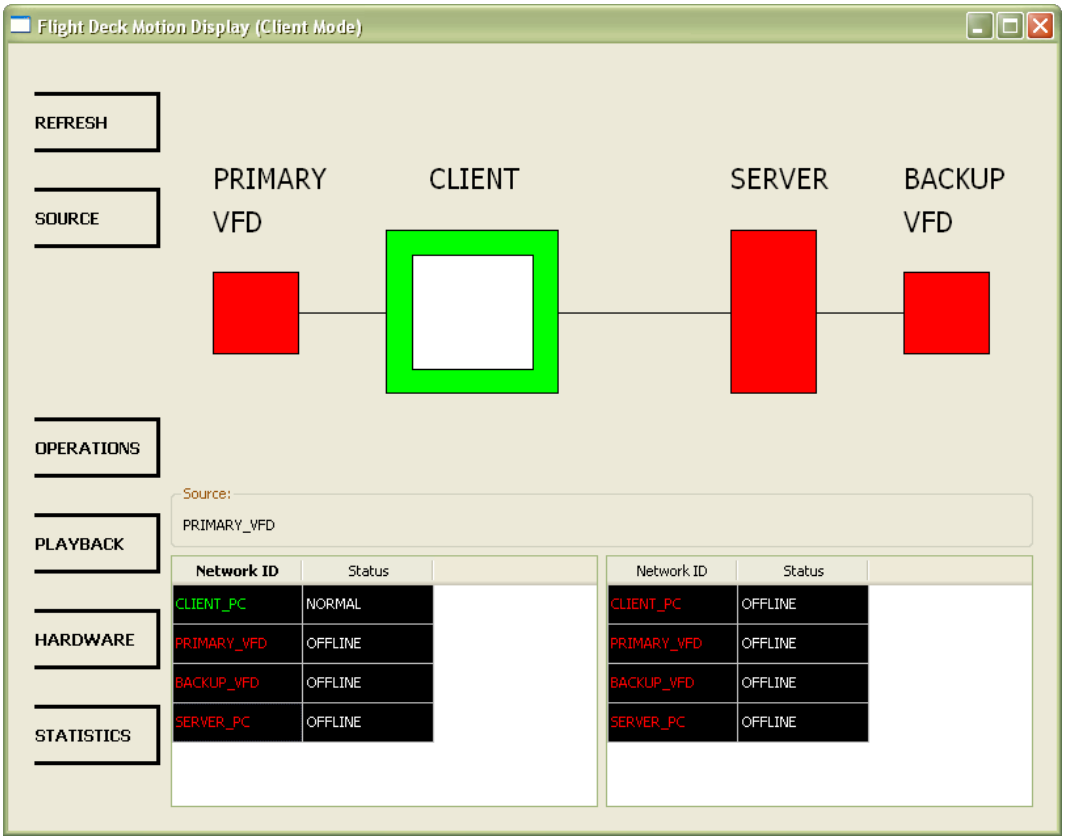

Figure 3.29: Screenshot of the test and configuration user interface.

Currently this operating mode appears to be very similar to the communications operating mode, and this is true as its current role is very similar. The purpose of this operating mode is to provide more detailed information on the hardware configuration and status than the communications operating mode. In the final FDMD system, this operating mode could provide even more detailed information than it currently does, and the communications mode could provide less.

There are two uses for this operating mode that are not available in the communications mode. The first is the additional status window. The status window on the bottom left displays the same information as the window on the bottom right in the communications mode. The window on the bottom right displays information on the internal status of the FDMD. The data processing rates in the data management and operations modules are calculated and transmitted to this module. These communication paths are illustrated in Figure 3.27.

The second, more important use for the test and configuration operating module is the ability to change the data source that is monitored in the operations module. The 'SOURCE' button on the left side of the screen toggles the monitored data source between primary and backup. The currently-monitored source is displayed in the centre of the screen. The 'REFRESH' button updates this display and it is present because the test and configuration user interface is set up before the monitored data source is assigned in the FDMD's initialization process.

Chapter 4

Testing and Evaluation

This chapter describes the testing and evaluation that took place to validate and refine the FDMD's software. The first part of this chapter discusses the various tests that were undertaken in order to ensure proper operation of the FDMD, and the test environments that were used in those tests. The second portion of this chapter details a user interface evaluation that took place with the co-operation of Sea King pilots from 12 Wing, Shearwater, which gave invaluable insight into the role that the FDMD can play in helicopter operations.

4.1 FDMD Systems Test Plan

The FDMD systems test plan was created to act as a guide and a checklist for verifying the operation of the FDMD. Three types of tests were undertaken: (1) unit tests, (2) system tests, and (3) lab integration tests. A summary of the tests undertaken is presented in the following sections. More details on the results can be found in the FDMD Systems Test Report [25].

4.2 Unit Testing

Unit testing took place during the development of the FDMD as each software component was completed. In general these tests verify that all possible inputs to each 'unit' of the software result in intended or satisfactory outputs. These tests took place for each individual software component and for combined components. A summary of the tests undertaken is presented here to convey the level of testing performed.

4.2.1 Communications

The communications module monitors the status of all hardware components in an FDMD network and manages data input and output of an FDMD instance. To ensure proper operation of the communications module it is necessary to verify that:

- error messages are displayed for each possible device status;
- hardware components automatically reconnect to each other if disconnected;
- data from incoming sources arrive with acceptable latency; and
- outgoing data arrive at their target with acceptable latency.

4.2.2 Data Management

The data management module is responsible for converting data to engineering units, saving data to disk, and loading saved data back into memory for analysis. In order to ensure proper operations of the data management module it is necessary to verify that:

- all user interface software buttons operate as expected;
- data and events are properly saved to disk, in proper locations and formats;

- data conversions are carried out properly; and
- data playback loads and converts the correct data, within the correct time limits.

4.2.3 Operations

The operations module is the only end-point for data transfered through the FDMD aside from being saved to disk or used in the minor operations of the statistics module. Once engineering data have been received and processed they are deleted here. In order to ensure proper operation of the operations module it is necessary to verify that:

- all eight possible quiescence limits settings are displayed and applied properly;
- instantaneous value indicators display their data properly, and without a noticeable delay;
- data history graphs are updated at the rate of data arrival and displayed accurately; and
- quiescence status updates behave as expected under all possible quiescence/data conditions.

4.3 Laboratory Integration and Testing

4.3.1 Test Environment

A parallel project being developed by the Applied Dynamics Research Group is the creation of a Real-Time Virtual Flight Deck (VFD-RT) simulation environment. It is designed to execute small scale real-time helicopter-ship simulations for a variety of engineering purposes.

The VFD-RT simulation environment is composed of three layers: a core application that manages the passage of time and coordinates communication between parts of the simulation, a set of data input applications called "providers" that contain the mathematical models of the simulated entities, and a set of simulation data output applications called "clients". Figure 4.1 shows the three layers and the data flow between layers. The entities in the top row are providers, the middle row is the VFD-RT core application, and the bottom row are the clients.

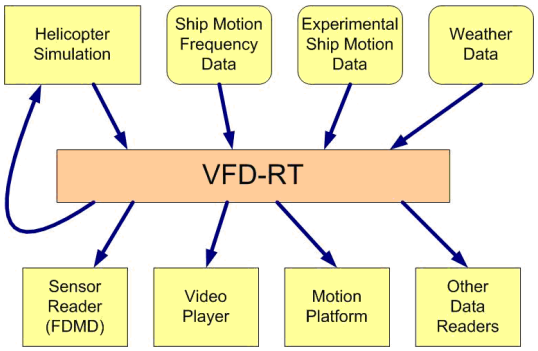

Figure 4.1: The three layers of the VFD-RT simulation environment.

4.3.2 Integration Testing

Much of the network functionality and operation can be tested by using the VFD-RT simulation environment. The VFD-RT can generate simulated motion sensor data at any location on the ship, and transfer that over a network connection to the FDMD. In addition, sensor location details are applied to the data at the data output client layer so that multiple TCP clients can be used to simulate data arriving from multiple sensors. A diagram of the FDMD lab integration system setup is shown in Figure 4.2. A single VFD-RT core application transmits data to two TCP clients, which are connected to the FDMD client and server instances. The FDMD client transmits its data to the FDMD server and the data from the two simulated sensors can be compared.

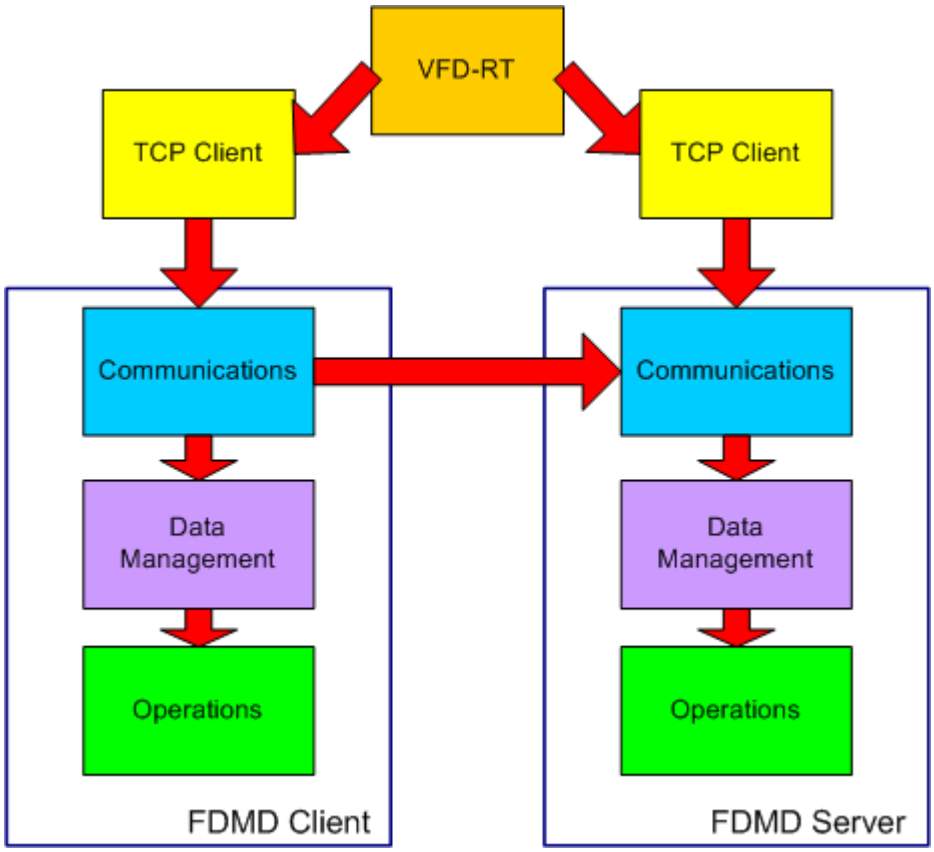

Figure 4.2: Lab integration setup for verify data processing abilities of the FDMD.

This laboratory setup was available from the start of the project and used to perform many of the verifications discussed in the preceding sections. This setup additionally allowed the following operations to be verified.

- Coordination of data recording of more than one data source, and the ability to switch the monitored source between them;
- Transfer of sensor data between FDMD instances;
- Transformation of sensor data to the hauldown location regardless of its measuring location; and
- Comparison of data from multiple data sources and detection of data measurement faults.

Two of the mathematical algorithms in the FDMD that needed to be verified were (1) numerical differentiation of the angular velocities in order to obtain angular accelerations and (2) transformation of data from any location on the ship to the hauldown point.

The numerical differentiation algorithm uses four-point backwards differencing to calculate the angular accelerations. The equation used is [26]:

$$\left(\frac{dy}{dt}\right)_n = \frac{-2y_{n-3} + 9y_{n-2} - 18y_{n-1} + 11y_n}{6h} \tag{4.1}$$

where y_n is the data point at index n, and h is the time step between data points. Once the angular accelerations have been calculated the equation used to transfer linear accelerations to the hauldown cable location is:

$$\mathbf{a_h} = \mathbf{a_p} + \mathbf{w} \times \mathbf{w} \times (\mathbf{r_h} - \mathbf{r_p}) + \boldsymbol{\alpha} \times (\mathbf{r_h} - \mathbf{r_p}) \tag{4.2}$$

where the definition of each variable is listed in Table 4.1.

Table 4.1: Definitions of acceleration equation variables.

Symbol	Definition
$\mathbf{a}_h$	linear acceleration vector at the hauldown cable
$\mathbf{a}_p$	linear acceleration vector at the sensor measurement location
$\mathbf{w}$	angular velocity vector of the ship
$\boldsymbol{\alpha}$	angular acceleration vector of the ship
$\mathbf{r}_h$	vector distance from the ship centre of gravity to the hauldown cable location in the ship's coordinate system
$\mathbf{r}_p$	vector distance from the ship centre of gravity to the sensor measurement location

Only linear accelerations need to be transformed as orientation measurements and their time derivatives are location independent in the ship's coordinate system.

The acceleration calculations were verified by configuring the VFD to transmit acceleration measurements to the FDMD, as they could then be compared to the calculated values. A graph of each of these values is shown in Figure 4.3.

The smooth line in Figure 4.3 is the measurement directly received from the VFD. The line with the largest variance is the result of the four-point backwards differentiation of the pitch angular velocity. The line in-between is the differentiated value averaged among four data points. The importance of this graph is that it shows that the differentiation algorithm is following the actual value and that as expected, there is significant error in the individual calculations. This error is due to the nature of numerical differentiation. Additionally, for display purposes at least, the data can be averaged to reduce the variance in the data.

Once the FDMD was able to calculate its own angular accelerations, the next algorithm to test was the transformation of data between ship locations, and then the combination of the two algorithms. Multiple motion parameters were tested, and the results for flight deck vertical acceleration are presented here. The assumed hauldown location was (-50,0,10) [metres] in the ship's coordinate system. The base vertical acceleration data set used is

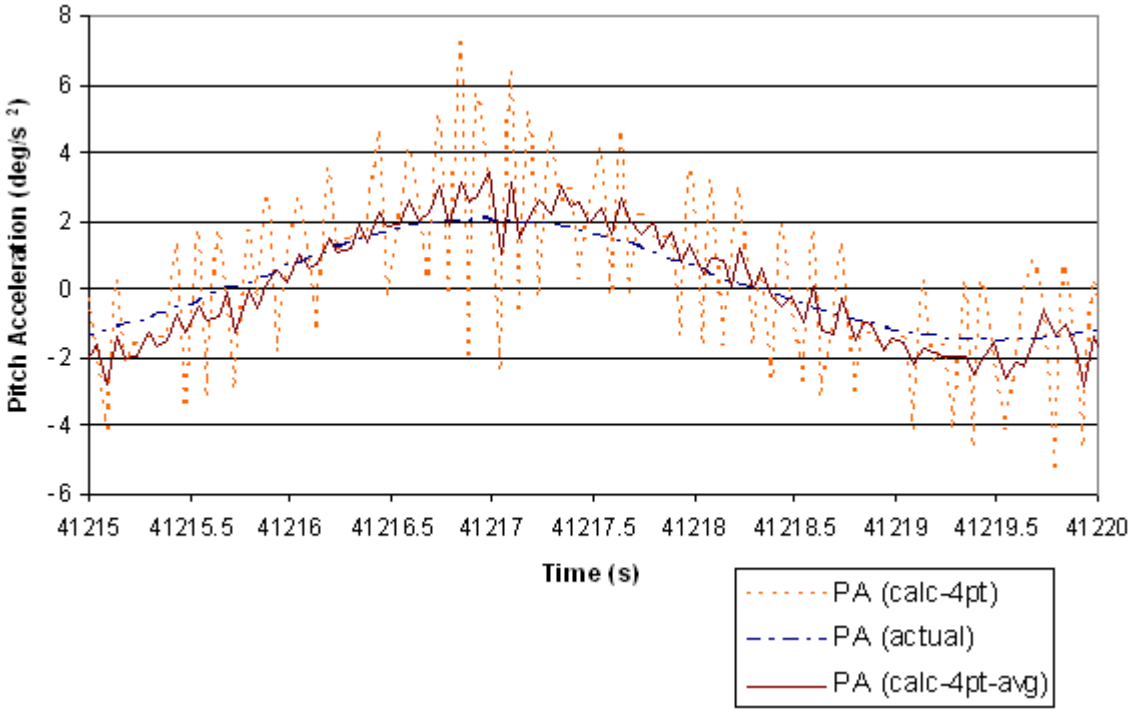

Figure 4.3: Simulated and calculated pitch angular acceleration data.

shown in Figure 4.4. Figure 4.5 displays the percent error between the base acceleration data and the data transformed from various locations. In this case the VFD angular acceleration data were used. This graph shows that the largest differences in the data occurred for the position the farthest away from the hauldown location, and that the peak errors were less than 0.25%.

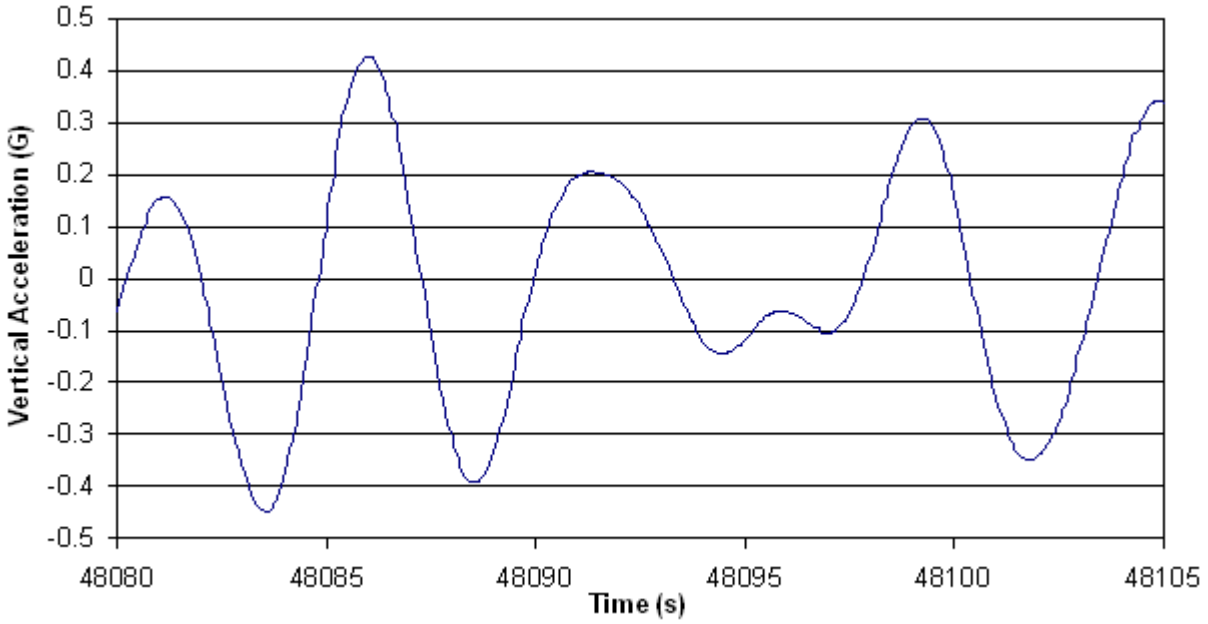

Figure 4.4: Vertical acceleration measurements at hauldown location (-50,0,10).

Figure 4.6 displays the percent errors when the calculated acceleration data are used. In this case the sensor measurements were made at (-70,-10,20) [metres]. One line is with only four-point backwards differentiation, and the second line is with a twenty-point average. (The sampling rate of the VFD used in this experiment was 20 Hz.) Higher and lower point averages were tested, but it was found that matching the points in the average with the frequency of the data arrival resulted in the best compromise between data smoothing and data display update latency. This figure shows that the error in measurement transformations is approximately ten times higher when the angular accelerations are calculated

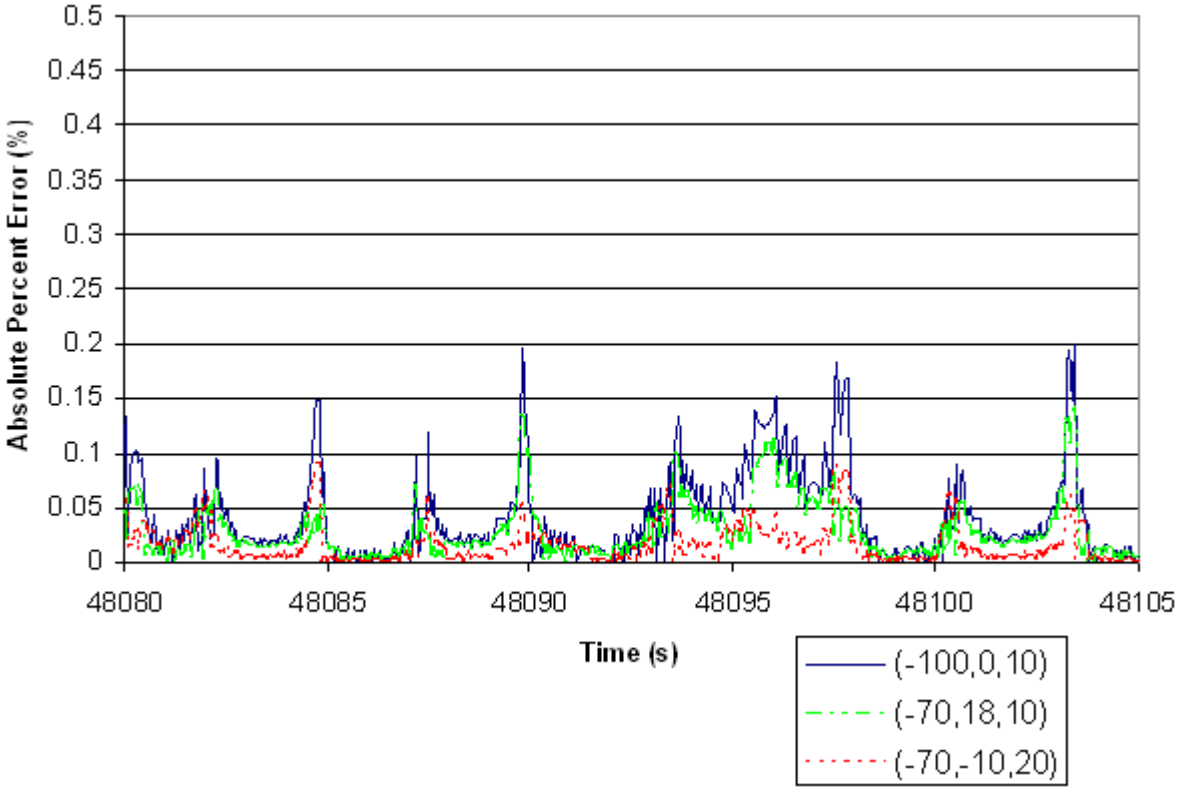

Figure 4.5: Percent errorss in data when transformed to hauldown location using VFD angular accelerations.

using numerical differentiation. However, the errors in this case still remain below 3%, and when the data points are averaged, the error can be reduced greatly, only peaking at a little over 0.5%.

The next step after transforming data from multiple sensors to the same location is to compare the data to detect any faults in the measurements. The numerical values quoted here are maximum peak values that are not consistently maintained through the data. Thus as long as the data fault algorithm watches for data differences above a specific threshold and also for a specific length of time, lower thresholds can be used and data faults can be detected earlier. Full details on the FDMD systems testing are available in the FDMD Systems Test Report [25].

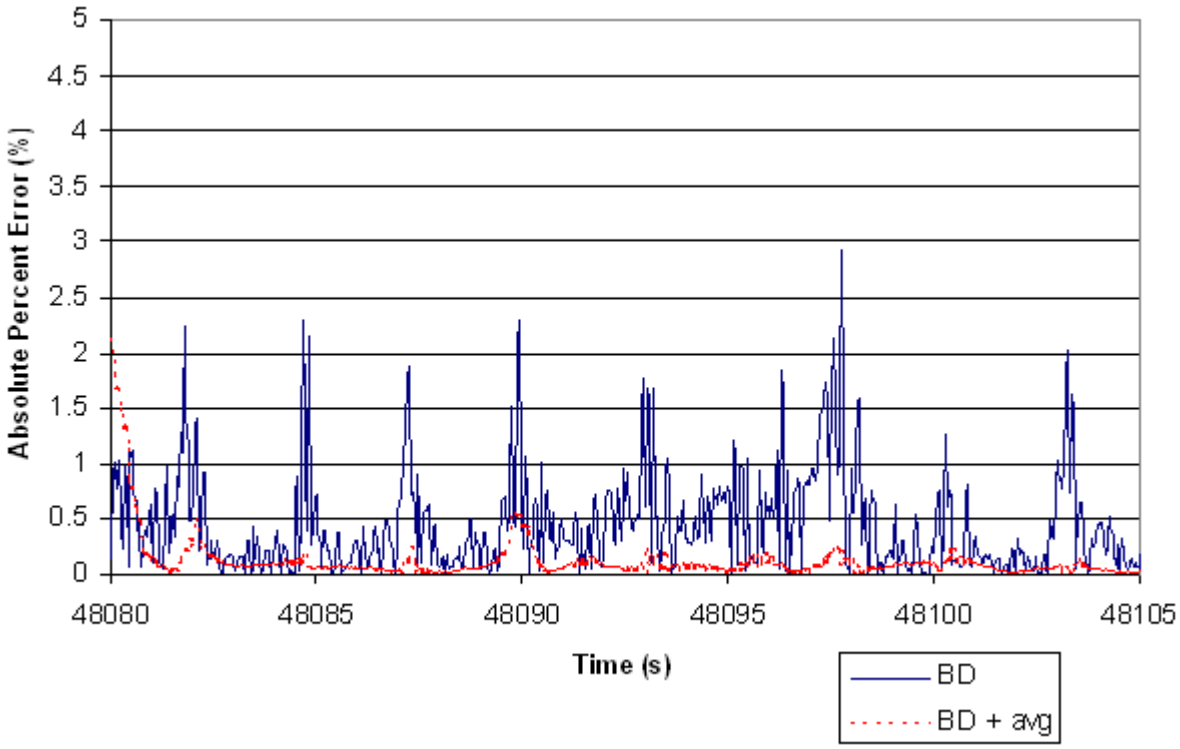

Figure 4.6: Percent errors in data when transformed to hauldown location using calculated angular accelerations.

4.4 User Interface Evaluation

The design of the FDMD's operations user interface is one of the most important aspects of the project. It has gone through a number of revisions, both internally and with aid from GDC human factors engineers.

In order to refine the user interface of the FDMD so that it would be of maximum utility and accepted by actual operators, an evaluation of the FDMD was arranged to take place on Tuesday, May 27, 2008 in Halifax, Nova Scotia. The evaluation was coordinated by three researchers from Carleton University, two employees from General Dynamics Canada (Ottawa) and hosted by General Dynamics Canada (Halifax). Six current military Sea King pilots with LSO experience and two DND observers from 12 Wing, Shearwater took part in the full-day evaluation. The following sections of this chapter describe the software and hardware configurations of the FDMD that were used for the evaluation, the evaluation processes that were used, the qualitative and quantitative results that were obtained, and a list of recommendations on improvements that were made based on the results.

4.4.1 Evaluation Software Configuration

The VFD simulation environment was used to generate simulated ship motions for the evaluation. In addition to being connected to the FDMD, in the evaluation setup, the VFD was also connected to a 3D visualization of the ship motion called the DynamicsViewer. A diagram showing the directions of flows of information in this test system is shown in Figure 4.7

The ship motion is generated based on ship response characteristics computed using the SHIPMO7 hydrodynamics code [27]. These were input and used by the SHPR module embedded within the VFD-RT to generate realistic time histories of ship motion. Taking advantage of the Windows internal message system, the VFD-RT environment produces

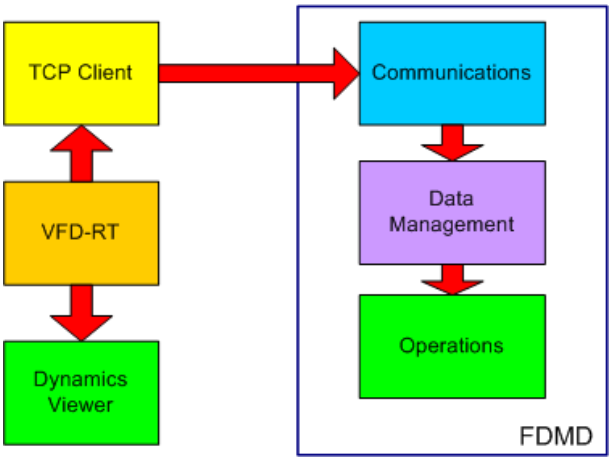

Figure 4.7: FDMD and VFD-RT evaluation software implementation.

simulated motion sensor data and sends it to a TCP client, the role of which is to forward the data in the proper format to the FDMD over Transmission Control Protocol/Internet Protocol (TCP/IP). Simultaneously, the environment sends the simulation data to the DynamicsViewer, a 3D visualization tool developed by Carleton University and extended to connect to the VFD-RT for displaying simulated ship motion. For the evaluation, the DynamicsViewer was configured to provide a visual representation of the ship motion as viewed from the LSO compartment, supplementing the FDMD with visual ship motion cues.

A modified version of the FDMD was used for the evaluation which replaced the normal day/night and deck/flight buttons in the operations user interface with buttons for cycling through the different user interface configurations for each test and for reseting the data recording for each test.

4.4.2 Evaluation Hardware Configurations

In order to use the evaluation time as efficiently as possible, two sets of evaluation equipment were set up. Each consisted of a digital data projector, a dual-core laptop, a portable

computer, and a network switch or router. The digital data projector was connected to the laptop which was running the VFD-RT on one core and the dynamics viewer visualization on the second core. The laptops were networked to the portable computers which ran the FDMD. One portable computer was an Xplore tablet PC which required the subjects to use a touch screen to control the FDMD. The second portable computer was an Argonaut Avalon mini computer connected to a General Dynamics rugged display. Buttons built into the bezel of the rugged display were used to control the FDMD. An image of the evaluation setup from a subject's point-of-view is shown in Figure 4.8.

4.4.3 Definition of Quiescence

A quiescent period is defined as a time when all monitored motion parameters are within limits for performing a specific activity. The two rules for changing quiescence as defined in Section 1.1.2 are: (1) the state is not quiescent when at least one limit is exceeded, and (2) in order to change state from a non-quiescent state to quiescent, each motion which has exceeded its limit must experience a subsequent motion peak below its limit. For the user interface evaluation a third intermediate state was placed between 80% and 100% of each motion limit.

In order to produce useful information from the evaluation results it was decided that an actual quiescent period where a helicopter operation could take place was required to be at least four seconds long. This minimum duration is consistent with operational practice.

4.4.4 Simulation Fidelity

The purpose of the evaluation was to obtain operator feedback on the design and layout of the user interface and not to assess the need or potential performance improvements achievable using such as system. Accomplishing the latter would require a fully immersive environment that at a minimum would include visual, motion, and auditory cues in

Figure 4.8: Evaluation setup from a subject's point-of-view.

addition to the view from the LSO compartment that was provided in this case.

The view provided did however show the ship motion with the roll and pitch orientation of the deck being most apparent. Due to the constant oblique view from the LSO compartment, it was not always possible to distinguish the roll and pitch contributions to deck motion. Despite this, it was apparent that pilot evaluators did make use of the visual cues for indicating or confirming quiescence. The extent to which this was done varied from subject to subject.

4.5 Evaluation Procedure

All of the user interface evaluation sessions involved providing the subjects with different pieces of information in the operations user interface and comparing their feedback and performance using the FDMD. The evaluation was divided into four different sessions, each with additional information provided on the FDMD operations screen. Each session consisted of a primary and secondary task for the subject to complete.

The primary task was to monitor a 3D view of the helicopter and ship, with the camera position in the LSO compartment. A helicopter model was positioned on the flight deck (for example, prior to launch) and periodically changed colour from grey to black. Each colour change to black lasted one second. The subject was then required to indicate verbally when the colour changes occurred. Each experiment coordinator had a list of all of the colour change events and kept a record of each subject's success in noticing them. The purpose of the primary task was to give the subjects something to pay attention to aside from the FDMD display, because in actual operations an LSO's concentration would be on the helicopter and the RSD.

The secondary task was to monitor the FDMD and to press the event marker key whenever the state of the flight deck quiescence first satisfied the applicable quiescence

definition. Each session provided a short amount of time for observing the ship motions before landing periods were to be marked. The 3rd and 4th sessions also included an extra event marking period where a higher sea state was used to simulate ship motions. Every time the event marker button was pressed it was recorded to disk for later analysis.

After each session the subject was asked to answer a few survey questions and to comment verbally on their experience. The question sheets used are included in Appendix A.

Two sessions were run at a time concurrently. Each consisted of one subject, one experiment coordinator, and a survey interviewer. The experiment coordinator was responsible for setting up the software for each simulation and recording the subject's accuracy at noting helicopter colour changes. The survey interviewer was responsible for recording the subject's answers to the multiple choice questions and their comments after each session.

A description of each session is summarized in Table 4.2. Screenshots of each of the user interface variations are shown in Figure 4.9.

4.5.1 Data Mining

In order to obtain accurate quiescent period identification results a software algorithm was written which identified the following:

- Start and end times of each quiescent period;
- Duration in seconds of each quiescent period and whether it was greater than four seconds;
- Whether the event marker was pressed within each quiescent period;
- The location of the two minute mark in each recording;
- Number of quiescent periods marked and total number of quiescent periods; and

Table 4.2: Evaluation descriptions and durations.

Session	Description	Sea State (SS)	Observing Time (minutes)	Marking Time (minutes)
1	Instantaneous value indicators only	Upper SS5	2	5
2	Instantaneous value indicators and data history	Upper SS5	2	5
3(a)	Instantaneous value indicators, data history and two-state quiescent period indicator	Upper SS5	2	5
3(b)		Upper SS6	-	5
4(a)	Instantaneous value indicators, data history and three-state quiescent period indicator	Upper SS5	2	5
4(b)		Upper SS6	-	5

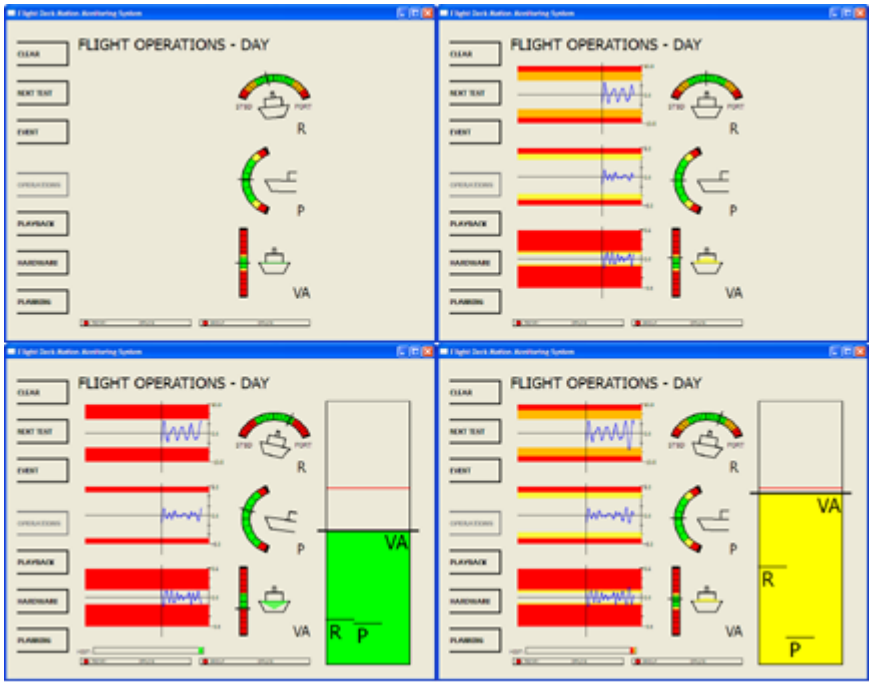

Figure 4.9: Screenshots of the user interface variations for each session. The top two correspond to sessions 1 and 2, and the bottom two correspond to 3 and 4.

- Number of four second quiescent periods marked and total number of four second quiescent periods.

Due to the fact that this analysis considered the length of quiescent periods, if a recording started or ended in a quiescent period, then those periods were ignored in order to ensure consistency in the results.

4.6 Results and Discussion

For reference, graphs of each session and lists of quiescent period time ranges are included in Appendix B.

4.6.1 Helicopter Colour Change Marking

All subjects were very accurate at the primary task of noting when the helicopter changed colour. The only mistakes appear to be random human error and no conclusions are able to be drawn from these data.

4.6.2 Quiescent Period Marking

One limitation of real-time motion parameter monitoring is that it is difficult to determine *a priori* how long a quiescent period will last. While it was suggested in a previous section that displaying accurate ship motions may have an effect on evaluating the user interface, it did provide a way for operators to judge whether a quiescent period would be long enough for a helicopter takeoff or recovery.

The results of the quiescent period event marking were analyzed in three ways: (1) to check how many times the subjects pressed the event marker when the ship was not in a quiescent period; (2) to compare how many quiescent periods were marked to the total number of quiescent periods; and (3) to compare how many quiescent periods of a four second duration were marked to the total number of four second quiescent periods. A duration of four seconds was chosen because during the Halifax-class flight deck certification trials carried out by DRDC in the 1990s it was estimated that four seconds is enough time for a takeoff or recovery.

Misplaced Event Marker Results

The results of marking quiescent periods were greatly affected by two primary factors. The first is whether the subjects considered the 3D ship animation to be an accurate representation of the orientation of the ship and the second is the assumption that the subjects became more proficient at marking the helicopter colour changes as they proceeded

through the trials.

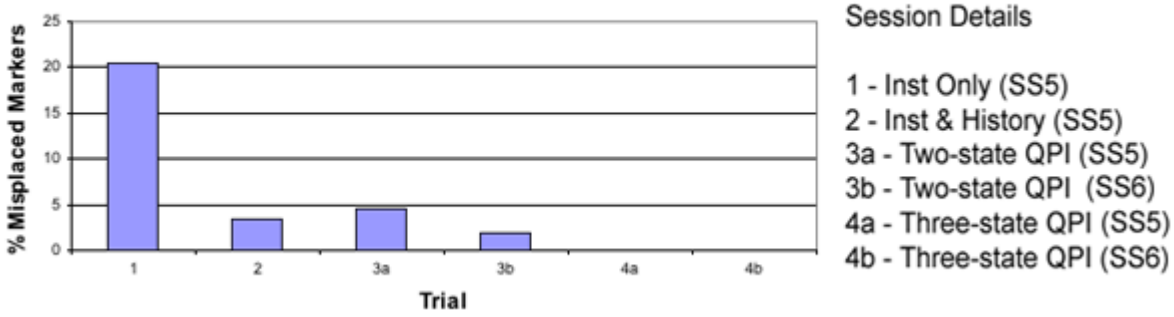

Figure 4.10: Average percentage of misplaced marker events.

The average percentage of misplaced event markers for each session can be seen in Figure 4.10. Misplaced markers were markers that were placed when the ship deck was not quiescent. These results are displayed as a percentage of the total number of at least four-second quiescent periods in each session. It clearly shows that as the evaluations proceeded the subjects pressed the event marker at the wrong times less frequently. No markers were misplaced in the higher sea state sessions as the subjects were very cautious about marking quiescent periods.

Quiescent Period Marking Results

As stated before, data were analyzed in two different ways. The first was to compare how many quiescent periods were marked to how many occurred with no regard to how long each quiescent period was. The second way was to measure how many quiescent periods were marked that lasted at least four seconds and compare it to the total number of quiescent periods that lasted at least four seconds. Overall there were very few cases where quiescent periods that were not four seconds in length were marked. The results are displayed in Figures 4.11 and 4.12. Note that on the graphs, trials 3(b) and 4(a) have

been switched so that the sea state 6 trials are the two on the right side of the figure.

The results show different trends for the high and low sea state situations. In the lower sea state trials the proficiency of the subjects increases as they are given additional information. It is interesting to note that the addition of the yellow quiescent state did not affect the number of quiescent periods marked but did increase the number of periods marked that were at least four seconds in duration. It is difficult to make conclusions due to the fact that trials 3a and 4a had different data and a different number of quiescent periods but it is possible that the addition of the yellow state in the quiescent period indicator increased the confidence of the subjects because in longer quiescent periods there is a greater chance of the status changing from red to yellow to green rather than just from red to yellow.

The dip in the results of Figure 4.12 between trials 1 and 2 is due to the fact that all of the quiescent periods in trial 2 are of at least four seconds while this is not the case for the other trials. Thus, it retains the same value percentage marked as in Figure 4.11 while all of the other values are increased due to the lower number of quiescent periods when only counting those of four-second length.

In the higher sea state it is shown that the subjects became more cautious with marking quiescent periods when the yellow state was added. This is consistent with the fact that some of the quiescent periods in the higher sea state were only displayed as a yellow state regardless of their length.

The numerical results are shown in Table 4.3.

4.6.3 Survey Results

After each session a series of survey questions were posed to the subject. Each survey question also included a follow up qualitative question to encourage feedback in addition to general feedback on the session.

Table 4.3: Quiescent period marking numerical results.

Trial	1	2	3a	4a	3b	4b
Avg. marked	4.60	5.00	5.60	5.60	2.40	1.40
σ	1.58	1.35	2.39	2.39	2.01	4.03
Num. QP	11	11	11	11	11	12
Num. 4s QP	9	11	10	8	3	5

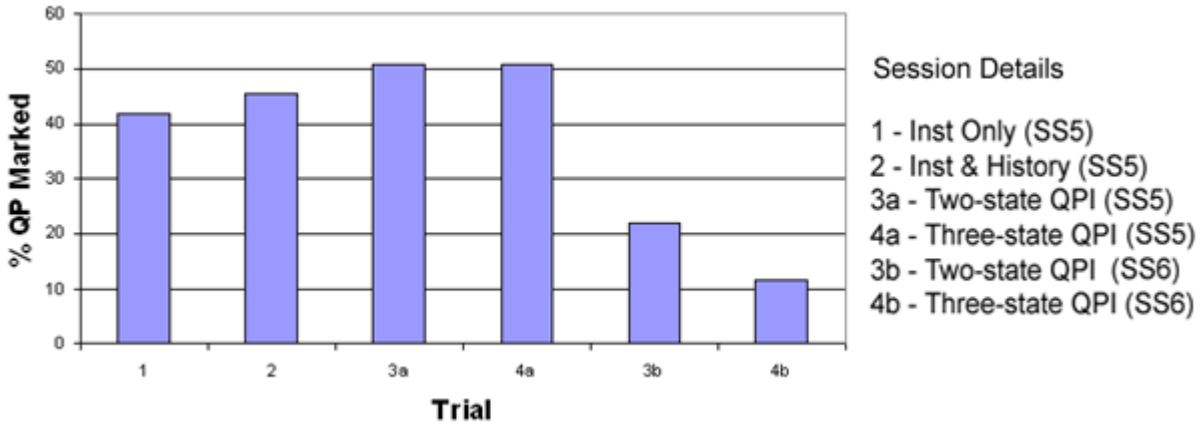

Figure 4.11: Average percentage of quiescent periods marked for each trial.

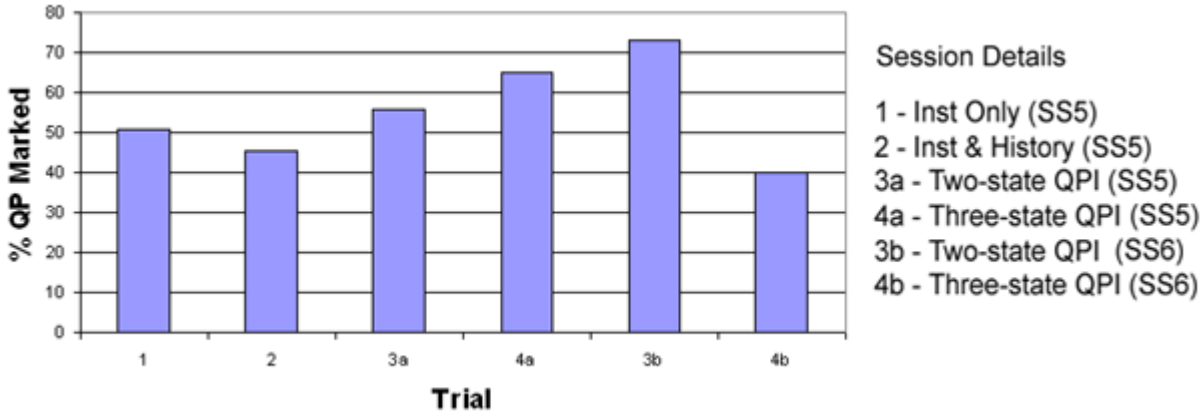

Figure 4.12: Average percentage of four-second quiescent periods marked for each trial.

Session 1 - Instantaneous Indicators Only

Table 4.4 contains the questions posed to each subject after session 1 and the averages of the answers.

Table 4.4: Session 1 - survey questions and results.

Label	Question	Average	Scale
1a	Is it clear which motion each instantaneous indicator is measuring?	4.67	1=not clear, 5=clear
1b	How easily are you able to determine the approximate magnitudes of ship motions using the scales provided?	3.92	1=difficult, 5=not difficult
1c-i	Rate the usefulness of the pitch indicator.	4.33	1=not useful, 5=useful
1c-ii	Rate the usefulness of the roll indicator.	4.33	1=not useful, 5=useful
1c-iii	Rate the usefulness of the vertical acceleration indicator.	3.83	1=not useful, 5=useful

The qualitative evaluator feedback is summarized in the following statements:

- The green area of the vertical acceleration instantaneous indicator is too small. One possible solution is to use a nonlinear scale to make the green areas larger.
- Since vertical acceleration is often the deciding factor, its placement should be optimal.
- The current LSO indicator shows combined pitch and roll so that an operator only has to look in one place. Doing the same thing for roll/vertical acceleration could be helpful.

In general, the subjects said that they spent most of their time monitoring the ship motion using the 3D display and only checked the FDMD when the ship's motions were visually identifiable as possibly within limits.

Session 2 - Instantaneous Indicators and Data History

Table 4.5 contains the questions posed to each subject after session 2 and the averages of the answers.

Table 4.5: Session 2 - survey questions and results.

Label	Question	Average	Scale
2a	Do you believe the history readout provides useful information?	2.70	1=not useful, 5=useful
2b	Did you find it easier to identify lulls in ship motion with a data history available?	3.50	1=harder, 5=easier, 3=no change
2c	Is the time history length appropriate?	4.00	1=shorter, 5=longer, 3=no change
2d	Would being able to change the time history length be useful?	3.83	1=not useful, 5=useful

The qualitative evaluator feedback is summarized in the following statements:

- A time history length of 5/10/15 minutes would be more useful for flight planning and dealing with rougher seas.
- Time history could be useful for training or subsequent analysis.
- Vertical acceleration history was used to estimate when the next quiescent period would occur in some cases.

The general response in this session was that the subjects would only find the data history useful if they were in a situation where they would have time to interpret the graphs. There was not a clear response on whether being able to adjust the time history length would be useful but it was agreed that a longer time history could be useful.

Session 3 - Two-State Quiescent Period Indicator and Quiescence History

Table 4.6 contains the questions posed to each subject after session 3 and the averages of the answers. Question (3b) was not numerical or a yes/no question so the answers are included in the qualitative feedback.

Table 4.6: Session 3 - survey questions and results

Label	Question	Average	Scale
3a	Do you believe that the QPI provides useful information?	4.33	1=not useful, 5=useful
3b	Do you get an immediate impression that any part of the QPI needs to be improved on?	-	results in discussion
3c	At any time did the QPI display information contrary to what you expected?	1.83	1=never, 5=often

The qualitative evaluator feedback is summarized in the following statements (including the results of question (3b)):

- When the green bar is really small it occasionally leaves peripheral vision.
- Short green periods can be misleading, especially in higher sea states.
- The quiescent period indicator (QPI) was useful for confirming the visual identification of a quiescent period.
- The inner details of the QPI were rarely used.
- With the QPI displayed the other user interface elements were used much less.
- Both responses to the QPI's movement were received (smooth movement preferred /discrete motion is fine).
- Screen customization (positions/sizes) would be useful.

- Since green intuitively means 'go', and may not mean 'go' with the QPI, that colour is somewhat inappropriate.
- History would be more useful as numerical values (ie. previous peak values, average time between quiescent periods, etc).
- Putting previous peaks on the QPI could be useful information for avoiding very short quiescent periods.
- Non-QPI data may be useful for particular situations, training, and possibly at night.
- There were times when the QPI showed an unintuitive status because of vertical acceleration (as was expected).

The general consensus among all of the subjects was that with the addition of the quiescent period indicator, the remaining user interface elements became considerably less useful. For the most part the response that the QPI would be very useful as something they could observe in their peripheral vision while keeping most of their attention on the helicopter in a low hover situation.

Session 4 - Addition of a Three-State Quiescent Period Indicator

Table 4.7 contains the questions posed to each subject after session 4 and the averages of the answers, except for question (4c) which was answered as yes or no.

The qualitative evaluator feedback is summarized in the following statements:

- There was too much information and it was easier to just concentrate on the QPI being red or not red.
- Yellow is difficult to pick up in periphery (yellow/green are difficult to differentiate).
- It is better for yellow to mean one of either approaching or leaving danger.

Table 4.7: Session 4 - survey questions and results.

Label	Question	Average	Scale
4a	Do you think a third intermediate state for communicating quiescence is useful?	2.00	1=not useful, 5=useful
4b	Do you find the meanings of the red/yellow/green states clear?	4.00	1=not clear, 5=clear
4c	Do you think the colours used are appropriate?	4x yes, 2x no	yes, no
4d	Would having the yellow limit larger/smaller improve its usefulness?	2.40	1=smaller, 5=larger, 3=current is ok

- Yellow helps a bit when judging trends.
- Waiting through a yellow for a green could be too time costly in real operations.

The yellow state replaces part of the green motion range, and basically means that it is safe to land, but the deck is close to the limits. In general, there was not much positive feedback about the addition of the yellow state. This may have been in part due to lack of understanding of its role by the evaluation participants.

General Evaluator Feedback

The feedback in this section was gathered during a general discussion after all of the evaluation sessions had been completed.

- FDMD position is very important. If the QPI is not in the peripheral vision of the pilot he will not use it because 90% of his attention is on the ship deck during low hover. (Note that 90% was an estimate by one of the subjects, and not a measured datum.)
- Additional information on the FDMD screen would be useful, such as true/relative wind speed and heading, and RADHAZ[1] information.

[1]RADHAZ is extremely high power electromagnetic radiation generated by a ship's radars.

- Last peak experienced would be useful in addition or instead of the time history. These typse of data would be useful for relaying to the ship bridge and helicopter.
- Layering operations user interface elements would be useful so that items that are not being used could be hidden.
- Overall some of the subjects saw the FDMD as having the potential to reduce radio chatter by providing sea condition information in multiple locations.
- The green quiescent status should be more prominent on the screen so that it can always be seen using peripheral vision. Two options for this would be to put the bar upside down or simply have the entire bar change colour and not move.
- One of the disadvantages of using flight planning to identify best flight operations headings is that often the ship only turns into the wind for a quick launch/recovery and then returns to its original course. Overall flight planning would be more useful on the bridge, with best heading predictions available on the bridge and expansive history and motion min/max data available in the LSO compartment.

4.7 Recommendations

4.7.1 FDMD Modifications

Based on the evaluation results, a list of recommended modifications to the FDMD was created. Short-term modifications to the FDMD were carried out, and more complex modifications were left as future recommendations only.

Short-Term Modifications

The following modifications were made based on the evaluation results.

- The quiescent period indicator was adjusted so that the coloured bar is stationary and continues to change colour while the individual parameter bars continue to move.
- The yellow quiescent status was removed from the quiescent period indicator.
- The size of the green area of the vertical acceleration indicator was increased by reducing the displayed minimum and maximum limits.

Future Modification Recommendations

The following tasks are recommended for future modifications to the FDMD.

- Reconsider and re-evaluate the role of the yellow state in quiescence status monitoring.
- Create multiple task-oriented operations screens that an operator can cycle through with a single button.
- Remove the pitch indicator and/or combine the roll and vertical acceleration indicators into a single element.
- Investigate the use of nonlinear axes scaling for use with the instantaneous value indicators.

4.7.2 Recommendations for Future Evaluations

Below is a list of suggestions for future FDMD evaluations.

- Include a trial with no FDMD instrumentation.
- Include a trial simulating night conditions where operators would be more dependent on the FDMD instrumentation.

- Consider using a single set of motion data for consistency, at the risk of subjects memorizing the pattern of quiescent periods.
- Simulate a helicopter approach and landing operation.
- Implement variations of use of the yellow limit such as replacing a portion of the green or red limits area.

Chapter 5

Discussion and Conclusion

5.1 Discussion

The Flight Deck Motion Display (FDMD) is a flexible system designed to provide real-time data monitoring tools to aid in the tasks of helicopter-ship operations. The FDMD is designed to help an LSO predict when a lull will occur in a ship's motion, but without performing any type of predictive calculation. It does this through the use of a quiescent period indicator, which allows the operator in a quick glance to determine (a) whether the ship's motions are within operational limits or not, (b) which motions are beyond their limits if they are, and (c) which of the ship's motions are close to their limits if they are not. This system essentially answers the question "is it safe to land now?", but is that what flight deck operators really need?

Initial user interface evaluations with actual pilots resulted in a variety of feedback, and there are some issues that stand out. The role that the FDMD plays in helicopter operations is extremely important. If one were to build a predictive flight deck monitoring system that was correct at predicting quiescent periods 100% of the time, or guaranteed that every quiescent period indicated would be at least four seconds long, then such a

device would be widely accepted, and fully instrumented helicopter landings would take place. This unfortunately is not the case. Any flight deck motion monitoring system is only going to be secondary to current procedures, operating practices, and the experience of pilots in helicopter-ship operations.

One of the conclusions from the evaluations that can be clearly drawn is that the subjects found the quiescent period indicator useful for reaffirming their intuitive beliefs that the flight deck was entering a quiescent period. Current flight deck operating procedures do work. The advantage of monitoring flight deck motions with a system like the FDMD is that it grants additional information to an LSO in high sea states when identifying the onset of quiescent periods can be difficult. In these situations, actions need to be performed quickly and efficiently, as dangerously fluctuating vertical accelerations can cause unexpected incidents during operations. Providing a display like the FDMD can help reduce pilot workload and overall help identify situations where vertical accelerations may cause the flight deck to move in unexpected ways.

It is very important that the context of the situation that a device like the FDMD will be used in be taken into account when fine-tuning its behaviour. Much of the data that were gathered during the evaluations would have been different if the subjects did not have the 3D image of the flight deck to base their decisions on. If the subjects had also been exposed to sensory inputs of physical motion and sound, the results may have shown that the FDMD was used in different ways, and that only specific pieces of information and presentation methods would useful to an LSO engaged in flight deck operations.

Positioning of the FDMD in the LSO compartment is also extremely important. During helicopter landings an LSO does not have time to closely monitor a video display as their attention is focused on the helicopter and the landing system. For an FDMD to be used during an operation like this, it must be in the pilot's peripheral vision, and the colours and symbols used must be visible in their periphery. For example, many of the subjects

were not able to distinguish the difference between yellow and green in their peripheral vision during the evaluations.

Regardless of its final operating role, the FDMD system designed and presented in this thesis can meet that challenge. The modular nature of its hardware architecture allows the use of a system that can easily range from a single sensor and computer to a wide range of measurement devices and monitoring stations. Software modularity based upon a standard of interoperability and communication easily allows additional software modules to be added to the FDMD as required. The stable software structure of the FDMD is also an asset in a field where software failure at specific times can be critical.

The format and behaviour of the quiescent period indicator has been a recurring topic of discussion throughout the FDMD project. The original vision of the operation of the indicator by Colwell [2] was of a collection of bar graphs that displayed the instantaneous values of each motion parameter. The parameter the closest to its limit was selected as the single overriding parameter when only a single bar was to be shown. The first version of the FDMD followed this design, but it was found that as all motion parameters are oscillatory, there were situations where the quiescent status could be red, but all parameters passing through zero simultaneously could give the opposite impression as the red bar would barely be visible at that time.

The second revision of the FDMD replaced the display of instantaneous values in the QPI with bar heights that represented the last peak value that each parameter had experienced. This removed the problems with the previous revision as any parameter with peak values in the red would remain displayed in the QPI as both red and over its limits, even as it passed through zero. Initially there was reluctance from some on accepting this display method as it results in discontinuous motion of the QPI. However, given that the intended use of the QPI was to get as much information from it as possible in a single glance, and the disadvantages of the previous QPI version, many were convinced of the

usefulness of this revision.

After the user interface evaluations, a revised quiescence definition was applied to the FDMD, where the current display method for roll and pitch was kept, but quiescent status calculations for vertical acceleration were modified to take instantaneous values of acceleration into account, and not wait for a peak within limits before changing quiescence status from a higher to a lower state. This allowed the discontinuous variation of the vertical acceleration level to be removed. Thus in the current version of the FDMD, the QPI updates roll and pitch angles when they experience peak values, and continuously updates vertical acceleration. This behaviour of acceleration monitoring is very similar to what was used in the first version of the FDMD, and thus has the potential to suffer from the same problems. However, implementation of one of the suggestions from the evaluation has removed this possibility. One of the recommendations was to not animate the entire coloured bar; just keep it stationary and change its colour to the current quiescent status. Thus the motion of the vertical acceleration marker now gives an impression to an operator as to how quickly ship motions are changing without compromising the ability of the QPI to convey the current quiescent status. Further evaluations will need to be carried out in the future as the FDMD continues to be refined.

Flight planning capabilities of the FDMD has been a topic of interest in the FDMD project. One of the requirements listed in the DND draft specification (not included in the FDMD requirements) is that the system should be capable of supporting the selection of best course and speed for helicopter operations. It has been determined that this capability is not possible to perform with only motion sensor data. In order to perform an analysis of the potential usefulness of a course change the FDMD would either require: (a) independent measurement of wind direction, wind speed, and ship heading information; or (b) access to the ship's data bus.

5.2 Conclusions

The FDMD project can be considered a success. A flexible, reliable, expandable flight deck motion monitoring system has been developed that meets all of the software requirements and hardware requirements that are needed for the successful implementation of a prototype system. Operator feedback on the system in its current state is positive. Specifically, the FDMD provides additional useful information that supports the LSO role and hence contributes to the safety of flight deck operations. Some specific conclusions include:

1. Integration of the FDMD's functionality into current flight deck operating procedures is important. The additional information it provides should supplement the LSO's own experience and judgement. The FDMD performs well as a device used to confirm the LSO's identification of a quiescent period from visual and motion cues.

2. Positioning of the FDMD in the LSO compartment is very important. If the LSO cannot see the FDMD in their peripheral vision, it will not be used in critical operations.

3. Simulation fidelity during operator evaluations can have a drastic effect on observed results. The availability of accurate visual and motion cues can significantly change the way that the QPI is used during evaluations and reveal unexpected issues involving the role it plays in flight deck operations.

4. User interface colour selection and behaviour needs to be taken into account. Green intuitively means go, and if the case is otherwise, it may not be appropriate to use. Additionally, care should be used when using yellow when transitioning between green and red in both directions.

5. Display of peak data values is a useful alternative to constantly updating instantaneous values.

5.3 Recommendations

- Carrying out an FDMD evaluation that includes physical motion cues would provide extremely useful information on what role the FDMD should play in actual flight deck operations, and may reveal previously unconsidered aspects of its positioning, behaviour, and potential uses.

- Future evaluations should include a test where no FDMD is available, and a simulation of night operations, as an LSO would likely be more dependent on instrumentation in such a situation.

- Landing of UAVs on flight deck is a situation where the pilot is located on the ship the entire time. An FDMD system has potential uses in providing real-time motion data in these operations. Its utility in this context should be investigated.

References

[1] Joint Maritime Command 1 Canadian Air Division. *Shipborne Helicopter Operating Procedures (SHOPS)*, 2003.

[2] J. L. Colwell. *Flight Deck Motion System (FDMS): Operating Concepts and System Description.* Defence Research and Development Canada, DRDC Atlantic, January 2004.

[3] J. L. Colwell. Real time ship motion criteria for maritime helicopter operations. In *ICAS Congress*, Toronto, Canada, 2002.

[4] Canadian Department of National Defence. Flight deck motion display requirements specfication. Accessed Sept. 2008.

[5] Fugro. Helideck motion monitoring. http:// www.geos.com/ dl/ datasheets/ rt02_helideckmotionmonitoring.pdf, Accessed Jan. 2008.

[6] Ship Motion Control. SMChms. http:// www.shipmotion.se/ download/ SMChms.pdf, Accessed Jan. 2008.

[7] Kongsberg. Helideck monitoring system - HMS 100 - Kongsberg Maritime. http:// www.km.kongsberg.com/ ks/ web/ nokbg0240.nsf/ AllWeb/CC83120083313D80C125719D00459BF0?OpenDocument, Accessed Aug. 2008.

[8] Miros. MIROS AS - Helideck monitoring systems (HMS). http:// www.miros.no/ helideck_monitoring.php, Accessed Aug. 2008.

[9] Communications Computer Intelligence Integration Systems. Helicopter take-off and landing system. http:// www.ccii.co.za/ products/ htls.html, Accessed Aug. 2008.

[10] Aeronautical and General Instruments Limited. AGI - ship helicopter operational limits display system (SHOLDS) - SHOL aircraft helo envelope flight course fox corpen. http:// www.agiltd.co.uk/ marine_instrumentation/ operational_limits/, Accessed Aug. 2008.

[11] Prism Defence. heliSAFE. http:// www.prismdefence.com.au/ defence/ helisafe.htm, Accessed Aug. 2008.

[12] Siri Marine. Safetymax motion monitoring. http:// www.sirimarine.nl/ siri/ sm_motion.htm, Accessed Jan. 2008.

[13] Ship Technology. Siri marine - ship technology. http:// www.ship-technology.com/ contractors/ controls/ siri-marine/ siri-marine3.html, Accessed Aug. 2008.

[14] R. Rouvari Oy. R. ROUVARI OY. http://www.rrouvari.fi, Aug. 2008.

[15] R. Rouvari Oy. Instructions for the operation of HULLMOS. http:// www.rrouvari.fi/ Manuals/ Hullmos_User_Instructions.pdf, Accessed Aug. 2008.

[16] Sirehna. Ship motion and hull monitoring system. http:// www.ec-nantes.fr/ Sirehna/ document/ Monitoring.pdf, Accessed Jan. 2008.

[17] D. Carico and B. Ferrier. Evaluating landing aids to support helicopter/ship testing and operations. In *IEEE Aerospace Conference*, page 13, 2006.

[18] B. Ferrier, D. Carico, S. Crockatt, and J. R. Von Hor. Visual landing aid evaluation using simulated dynamic interface methods in the manned flight simulator (MFS). In *American Helicopter Society 64th Annual Forum*, Montreal, Canada, 2008.

[19] T. Gray and S. MacTaggart. A safety index for embarked helicopters by real-time simulation. In *American Society of Naval Engineers Symposium: Launch and Recovery of Manned and Unmanned Vehicles from Surface Platforms*, Annapolis, Maryland, 2008.

[20] N. Bourgeois and R. Langlois. FDMD prototype hardware report. Technical Report ADL/08/NB/6, Carleton University, Sept. 2008.

[21] N. Bourgeois and R. Langlois. FDMD prototype software report. Technical Report ADL/08/NB/7, Carleton University, Sept. 2008.

[22] Trolltech. Code less. create more. deploy everywhere. - Trolltech. http://trolltech.com, Accessed Sept. 2008.

[23] Trolltech. Qt cross-platform application framework - Trolltech. http:// trolltech.com/ products/ qt/, Accessed Sept. 2008.

[24] Trolltech. Online reference documentation. http:// doc.trolltech.com, Accessed Sept. 2008.

[25] N. Bourgeois and R. Langlois. FDMD systems test report. Technical Report ADL/08/NB/8, Carleton University, Sept. 2008.

[26] D. V. Griffiths and I. M. Smith. *Numerical Methods for Engineers*. CRC Press, 2006.

[27] N. Bourgeois, R. Langlois, and K.W. Tsui. FDMD user interface evaluation report. Technical Report ADL/08/NB/3, Carleton University, Sept. 2008.

Appendix A

FDMD Evaluation Survey Forms

Appendix B

Ship Motions Used for Each Trial and Quiescent Time Periods