Ship On Sea Roll Simulator

(Team U1 Report)

François-Michel Brière (4474082)
Negar Famili (4481887)
Marc-André Laverdière (4469526)
Yann McCready (4463072)
Frédéric Rioux (4436288)
Jia-Wei Zhang (4466926)

Submitted to Dr. A. Allievi
Section U

March 28, 2002

Concordia
UNIVERSITY

Table of Contents

Table of Figures

Table of Tables

Table of Screenshots

Part I

SOS Roll Simulator

(Software Requirements Specifications)

1. Introduction

This part consists of an SRD (Software Requirement Document) defining the overall functionality of the *SOS Roll Simulator* system. The development team will use this document as guidelines for the development of the system, and nothing but what is specified in this document will be implemented.

1.1. Purpose

The purpose of this SRS (Software Requirement Specification) is to clearly set the features that will be implemented by the development team for the *SOS Roll Simulator*, as well as the features' sequence of implementation, in a manner to avoid confusion regarding the final product. The targeted audience of this document is both the client and the development team.

1.2. Scope

The *SOS Roll Simulator* is to be a working scaled-down version program of a real simulator system to be sold to customers. This customer's demo will show the potential users an overview of the real system's capabilities.
The *SOS Roll Simulator* will be produced as a standalone simulation to be run on *Windows*® 2000® or XP® desktop machine.

The simulator, based on user-defined values, will approximate the movement of a ship in waves, plot the roll angle and roll speed as 2D and 3D graphs (θ vs. t, θ' vs. t and θ vs. θ' vs. t).

The system will also display a view of the crossection of the ship, facilitating analysis of the ship's movements.

The system only simulates the effect of a side wave on the ship's angle, and doesn't determine the ship's reaction in case of extreme angles for complete capsizing.

The *SOS Roll Simulator* includes many features that will be implemented sequentially, as described in Table 1 below.

Release #	Functionality Overview
0.1	Mathematical Engine offers approximation of results, data must be plotted using a separate software package (Such as Excel or GNUplot)
0.2	Mathematical Engine offers improved algorithms and graphics are produced by the Plotting Engine
0.3	GUI Integration for user-friendliness, Mathematical Engine is completed and optimized
1.0	Full-featured GUI with 2D graphical plotting of data
1.1	Full-featured GUI with 3D graphical plotting of data
2.0	Full-featured GUI with 3D graphical plotting of data and save/load options for data files
2.1	Full-featured GUI with 3D graphical plotting of data and save/load options for data files and graphics/animations

Table 1: Implementation Sequence

Prototype #	Functionality Overview
0.1	Validation of Approximation Algorithm using Microsoft Excel.
1	Mathematical Engine offers approximation of results, data must be graphically plotted using a separate software package (Such as Excel or GNUplot).
2	Mathematical Engine offers improved algorithms and Plotting Engine provides 2D graph representations.
3	GUI Integration for user-friendliness and implementation of further algorithms.

Table 2: Sequence of Prototypes

1.3. Definitions, Acronyms and Abbreviations

The following is a list of definitions, acronyms and abbreviations that will facilitate the readers understanding of this document

1.3.1. Definitions

T	Time
Δt	Time Step
θ	Roll Angle
θ'	Roll Angular velocity
θ''	Roll Acceleration
θ_0	Initial Roll Angle
θ'_0	Initial Roll Angular velocity
θ''_0	Initial Roll Acceleration
ω_w	Wave Natural Frequency
ω_e	Wave Encounter Frequency
R	Frequency Ratio
B	Relative Fluctuation of Transverse Metacentric Height
I_{xx}	Roll Virtual Mass' Moment of Inertia
K_{xx}	Roll Virtual Radius of Gyration
L	Ship's Length
B	Ship's Beam
D	Ship's Depth
T	Ship's Draft
KM	Ship's Metacentric Height
GM	Ship's Transverse Metacentric Height
W	Ship's Displacement
V	Ship's Speed
C_0	Ship Block Coefficient
C_u	Ship's Upper Deck Area Coefficient
A_d	Ship's Projected lateral area of superstructures and deck houses above main deck
H_e	Effective depth of the ship structure
LBP	Ship length between perpendiculars
g	Acceleration of Gravity
θ_{max}	Maximal Rotation Angle: beyond this value, the ship is considered as having flipped

Table 3: Definitions

1.3.2. Acronyms

SRS	Software Requirements Specifications
GUI	Graphical User Interface
MFC	Microsoft Foundation Classes
OOD	Object-Oriented Design
PDI	Parameter Definition Interface

Table 4: Acronyms

1.4. References

Dr. Alejandro Allievi: Algorithms to Integrate the Equation of Roll Motion
Dr. Alejandro Allievi: Ship stability via the Mathieu equation
S.L. Pfleeger: Software Engineering, Theory and Practice (as summarized by Dr. Allejandro Allievi)

2. Overall Description

2.1. Product Perspective

The product in itself doesn't constitute a useful simulator for advanced ship engineering. However, it can help ship engineers quickly evaluate design impact on a ship's ability to survive side waves. It also offers this capability at very low implementation cost, since almost every desktop computer can run the simulation in a few seconds.

This gives an advantage to the sales personnel who make demonstrations to potential clients using laptops or any on-location desktop.

Since the system is entirely standalone, no particular consideration to inter-compatibility has been given.

2.2. Product Functions

A complete set of functions for the software package has been derived, based on preliminary specifications from the customer and has been summarized in Figure 1 and Table 6.

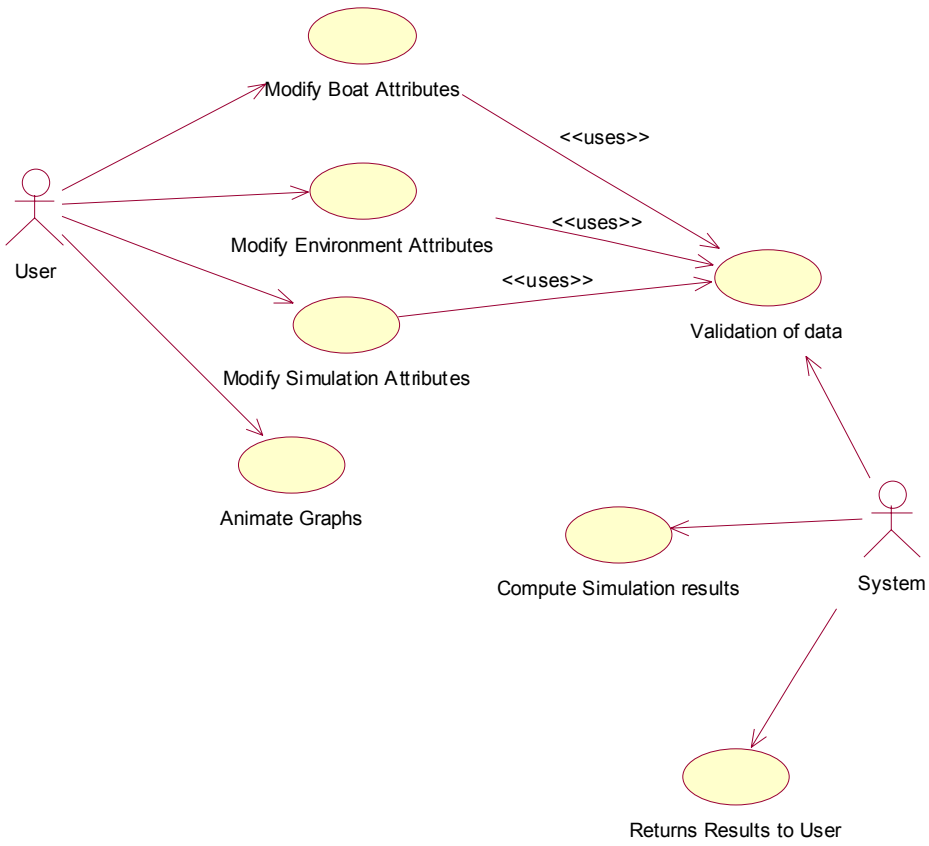


Figure 1: Use Cases for SOS Roll Motion Simulator

Priority	Detail
1	Must have
2	Should have
3	Nice to have
4	Unprioritary
removed	Removed from requirements

Table 5: Priority Scale

ID	Function	Description	Priority
1	Mathematical Engine	Offers an object-oriented implementation of approximation algorithms for solving ODE. The Engine stops computing when the angle reaches θ_{max}.	1
2	Data Bridge	Offers a gateway for the data to be passed from the Mathematical Engine components of the system.	Removed (see section 3.1 for reason)
3	2D Data Plotting of Roll Angle	Displays the graph of the roll angle vs. time.	1
4	2D Data Plotting of Roll Speed	Displays the graph of the roll speed vs. time.	1
5	Integrated Interactive GUI	Allows setting of parameters, run simulations and visualize results without restarting the program. The main window is divided as described in §2.3.2.1 This module is also responsible to warn the user of any error in parameters and/or general errors.	1
6	3D Data Plotting of Roll Speed and Roll Angle	Displays a graph of roll angle • roll speed • time.	2
6.5	2D Data Plotting of Comparison	Displays the graph of the roll speed vs. roll angle	3
7	3D Data Plotting of Ship's Crossection	Displays an animation of the motion of the crossection of the ship trough the time of the simulation.	2
7.5	Display of Graphs	Displays a graph selected by the user in full screen mode.	Removed (see section 3.1 for reason)
8	Playback Controls	Define controls for playback of the animation sequence yielded by (7). Playback will highlight the progression of the simulation on (7), (6), (4) and (3).	3

Table 6: Function List

2.3. Actor Characteristics

Actor	Description
General User	The user should be knowledgeable of ship design and wave dynamics. The user will use the system on a regular basis and might ask the system to process long unattended computation (such as overnight simulations).
Customer	The customer is knowledgeable of ship design and ship motion simulations. The customer provides the scientific framework needed to develop the software.
Team Leader	The team leader is knowledgeable in all aspects of software development and software engineering. He/She coordinates the activities of the development team and ensures communication with the users, including requirement elicitation, validates all documents and code.
Software Architect	The Software Architect is specialized in software design methodologies and provides a framework and guidelines for the developers.
GUI Designer	The GUI Designer defines an easy to use, easy to learn and difficult to make mistakes GUI.
Software Developer	The Software Developer implements the software. He/She is also responsible for testing.
Technical Writer	The Technical Writer is an expert in writing and programming topics. He/She formalizes reports and provides complete documentation for the project.

Table 7: Actors

2.4. Constraints

The system must be developed using Object-Oriented strategies, offer a full-featured, easy to use GUI, featuring MFC and OpenGL.
All code must be written as to enhance reusability.

2.5. Assumptions and Dependencies

The system will be developed and run under Windows® 2000®.

We assume that the users will be knowledgeable of shipbuilding and of rotational mechanics, which implies there is no need for a help module.

We assume the user will input the variables using the specified measurement units. As such, we do not need to implement identification/conversion algorithms for the measurement units.

We also assume a processing and display that will be faster than the user-required simulation time. As such, some variables will have minimal values to enforce this capability. Such minimal values will be determined after performance evaluation on the minimal supported hardware.

3. Specific Requirements

Priority	Detail
1	Must have
2	Should have
3	Nice to have
4	Unprioritary
removed	Removed from requirements

Table 8: Priority Index

3.1. Functional Requirements

ID	1	
Name		Mathematical Engine
Description		Offers an object-oriented implementation of approximation algorithms for solving ODE. The Engine stops computing when the angle reaches θ_{max}.
Priority		1
Status		Detailed Description, Incomplete Parts
Actors		Software Developer, Team Leader
Pre-Conditions		All parameters must've been set prior to run the Mathematical Engine
Inputs		Simulation Parameters, Ship Parameters, Environment Parameters, Algorithm specification
Flow of Events		
Basic Path		1. Solver is created 2. Solver is initialized with initial conditions, reference to buffers and function for F and Stop condition 3. Solver solves ODE and fill buffers with data
Alternative Paths		none
Post-Conditions		The internal data structure holding the data is full with the computed values
Constraint(s)		• Result of computation must be generated in less than user-defined Δt. • Must assure measurement unit conversion.
Related Screenshot		none
Source(s)		Dr. Allievi, Customer François-Michel Brière

ID	**2**	
Name		Data Bridge
Description		Offers a gateway for the data to be passed from the Mathematical Engine components of the system.
Priority		**Removed**
Status		The feature was integrated inside the Mathematical Engine as a member function of Simulation class.
Actors		Software Developer, Team Leader
Pre-Conditions		The data must be currently crunching inside the Mathematical Engine.
Inputs		Data from the Mathematical Engine
Flow of Events		
Basic Path		1. The Mathematical Engine generates data at its own rhythm and begins to fill the internal data structure. 2. The consumer takes data away from the data structure and effectively consumes it, at its own rhythm, concurrently with the data producing.
Alternative Paths		none
Post-Conditions		none
Outputs		Array holding data elements fetched from memory.
Contraint(s)		• Must allow concurrent access as per the producer/consumer problem
Related Screenshot		none
Source(s)		Marc-André Laverdière, Technical Writer

ID	**3**	
Name		2D Data Plotting of Roll Angle
Description		Displays the graph of the roll angle vs. time.
Priority		1
Status		Coded
Actors		Software Developer, GUI Designer, Team Leader
Pre-Conditions		The Mathematical Engine must have processed its data.
Inputs		Data from the Mathematical Engine
Flow of Events		
Basic Path		1. Mathematical Engine is initialized and launched. 2. Data is obtained and sent for the object to draw a 3D plot of the data in OpenGL, from (6). 3. A 2D projection of the graph is displayed to the user, representing the roll angle vs. time graph.
Alternative Paths		If there is an error in the Mathematical Engine, then the process stops after step 1 above
Post-Conditions		none
Outputs		Graphical output
Constraint(s)		• Result of computation must be generated in less than user-defined Δt.
Related Screenshot		Screenshot 2
Source(s)		Dr. Allievi, Customer Frédéric Rioux, Software Architect

ID	**4**
Name	2D Data Plotting of Roll Speed
Description	Displays the graph of the roll speed vs. time.
Priority	1
Status	Coded
Actors	Software Developer, GUI Designer, Team Leader
Pre-Conditions	The Mathematical Engine must have processed its data.
Inputs	Data from the Mathematical Engine
Flow of Events	
Basic Path	1. Mathematical Engine is initialized and launched. 2. Data is obtained and sent for the object to draw a 3D plot of the data in OpenGL, from (6). 3. A 2D projection of the graph is displayed to the user, representing the roll angle vs. time graph.
Alternative Paths	If there is an error in the Mathematical Engine, then the process stops after step 1 above
Post-Conditions	none
Outputs	Graphical output
Constraint(s)	• Result of computation must be generated in less than user-defined Δt.
Related Screenshot	Screenshot 2
Source(s)	Dr. Allievi, Customer Frédéric Rioux, Software Architect

ID	**5**
Name	Integrated Interactive GUI
Description	Allows setting of parameters, run simulations and visualize results without needing to exit and restart the program. The main window is divided as described in §3.2.1 This module is also responsible to warn the user of any error in parameters and/or general errors.
Priority	1
Status	Coded
Actors	Software Developer, GUI Designer, Team Leader
Pre-Conditions	The system is ready to receive user input
Inputs	none
Flow of Events	
Basic Path	1. The user inputs the parameters trough the Parameter Input Panel, or by loading a file, as defined by (9). 2. The user launches the calculations by clicking on a button 3. The System displays the graphs specified by the user.
Alternative Paths	none
Post-Conditions	The system is ready to receive user input
Outputs	none
Related Screenshot	Screenshot 2
Source(s)	Dr. Allievi, Customer Yann McCready, GUI Designer & Software Programmer François-Michel Brière, Team Leader

ID	**6**
Name	3D Data Plotting of Roll Speed and Roll Angle
Description	Displays a graph of roll angle • roll speed • time.
Priority	2
Status	Coded
Actors	Software Developer, Team Leader
Pre-Conditions	The Mathematical Engine must have processed its data.
Inputs	Data from the Mathematical Engine
Flow of Events	
Basic Path	1. Mathematical Engine is initialized and launched. 2. Data is obtained and sent for the object to draw a 3D plot of the data in OpenGL.
Alternative Paths	If there is an error in the Mathematical Engine, then the process stops after step 1 above
Post-Conditions	none
Outputs	Graphical output
Contraint(s)	• Result of computation must be generated in less than user-defined Δt.
Related Screenshot	Screenshot 2, Screenshot 2
Source(s)	Dr. Allievi, Customer Frédéric Rioux, Software Architect

ID	**6.5**
Name	2D Data Plotting of Comparison
Description	Displays the graph of the roll speed vs. roll angle
Priority	3
Status	Coded
Actors	Software Developer, GUI Designer, Team Leader
Pre-Conditions	The Mathematical Engine must have processed its data.
Inputs	Data from the Mathematical Engine
Flow of Events	
Basic Path	1. Mathematical Engine is initialized and launched. 2. Data is obtained and sent for the object to draw a 3D plot of the data in OpenGL, from (6). 3. A 2D projection of the graph is displayed to the user, representing the roll angle vs. time graph.
Alternative Paths	If there is an error in the Mathematical Engine, then the process stops after step 1 above
Post-Conditions	none
Outputs	Graphical output
Constraint(s)	• Result of computation must be generated in less than user-defined Δt.
Related Screenshot	Screenshot 2
Source(s)	Dr. Allievi, Customer Frédéric Rioux, Software Architect

ID	**7**
Name	3D Data Rendering of Ship's Crossection
Description	Displays an animation of the motion of the crossection of the ship trough the time of the simulation.
Priority	2
Status	Coded
Actors	Software Developer, GUI Designer, Team Leader
Pre-Conditions	The Mathematical Engine must have processed its data.
Inputs	Data from the Mathematical Engine
Flow of Events	
Basic Path	1. Mathematical Engine is initialized and launched. 2. Data is obtained and sent for the object to draw an animation of the crossection in OpenGL.
Alternative Paths	If there is an error in the Mathematical Engine, then the process stops after step 1 above
Post-Conditions	The system will be ready
Outputs	Graphical output
Contraint(s)	none
Related Screenshot	Screenshot 2, Screenshot 2
Source(s)	Dr. Allievi, Customer

ID	**7.5**
Name	Display of Graphs
Description	Displays a graph selected by the user in full screen mode.
Priority	removed
Status	High Level Description
Actors	Software Developer, GUI Designer, Team Leader
Pre-Conditions	Result plots must be drawn in (5)
Inputs	User selection of graph to zoom in. Data from the Mathematical Engine
Flow of Events	
Basic Path	1. Data is obtained from the Mathematical Engine and the graph is redrawn on a wider resolution
Alternative Paths	none
Post-Conditions	
Outputs	Graphical output
Contraint(s)	none
Related Screenshot	none
Source(s)	Marc-André Laverdière, Technical Writer

ID	**8**	
Name		Playback Controls
Description		Define controls for playback of the animation sequence yielded by (7). Playback will highlight the progression of the simulation on (7), (6), (4) and (3).
Priority		3
Status		Coded
Actors		Software Developer, GUI Designer, Team Leader
Pre-Conditions		Result plots must be drawn in (7)
Inputs		User selection of graph to zoom in. Data from the Mathematical Engine
Flow of Events		
Basic Path		1. User presses the required button generating a replay 2. Data is fetched from the Mathematical Engine 3. (3), (4),(6) are modified to highlight the current position on the graph. 4. (7) is modified for the current roll angle at the current time only. 5. Current time, position is incremented and steps 2 to 5 are repeated until the end is reached. 6. The current position markers from (3), (4), (6) are removed from these graphs.
Alternative Paths		The user can "pause" the playback, which interrupts temporarily the sequence above. The user might stop the playback. If this is the case, the process returns waiting for step 1 above to become true.
Post-Conditions		
Outputs		Graphical output
Contraint(s)		none
Related Screenshot		Screenshot 2
Source(s)		Marc-André Laverdière, Technical Writer Yann McCready, GUI Designer & Software Programmer

3.2. External Interface Requirements

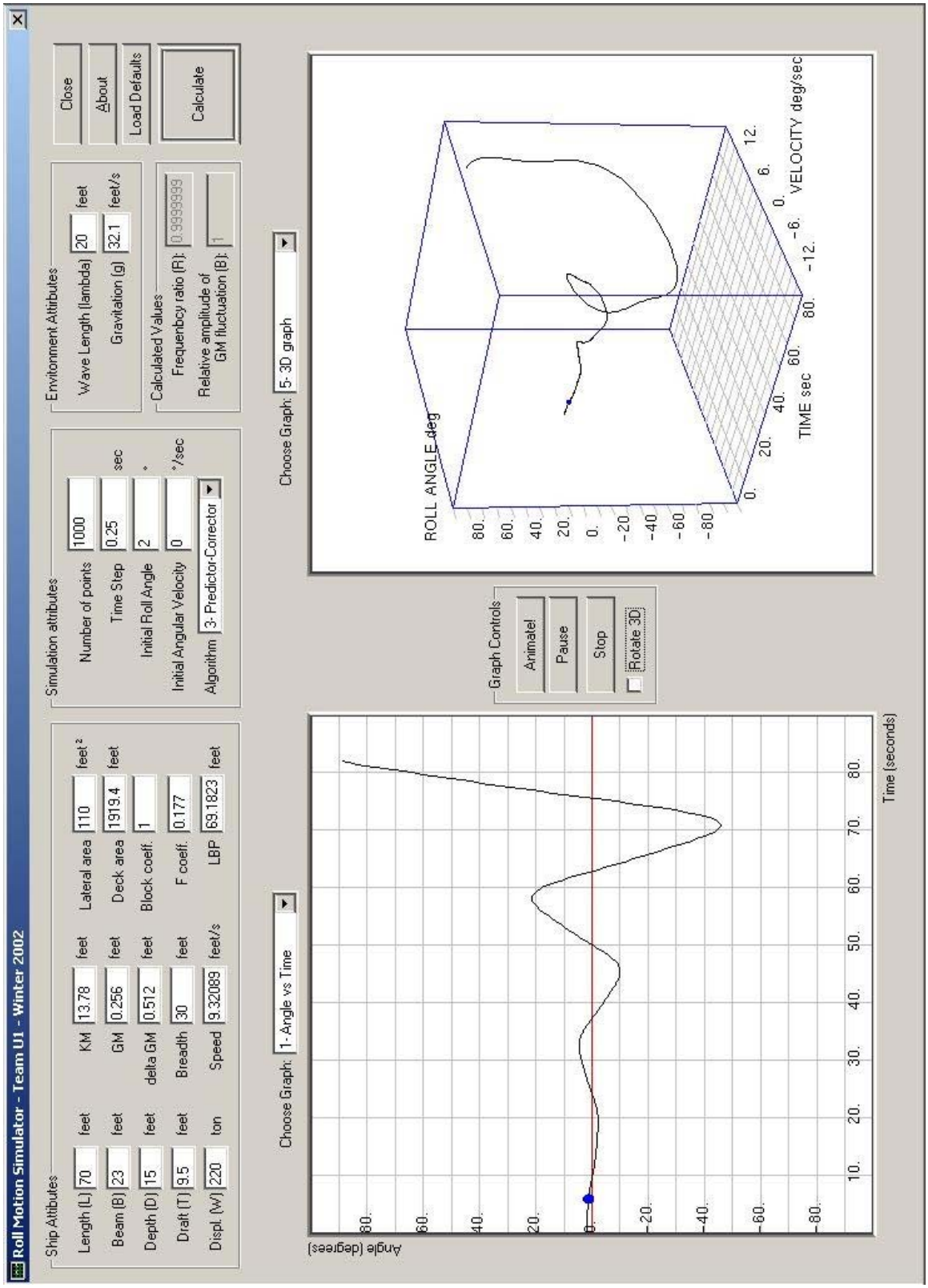
This section describes the interface for the SOS Roll Simulation, in terms of user, hardware, software and communication.

3.2.1. User Interface

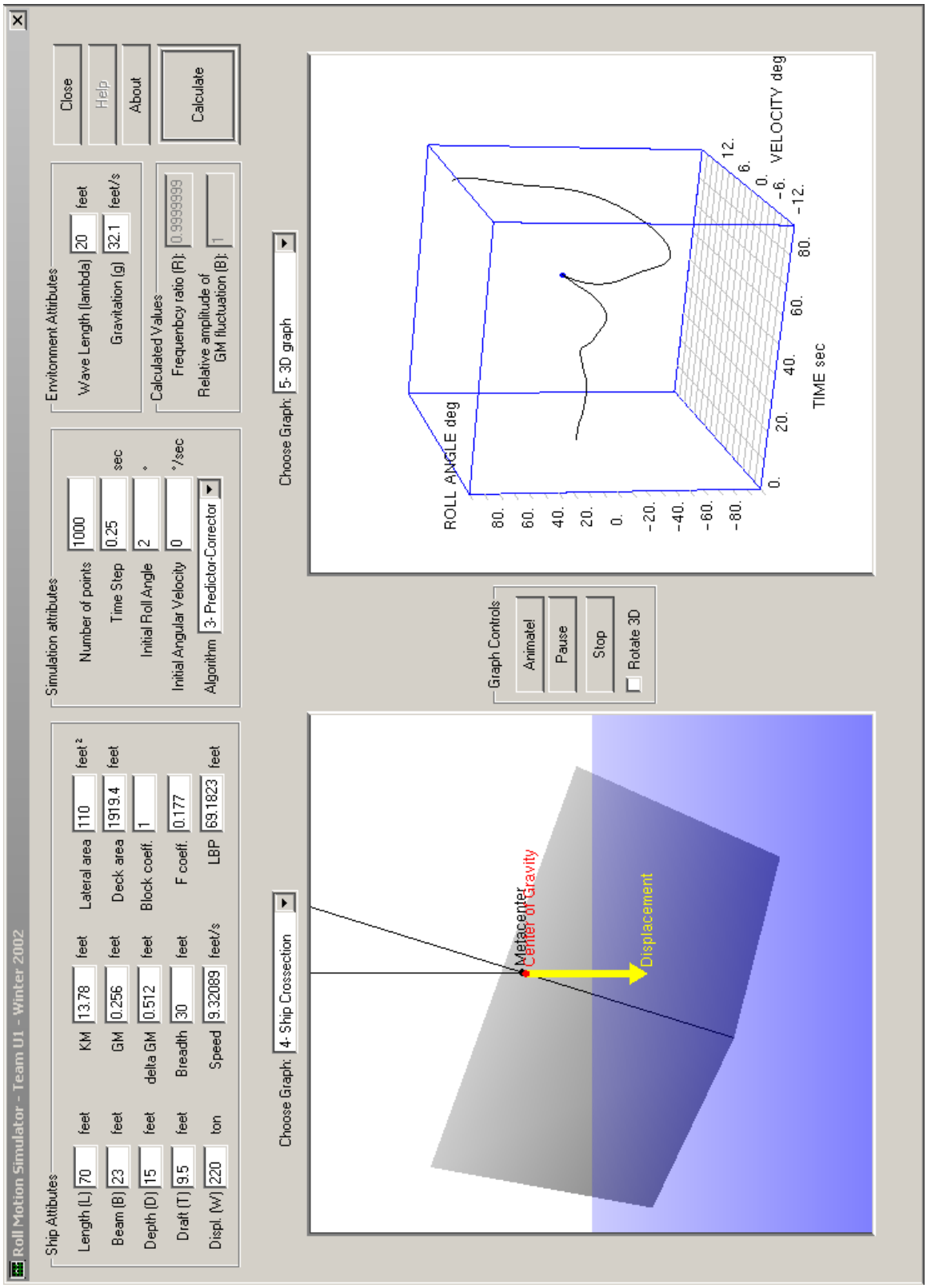
The system will be in a single graphical window, developed in order to facilitate the simulation definition, calculation and plotting.

The upper part of the main window consists of the *Parameter Definition Interface* (PDI). The PDI will offer buttons and textboxes to set parameters, as well as the required buttons to redraw the graphs.

Two plotting areas will occupy the rest of the screen. Each plotting area will have a series of dropdown lists over it that will allow the user to select which graph to visualize. The possible visualization will include the 2D graph of roll angle, 2D graph of roll speed, 2D graph of roll speed vs. angle, 3D graph and crossection rendering.
Playback buttons, allowing redrawing the animation of the movement, will be separating the plotting areas. Screenshot 2 shows the basic layout of the GUI. Screenshot 1 and Screenshot 2 below highlight this arrangement.



Screenshot 1: Graphical User Interface with Angle Vs. Time & 3D Rendering



Screenshot 2: 3D & Crossection Rendering, Showing Animation

3.2.2. Hardware Interfaces

No Hardware Interfaces will be needed for this project.

3.2.3. Software Interfaces

No Software Interfaces will be implemented for this project.

3.2.4. Communication Interfaces

No Communication Interfaces will be implemented for this project.

3.3. Performance Requirements

The simulator must be able to compute and display the simulation results in less time than simulation time specified by the user.

3.4. Design Constraints

The design must take in consideration Object-Oriented implementation.

3.5. Logical Database Requirements

No database will be used to implement this simulator.

3.6. Software System Attributes

3.6.1 Reliability

A reliability level of 10^{-4} will be met.

3.6.2 Availability

Version 1.0 of the system will be available by March 27th 2002 or sooner. Our development strategies and experience in the fields related to GUIs allows us to guarantee the basic set of features to be delivered on time.

3.6.3 Security

No security features will be implemented.

3.6.4 Maintainability

The system's maintenance will be facilitated by our OOD, which allows internal code modification transparently.

3.6.5 Transferability/Portability

The system is to be run only on Windows machines supporting MFC and OpenGL. The minimum system requirements are of a Pentium 200 MHz or better, with at least 128 Mb of RAM, running Windows® 2000® or better, supporting a screen resolution of 1024 x 768 pixels or better.

3.6.6 Learnability

The user should be taught the system in less than 15 minutes from an experienced user. Self-learning the system should take less than 30 minutes hours.

3.7. Other Requirements

The development team must provide the customer three prototypes during the course of the project.

Prototype #	Product Version (as per Table 1)	Due Date
1	0.1	January 30th, 2002
2	0.2	February 14th, 2002
3	0.3	February 28th, 2002
Final Release	1.0+	March 27th, 2002

Table 9: Release Timeline

Part II

SOS Roll Simulator

(Project Management Information)

4. Introduction

This part defines the basic project management relevant information on the SOS Project of Team U1.

5. Time Estimation

The project should be completed in three months on a part-time schedule.

	Task	Duration	Start Date	End Date	Cost	Time
1	Prototype 1	0 days	Wed 30/01/02	Wed 30/01/02	$0.00	0h
2	Prototype 2	0 days	Fri 15/02/02	Fri 15/02/02	$0.00	0h
3	Prototype 3	0 days	Fri 01/03/02	Fri 01/03/02	$0.00	0h
4	**Preliminary Work**	5 days	Wed 16/01/02	Sun 20/01/02	$7,392.00	616h
5	Preliminary Problem Analysis	1 day	Wed 16/01/02	Wed 16/01/02	$576.00	48h
6	Determination of Software Process	2 days	Wed 16/01/02	Thu 17/01/02	$1,152.00	96h
7	Preliminary Math Analysis	3 days	Wed 16/01/02	Fri 18/01/02	$1,152.00	96h
8	Determination of Language used	1 day	Wed 16/01/02	Wed 16/01/02	$576.00	48h
9	Formation of Work Teams	3 days	Wed 16/01/02	Fri 18/01/02	$576.00	48h
10	MFC Research	5 days	Wed 16/01/02	Sun 20/01/02	$480.00	40h
11	**Software Process**	69.1 days	Thu 17/01/02	Wed 27/03/02	$17,568.00	1,576h
12	**Capturing the Requirements**	46 days	Thu 17/01/02	Sun 03/03/02	$3,072.00	256h
13	**Requirements Elicitation**	3 days	Thu 17/01/02	Sat 19/01/02	$768.00	64h
14	Information gathering on boats	2 days	Thu 17/01/02	Fri 18/01/02	$192.00	16h
15	Information Gathering on Mathieu Equation	3 days	Thu 17/01/02	Sat 19/01/02	$288.00	24h
16	Information Gathering on Euler Explicit formula	1 day	Thu 17/01/02	Thu 17/01/02	$288.00	24h
17	**Requirement Definition and Specification documents**	33 days	Mon 21/01/02	Fri 22/02/02	$2,112.00	176h
18	IEEE Template received	1 day	Mon 21/01/02	Mon 21/01/02	$0.00	0h
19	Prototype 1 Draft	5 days	Thu 24/01/02	Mon 28/01/02	$480.00	40h
20	Revision of Draft 1	1 day	Tue 29/01/02	Tue 29/01/02	$96.00	8h
21	Prototype 2 Draft	5 days	Thu 31/01/02	Mon 04/02/02	$480.00	40h
22	Revision of Draft 2	2 days	Tue 05/02/02	Wed 06/02/02	$192.00	16h

23	Prototype 3 Draft	5 days	Fri 15/02/02	Tue 19/02/02	$480.00	40h
24	Revision of Draft 3	2 days	Wed 20/02/02	Thu 21/02/02	$192.00	16h
25	Final Draft	1 day	Fri 22/02/02	Fri 22/02/02	$192.00	16h
26	Requirements Verification	0.33 days	Sat 02/03/02	Sat 02/03/02	$192.00	16h
27	Requirement Validation	1 day	Sun 03/03/02	Sun 03/03/02	$0.00	0h
28	**System Design**	39 days	Thu 17/01/02	Sun 24/02/02	$2,304.00	192h
29	Determination of Deign Method	1 day	Thu 17/01/02	Thu 17/01/02	$96.00	8h
30	Overall Architecture	4 days	Fri 18/01/02	Mon 21/01/02	$384.00	32h
31	Modularization	4 days	Sat 19/01/02	Tue 22/01/02	$384.00	32h
32	Communication methods between modules	2 days	Sun 20/01/02	Mon 21/01/02	$384.00	32h
33	System Design Approval	1 day	Tue 22/01/02	Tue 22/01/02	$96.00	8h
34	System Design Documentation	10 days	Fri 15/02/02	Sun 24/02/02	$960.00	80h
35	**Program Design**	28 days	Wed 23/01/02	Tue 19/02/02	$3,456.00	288h
36	**Math Engine Design**	28 days	Wed 23/01/02	Tue 19/02/02	$1,248.00	104h
37	Algorithm 1 design	2 days	Wed 23/01/02	Thu 24/01/02	$192.00	16h
38	Algorithm 2 design	1 day	Thu 24/01/02	Thu 24/01/02	$96.00	8h
39	Algorithm 3 design	5 days	Fri 15/02/02	Tue 19/02/02	$960.00	80h
40	**GUI Design**	7 days	Thu 31/01/02	Wed 06/02/02	$768.00	64h
41	Preliminary sketch	2 days	Thu 31/01/02	Fri 01/02/02	$192.00	16h
42	User friendliness	3 days	Mon 04/02/02	Wed 06/02/02	$576.00	48h
43	**Plotting/Rendering design**	5 days	Thu 31/01/02	Mon 04/02/02	$1,440.00	120h
44	OpenGL research	5 days	Thu 31/01/02	Mon 04/02/02	$1,440.00	120h
45	**Implementation**	56 days	Sat 19/01/02	Fri 15/03/02	$5,760.00	480h
46	**Object Class Implementation**	2 days	Tue 22/01/02	Wed 23/01/02	$192.00	16h
47	Ship Class Implementation	1 day	Tue 22/01/02	Tue 22/01/02	$96.00	8h
48	Boat Class Implementation	1 day	Wed 23/01/02	Wed 23/01/02	$96.00	8h
49	**Math Engine Implementation**	31 days	Fri 25/01/02	Sun 24/02/02	$1,632.00	136h
50	Algorithm 1 Implementation	5 days	Fri 25/01/02	Tue 29/01/02	$480.00	40h
51	Algorithm 2 Implementation	2 days	Sat 26/01/02	Sun 27/01/02	$192.00	16h
52	Algorithm 3 Implementation	5 days	Wed 20/02/02	Sun 24/02/02	$960.00	80h
53	**Plotting/Rendering**	16.33 days	Sat 09/02/02	Mon 25/02/02	$1,824.00	152h

	Implementation					
54	3D plot	3.33 days	Fri 22/02/02	Mon 25/02/02	$960.00	80h
55	2D projection for theta	2 days	Sat 09/02/02	Sun 10/02/02	$192.00	16h
56	2D projection for theta prime	2 days	Sat 09/02/02	Sun 10/02/02	$192.00	16h
57	Crossection of the ship	5 days	Tue 19/02/02	Sat 23/02/02	$480.00	40h
58	**GUI Implementation**	40 days	Sat 19/01/02	Wed 27/02/02	$1,248.00	104h
59	Input screen	3 days	Sat 19/01/02	Mon 21/01/02	$288.00	24h
60	MFC/OpenGL integration	3 days	Wed 06/02/02	Fri 08/02/02	$288.00	24h
61	Animation	2 days	Wed 20/02/02	Thu 21/02/02	$192.00	16h
62	Playback controls	5 days	Sat 23/02/02	Wed 27/02/02	$480.00	40h
63	Implementation Document	3 days	Wed 13/03/02	Fri 15/03/02	$864.00	72h
64	**Testing the Program**	14 days	Wed 30/01/02	Tue 12/02/02	$1,344.00	224h
65	Develop Math test	2 days	Wed 30/01/02	Thu 31/01/02	$576.00	48h
66	Test math section	2 days	Fri 01/02/02	Sat 02/02/02	$192.00	16h
67	Develop GUI test	2 days	Fri 08/02/02	Sat 09/02/02	$384.00	32h
68	Test GUI	2 days	Mon 11/02/02	Tue 12/02/02	$192.00	16h
69	**Testing the System**	17 days	Sat 02/03/02	Mon 18/03/02	$1,344.00	112h
70	Develop module communication test	1 day	Sat 02/03/02	Sat 02/03/02	$192.00	16h
71	Test module communication	2 days	Fri 08/03/02	Sat 09/03/02	$384.00	32h
72	Test Documentation	0.8 days	Mon 11/03/02	Mon 11/03/02	$384.00	32h
73	Design Inspection	2 days	Wed 13/03/02	Thu 14/03/02	$192.00	16h
74	Code Inspection	2 days	Sun 17/03/02	Mon 18/03/02	$192.00	16h
75	**Delivery and Maintenance**	57.1 days	Tue 29/01/02	Wed 27/03/02	$288.00	24h
76	Packaging and Delivery Proto 1	0.1 days	Tue 29/01/02	Tue 29/01/02	$24.00	2h
77	Packaging and Delivery Proto 2	0.1 days	Fri 15/02/02	Fri 15/02/02	$24.00	2h
78	Packaging and Delivery Proto 3	0.1 days	Fri 01/03/02	Fri 01/03/02	$24.00	2h
79	Packaging and Delivery FINAL	0.1 days	Wed 27/03/02	Wed 27/03/02	$24.00	2h
80	Code Maintenance - Change Management	3 days	Sat 02/03/02	Mon 04/03/02	$288.00	24h

Table 10: Time and Working Cost Analysis

6. Cost Estimation

Software	Licenses	Total Price (Can$)
Microsoft Office 2000 (Academic license)	6	1500
Microsoft Visual Studio 6	5	20000
Microsoft Project 2000	2	1600
Microsoft SourceSafe 6	2	1800
Microsoft Visio 2002	2	650
FTP Server	1	0 (offered by customer)
Adobe Acrobat 5	1	400
Adobe Photoshop 6	1	1000
Bryce 5	1	500
Total		27450

Table 11: Software Costs

Equipment	Units	Total Price (Can$)
Development Workstation	6	7300
Demonstration Laptops	2	5000
PDAs	4	2000
Internet Connections	3 months	250
Printers	2	1000
Furniture	6 (desk, chairs, lamps, etc.)	5000
Office Supplies	various	500
Facilities (4 ½)	3 months	1500
Electricity	3 months	300
Transportation fees	200 km	100
Total		22950

Table 12: Other Costs Estimation

Budget Item	Estimation
Software	27450
Equipment & Facilities	22950
Staff	25000
Total	75400

Table 13: Cumulative Cost Estimation

7. Team Members and Assignments

Member	Task(s)
François-Michel Brière	Team Leader
Negar Family	Software Developer
Marc-André Laverdière	Technical Writer
Yann McCready	GUI Designer, Software Developer
Frédéric Rioux	Software Architect, Software Developer
Jia-Wei Zhang	Software Developer

Table 14: Team Members & Tasks

Resources and Assignments
Frédéric Rioux
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Determination of Language used
Formation of Work Teams
Information Gathering on Euleur Explicit formula
Requirements Verification
Determination of Deign Method
Overall Architecture
Modularization
Communication methods between modules
System Design Documentation
OpenGL research
Develop GUI test
Test GUI
Test module communication
Test Documentation
Code Inspection
Packaging and Delivery Proto 1
Packaging and Delivery Proto 2
Packaging and Delivery Proto 3
Packaging and Delivery FINAL
Yann McCready
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Determination of Language used
MFC Research
Information Gathering on Mathieu Equation
Information Gathering on Euleur Explicit formula
Requirements Verification
Communication methods between modules
User friendliness
OpenGL research
3D plot
Input screen
MFC/OpenGL integration
Animation
Playback controls
Implementation Document
Develop module communication test
Test module communication
Test Documentation

Design Inspection
Negar Family
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Preliminary Math Analysis
Determination of Language used
Requirements Verification
Algorithm 1 design
Algorithm 2 design
Algorithm 3 design
Algorithm 1 Implementation
Algorithm 2 Implementation
Algorithm 3 Implementation
Implementation Document
Develop module communication test
Test Documentation
Code Maintenance - Change Management
Jia-Wei Zhang
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Preliminary Math Analysis
Determination of Language used
Information Gathering on Euleur Explicit formula
Requirements Verification
Algorithm 3 design
Ship Class Implementation
Boat Class Implementation
Algorithm 3 Implementation
3D plot
Implementation Document
Develop Math test
Test math section
Test Documentation
Francois-Michel Briere
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Preliminary Math Analysis
Determination of Language used
Formation of Work Teams
Information gathering on boats
Revision of Draft 1
Revision of Draft 2

Revision of Draft 3
Final Draft
Requirements Verification
System Design Approval
Preliminary sketch
User friendliness
OpenGL research
3D plot
2D projection for theta
2D projection for theta prime
Crossection of the ship
Develop Math test
Develop GUI test
Marc-Andre Laverdière
Preliminary Work
Preliminary Problem Analysis
Determination of Software Process
Preliminary Math Analysis
Determination of Language used
Prototype 1 Draft
Prototype 2 Draft
Prototype 3 Draft
Final Draft
Requirements Verification
Develop Math test
Test Documentation

Table 15: Task Allocation & Follow-Up

8. Task Organization – Gant Chart

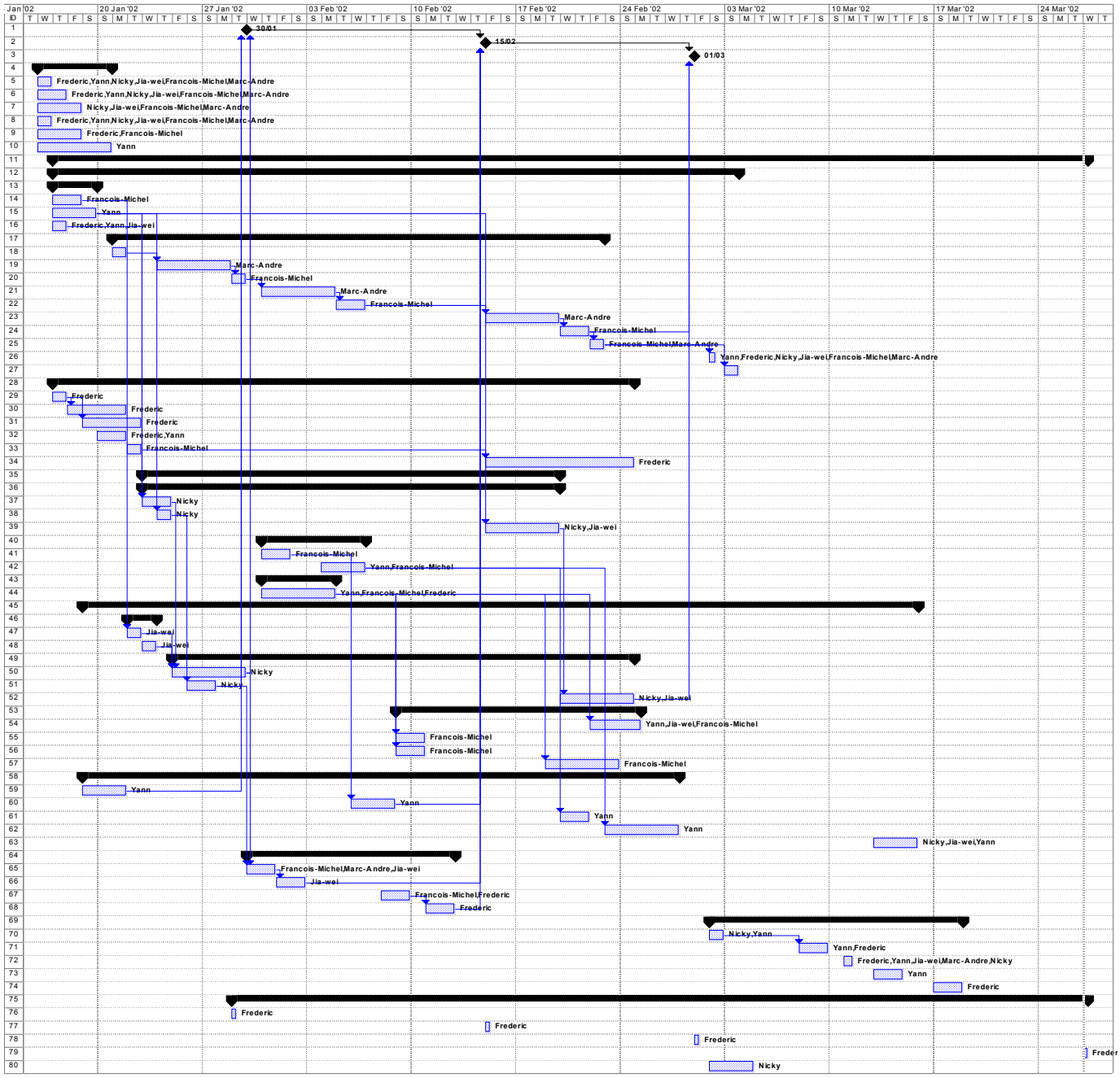


Figure 2: Gantt Chart for Tasks

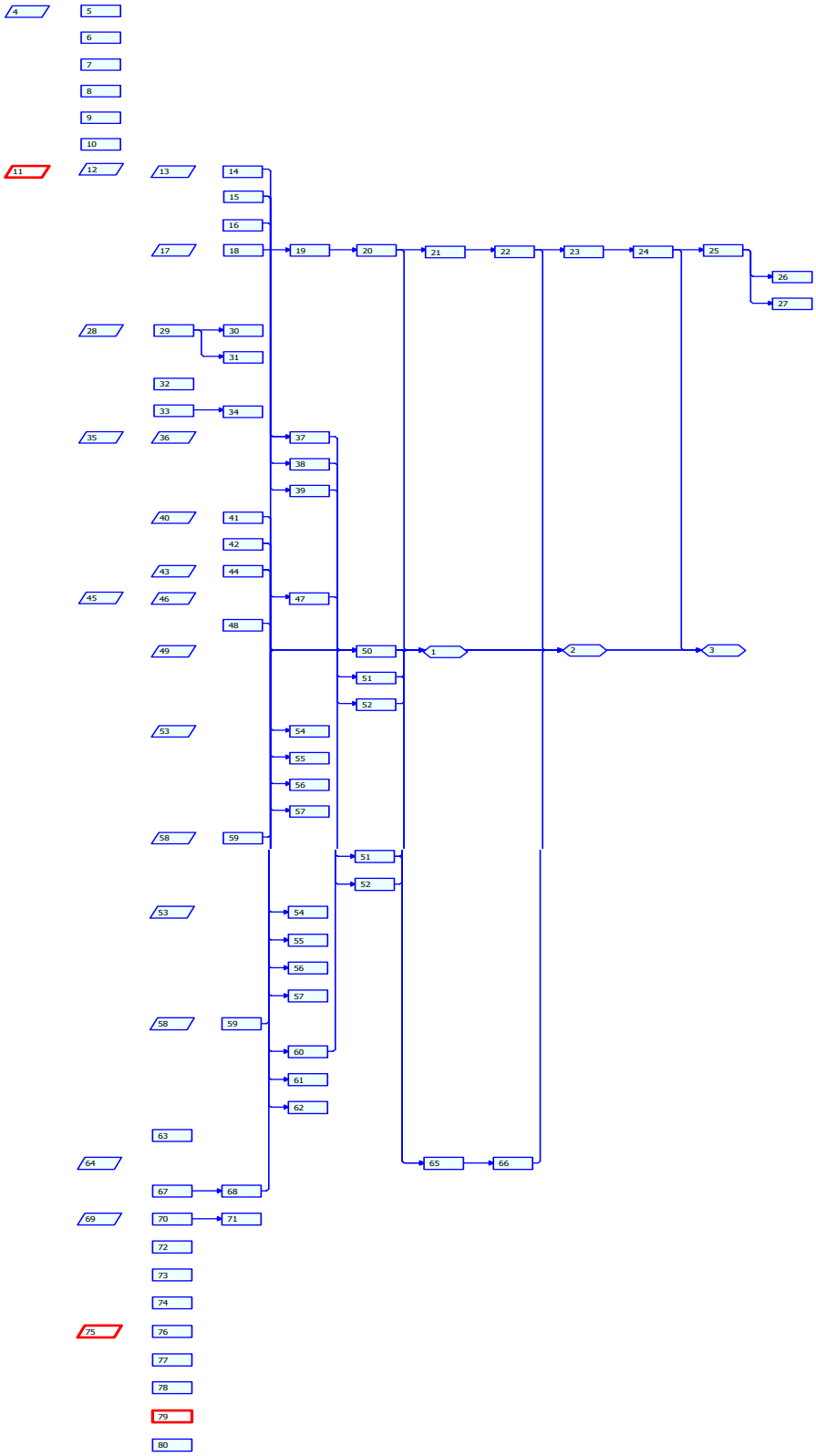


Figure 3: Activity Graph

9. Development Methodology

The development team will use the Phased Development Strategy with prototyping in order to develop the software.

An Incremental Strategy will be applied for the development of components in the GUI and the Iterative approach will be considered for the internal components of the Mathematical Engine.

Part III

SOS Roll Simulator

(Software Design Document)

10. Introduction

This part defines the system design for the SOS project. The system design document offers a general view of the implementation strategy for the project in non-technical terms.

The design of the system is based on assumptions by the development team related to the expected usage of the software.

11. Feature Implementation Concepts

ID	**1**	
Name		Mathematical Engine
Description		Offers an object-oriented implementation of approximation algorithms for solving ODE. The Engine stops computing when the angle reaches θ_{max}.
Algorithm(s)		**General Mathematical Remarks** Three different algorithms will be implemented *1.* Explicit Euler Scheme **Angle**: $\theta_{n+1} = \theta_n + \Delta t \dot{\theta}_n$ (1) **Angular Velocity**: $\dot{\theta}_{n+1} = \dot{\theta}_n + \Delta t \ddot{\theta}_n$ (2) *2.* Crank-Nicholson Scheme **Angle**: $\theta_{n+1} = \theta_n + \Delta t \dot{\theta}_{n+\frac{1}{2}} = \theta_n + \Delta t \frac{1}{2}(3\dot{\theta}_n - \dot{\theta}_{n+1})$ (3) We use the following approximation to compute the angle: $3\dot{\theta}_n - \dot{\theta}_{n+1} = \dot{\theta}_n + \dot{\theta}_n + \dot{\theta}_n - \dot{\theta}_{n+1} = \dot{\theta}_n + \dot{\theta}_n + \Delta\dot{\theta} \cong \dot{\theta}_n + \dot{\theta}_{n+1}$ **Angular Velocity**: $\dot{\theta}_{n+1} = \dot{\theta}_n + \Delta t \ddot{\theta}_{n+\frac{1}{2}} = \dot{\theta}_n + \Delta t \frac{1}{2}(\ddot{\theta}_n + \ddot{\theta}_{n+1})$ (4) *3.* Predictor-Corrector Scheme This case determines a first value (predictor) that is adjusted by another calculation (corrector). **Angle**: $\theta^P_{n+1} = \theta_n + \Delta t[\sum_{K=1}^{M} \beta^P_{MK} \dot{\theta}_{n-K+1}]$ (5) $\theta^C_{n+1} = \theta_n + \Delta t[\sum_{K=0}^{M} \beta^C_{MK} \dot{\theta}_{n-K+1}]$ (6) **Angular Velocity:** $\dot{\theta}^C_{n+1} = \dot{\theta}_n + \Delta t[\sum_{K=0}^{M} \beta^C_{MK} \ddot{\theta}_{n-K+1}]$ (7) We use our already defined θ_{n+1} from the predictor's angle. **Implementation Detail** *4.* Explicit Euler Scheme Initial Data: Δt , θ_0 , $\dot{\theta}_0$, t_0

Step 1: find θ_1 with equation 1.

Step 2: find $\dot{\theta}_1$ with equation 2. Here is its expansion:

$$\dot{\theta}_1 = \dot{\theta}_0 + \Delta t(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)$$

Step 3: Repeat step 1 and 2 to get more points. Here is its expansion:

Angle:

$$\theta_{n+1} = \theta_n + \Delta t\dot{\theta}_n$$

Angular Velocity:

$$\dot{\theta}_{n+1} = \dot{\theta}_n + \Delta t(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_n))\theta_n)$$

5. Crank-Nicholson Scheme

Initial Data: Δt , θ_0 , $\dot{\theta}_0$, t_0

Step 1: find θ_1 with equation 1.

Step 2: find $\dot{\theta}_1$ with equation 2. Here is its expansion:

$$\dot{\theta}_1 = \dot{\theta}_0 + \Delta t(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)$$

Step 3: find θ_2 with equation 3.

Step 4: find $\dot{\theta}_2$ with equation 4. Here is its expansion:

$$\dot{\theta}_2 = \dot{\theta}_1 + \Delta t(1/2)[(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_1))\theta_1)$$

$$+(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_2))\theta_2)]$$

Step 5: repeat step 3 and 4 to get more points. Here is its expansion:

Angle:

$$\theta_{n+1} = \theta_n + \Delta t\frac{1}{2}(3\dot{\theta}_n - \dot{\theta}_{n+1})$$

Angular Velocity:

$$\dot{\theta}_{n+1} = \dot{\theta}_n + \Delta t(1/2)[(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_n))\theta_n)$$

$$+(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_{n+1}))\theta_{n+1})]$$

6. Predictor-Corrector Scheme

Initial Data: Δt , θ_0 , $\dot{\theta}_0$, t_0 and the table 22, 23

Step 1: find θ_1^P with equation 5. Here is its expansion:

$$\theta_1^P = \theta_0 + \Delta t[(1)\dot{\theta}_0]$$

Step 2: find $\dot{\theta}_1$ with equation 6. Here is its expansion:

$$\dot{\theta}_1^C = \dot{\theta}_0 + \Delta t[(1/2)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_1))\theta_1^P)$$
$$+(1/2)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)]$$

Step 3: find θ_1^C with equation 7. Here is its expansion:

$$\theta_1^C = \theta_0 + \Delta t[(1/2)(\dot{\theta}_1) + (1/2)(\dot{\theta}_0)]$$

Step 4: find θ_2^P with equation 5. Here is its expansion:

$$\theta_2^P = \theta_1 + \Delta t[(3/2)\dot{\theta}_1 + (-1/2)\dot{\theta}_0]$$

Step 5: find $\dot{\theta}_2$ with equation 6. Here is its expansion:

$$\dot{\theta}_2^C = \dot{\theta}_1 + \Delta t[(5/12)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_2))\theta_2^P)$$
$$+(8/12)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_1))\theta_1)$$
$$+(-1/12)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)]$$

Step 6: find θ_2^C with equation 7. Here is its expansion:

$$\theta_2^C = \theta_1 + \Delta t[(5/12)(\dot{\theta}_2) + (8/12)(\dot{\theta}_1) + (-1/12)(\dot{\theta}_0)]$$

Step 7: find θ_3^P with equation 5. Here is its expansion:

$$\theta_3^P = \theta_2 + \Delta t[(23/12)\dot{\theta}_2 + (-16/12)\dot{\theta}_1 + (5/12)\dot{\theta}_0]$$

Step 8: find $\dot{\theta}_3$ with equation 6. Here is its expansion:

$$\dot{\theta}_3^C = \dot{\theta}_2 + \Delta t[(9/24)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_3))\theta_3^P)$$
$$+(19/24)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_2))\theta_2)$$
$$+(-5/24)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_1))\theta_1)$$
$$+(1/24)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)]$$

Step 9: find θ_3^C with equation 7. Here is its expansion:

$$\theta_3^C = \theta_2 + \Delta t[(9/24)(\dot{\theta}_3) + (19/24)(\dot{\theta}_2)$$
$$+(-5/24)(\dot{\theta}_1) + (1/24)(\dot{\theta}_0)]$$

Step 10: find θ_4^P with equation 5. Here is its expansion:

$$\theta_4^P = \theta_3 + \Delta t[(55/24)\dot{\theta}_3 + (-59/24)\dot{\theta}_2 + (37/24)\dot{\theta}_1$$
$$+(-9/24)\dot{\theta}_0]$$

	Step 11: find $\dot{\theta}_4$ with equation 6. Here is its expansion: $\dot{\theta}_4^C = \dot{\theta}_3 + \Delta t[(251/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_4))\theta_4^P)$ $+(646/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_3))\theta_3)$ $+(-264/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_2))\theta_2)$ $+(106/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_1))\theta_1)$ $+(-19/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_0))\theta_0)]$ Step 12: find θ_4^C with equation 7. Here is its expansion: $\theta_4^C = \theta_3 + \Delta t[(251/720)(\dot{\theta}_4)+(646/720)(\dot{\theta}_3)$ $+(-264/720)(\dot{\theta}_2)+(106/720)(\dot{\theta}_1)+(-19/720)(\dot{\theta}_0)]$ Step 13: repeat step 10 to 12 to get more points. Here is its expansion: **Angle Predictor**: $\theta_{n+1}^P = \theta_n + \Delta t[(55/24)\dot{\theta}_n + (-59/24)\dot{\theta}_{n-1} + (37/24)\dot{\theta}_{n-2}$ $+(-9/24)\dot{\theta}_{n-3}]$ **Angular Velocity**: $\dot{\theta}_{n+1}^C = \dot{\theta}_n + \Delta t[(251/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_{n+1}))\theta_{n+1}^P)$ $+(646/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_n))\theta_n)$ $+(-264/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_{n-1}))\theta_{n-1})$ $+(106/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_{n-2}))\theta_{n-2})$ $+(-19/720)(-\omega_0^2(1-(\frac{\Delta GM}{2.GM})\sin(2\omega_e t_{n-3}))\theta_{n-3})]$ **Angle Corrector**: $\theta_{n+1}^C = \theta_n + \Delta t[(251/720)(\dot{\theta}_{n+1})+(646/720)(\dot{\theta}_n)$ $+(-264/720)(\dot{\theta}_{n-1})+(106/720)(\dot{\theta}_{n-2})+(-19/720)(\dot{\theta}_{n-3})]$ Furthermore, the implementation of the algorithms takes into account θ_{max}. If any of these values are reached, the computing is stopped.
Data Structures	List holds the computed values of θ, θ', t

Source(s)	Dr. Allejandro Allievi, customer Frédéric Rioux, Software Architect Jia-We Zhang, Software Developer

ID	**3**
Name	2D Data Plotting of Roll Angle
Description	Displays the graph of the roll angle vs. time.
Algorithms	Computed data from Mathematical Engine is fetched, by pair of points. A vertex is drawn between that pair of points such that a continuous graph results as a 3D plot. Axes are then drawn and numbered. The graph is then projected in 2D as to show the 2D Data Plotting of Roll Angle.
Data Structures	none
Source(s)	Frédéric Rioux, Software Architect

ID	**4**
Name	2D Data Plotting of Roll Speed
Description	Displays the graph of the roll speed vs. time.
Algorithms	Computed data from Mathematical Engine is fetched, by pair of points. A vertex is drawn between that pair of points such that a continuous graph results as a 3D plot. Axes are then drawn and numbered. The graph is then projected in 2D as to show the 2D Data Plotting of Roll Speed.
Data Structures	None
Source(s)	Frédéric Rioux, Software Architect

ID	**5**
Name	Integrated Interactive GUI
Description	Allows the user to set parameters, run simulations and visualize results without needing to exit and restart the program. The main window contains two graphs and the user can select the type of graph in order to be able to compare any two types of graph at the same time. Graph types are: Angle vs. Time, Velocity vs. Time, Velocity vs. Angle, Ship Crossection, Velocity vs. Angle vs. Time (3D). The user sees the parameters, the graphs and all the controls on the same window (for quick access and convenience). The main window is divided as described in §3.2.1 This module is also responsible to warn the user of any error in parameters and/or general errors.
Algorithm	The GUI will validate the user's data for invalid parameters at the moment of the data entry and at the computing request from the user. A message will then notify the user of incorrect parameter(s).
Data Structures	none
Source(s)	Yann McCready, GUI Designer & Software Programmer François-Michel Brière, Team Leader

ID	**6**
Name	3D Data Plotting of Roll Speed and Roll Angle
Description	Displays a graph of roll angle • roll speed • time.
Algorithm	Computed data from Mathematical Engine is fetched, by pair of points. A vertex is drawn between that pair of points such that a continuous graph results as a 3D plot. Axes are then drawn and numbered. The graph is then projected as to optimize the user's view.
Data Structures	none
Source(s)	Frédéric Rioux, Software Architect

ID	**6.5**
Name	2D Data Plotting of Comparison
Description	Displays the graph of the roll speed vs. roll angle
Algorithm	Computed data from Mathematical Engine is fetched, by pair of points. A vertex is drawn between that pair of points such that a continuous graph results as a 3D plot. Axes are then drawn and numbered. The graph is then projected in 2D as to show the 2D Data Plotting of Comparison.
Data Structures	none
Source(s)	Frédéric Rioux, Software Architect

ID	**7**
Name	3D Data Rendering of Ship's Crossection
Description	Displays an animation of the motion of the crossection of the ship trough the time of the simulation.
Algorithm	Display the ship's crossection and its reference axes. Data is fetched from the Mathematical Engine and the viewpoint is rotated based on the angle resulting from the computation. At every incrementation in time, update the angle and display a dot on the other displayed graph, if shown to the user, highlighting the evolution of the simulation.
Data Structures	Time displayed counter and Drawing flag
Source(s)	Yann McCready, GUI Designer & Software Programmer

ID	**8**
Name	Playback Controls
Description	Define controls for playback of the animation sequence yielded by (7). Playback will highlight the progression of the simulation on (7), (6), (4) and (3).
Algorithm	The buttons will modify (7) in its data to reflect the effect of the button pressed.
Data Structures	none
Source(s)	Marc-André Laverdière, Technical Writer Yann McCready, GUI Designer & Software Programmer

12. Conceptual Design

The *Ship On Sea (SOS) Roll Motion Simulator* is a software program with a user interface intended to help ship operator and designer analyse the survivability of a given ship in waves.

The user can set the attributes of the ship to the desired values using the GUI. He or she can also modify the environment in which the ship evolves. More precisely, the following parameters can be modified:

- Ship
  - ≈ Length
  - ≈ Beam
  - ≈ Depth
  - ≈ Draft
  - ≈ Displacement
  - ≈ Height of metacentre
  - ≈ Transverse metacentric height
  - ≈ Fluctuation of the transverse metacentric height due to wave action
  - ≈ Projected Lateral area
  - ≈ Block coefficient
  - ≈ Breadth
  - ≈ Deck area
  - ≈ Length of perpendiculars
  - ≈ Speed

- Environment
  - ≈ Wave Length
  - ≈ Gravitation

- Simulation
  - ≈ Number of points to be calculated
  - ≈ Time step
  - ≈ Initial roll angle
  - ≈ Initial angular velocity
  - ≈ Algorithm to be used (Explicit Euler, Crank-Nicholson, Predictor-Corrector)

Using the given data, the software will calculate the roll motion of the ship and the angular velocity with respect to time using the algorithm the user wants.

For comparison purposes, the user will be able to display two graphs at the time. He or she will choose any two of the possible graphs to be displayed side by side. The available graphs are:

- Roll Angle vs. Time
- Angular Velocity vs. Time
- Angular Velocity vs. Roll Angle
- Cross-section of the ship
- 3D Roll Angle vs. Angular Velocity vs. Time

The above graph will offer the possibility of being animated. Play, pause and stop controls will be provided to the user. The Cross-section graph animation will represent how the ship moves with time. All other graphs will display a marker (small sphere) that will move along the graph curve with time. The 3D graph will have the extra possibility of being rotated around a vertical axis, so that the curve can be viewed from all angles. A control in the GUI lets the user decide to rotate the 3D graph or to leave it still.

The system will be built to follow the sequence defined in Figure 4.

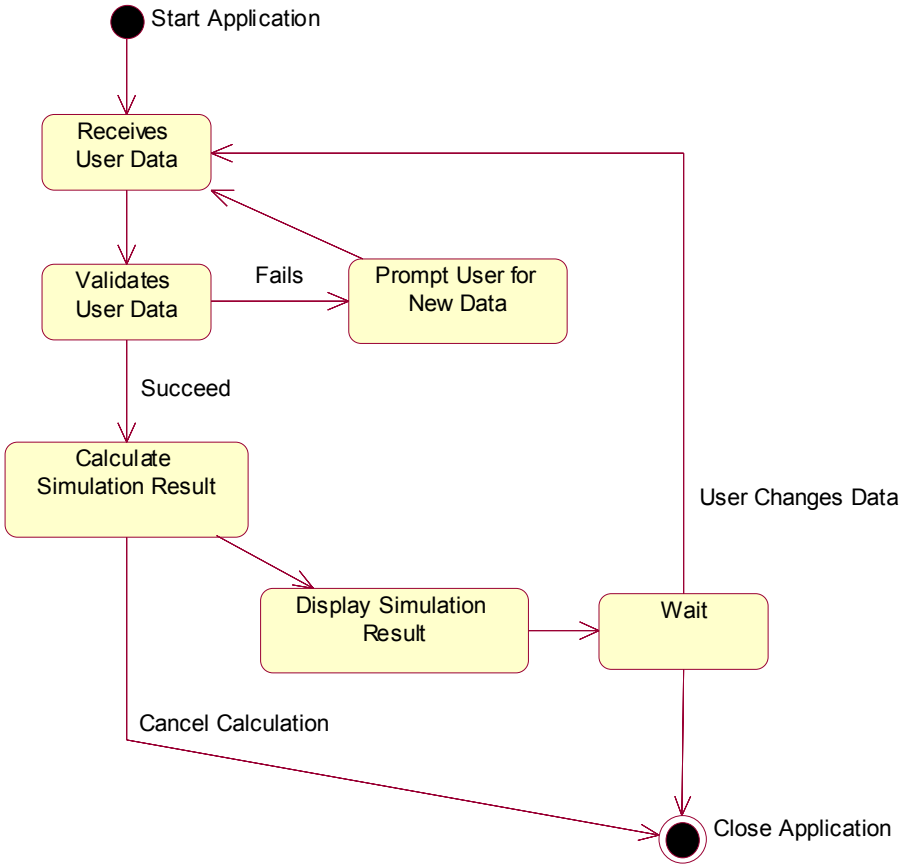


Figure 4: Activity Diagram for System

13. Technical Design

The system will be implemented with low coupling and high cohesion using an Object-Oriented design. It will be modular and will be easily portable.

From a high-level perspective the system can be viewed as two parts. The first part, the "core", is the mathematical engine and the set of classes that are used in the calculation of the points. The second part is the GUI, which takes the user input and output the graphs. This interaction and separation is well detailed in Figure 5.

If ever the system is to be ported, only the GUI will have to be written since this layer is independent from the simulation.

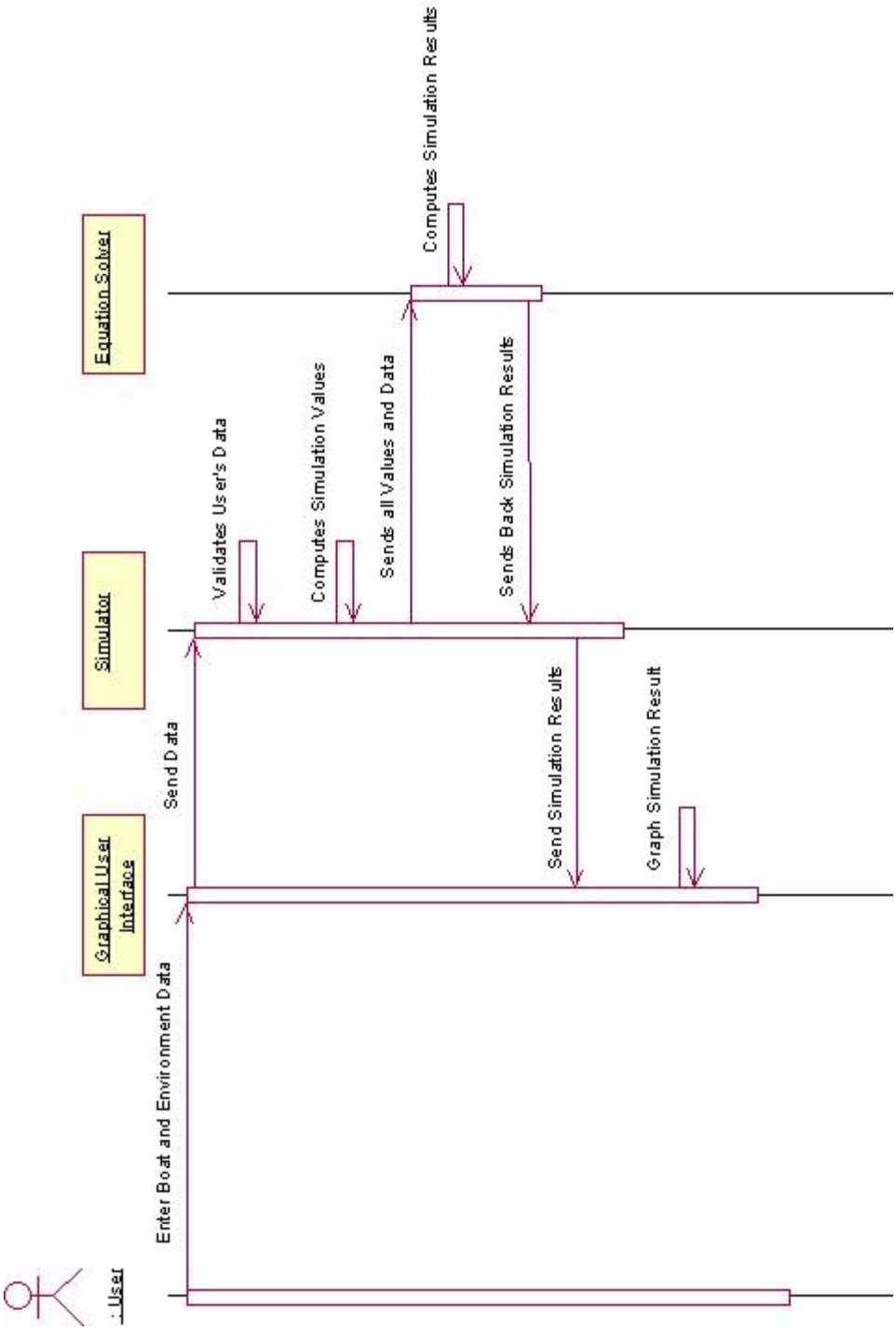


Figure 5: Sequence Diagram for the System

13.1. The Core

The core will be independent from the GUI and will offer a complete API. The core can be further divided into two main parts:

- Mathematical Engine
- Simulation

The mathematical is completely independent of the problem (in our case, the calculation of the roll angle and the angular velocity of a ship in waves over time). It will be completely reusable. The Mathematical Engine is in fact an ODE solver. Currently, three algorithms are supported (Explicit Euler, Crank-Nicholson, Predictor-Corrector).

The implementation of the Mathematical engine will take the form of the EquationSolver abstract class and each algorithm will be implemented as a class derived from EquationSolver.

- EquationSolver
  - (inherit) ExplicitEuler
  - (inherit) CrankNicolson
  - (inherit) PredictorCorrector

The other part of the core, the simulation, will be a class that warps all information needed to do the calculation and will contain buffers to store the results of such calculations.

- RollMotionSimulation
  - (has a) Ship
  - (has a) Environment
  - (uses) ExplicitEuler
  - (uses) CrankNicolson
  - (uses) PredictorCorrector

The architecture of the core can thus be summarized trough an UML Diagram, as shown in Figure 6.

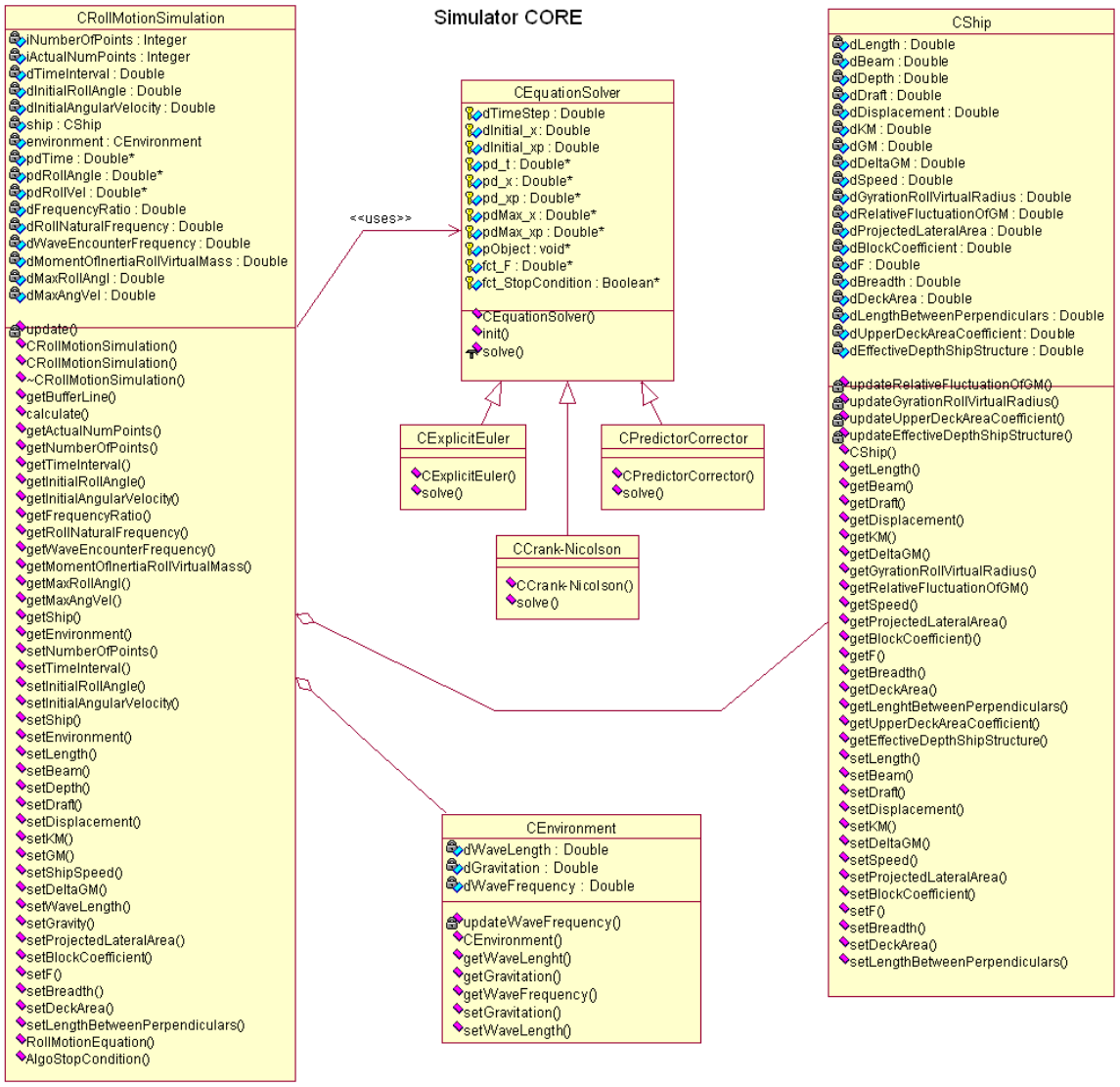


Figure 6: UML Diagram for Mathematical Engine

13.2. The GUI

13.2.1 Introduction

The system's GUI is the agglomeration of two major subparts:

- User Input handling
- Graphical display of results

In order to optimize simplicity and user-friendliness, all subcomponents will be integrated inside a single window.

The user will input all necessary data for the simulation by changing the values in text boxes.

Buttons will give the user control of the simulation: Calculate, Close, About. When the user wants to see the result of the simulation for the

current entered values, he/she will press the Calculate button and the results will be plotted on the two graphs. Pressing the About button will show the splash screen of the program. Pressing the Close button will close the program instantly. Also, one drop-down list control will be provided for each graph so that the user can select the graph type he or she wants in each graph.

13.2.2 Tools

The user input handling part of the program will be implemented using Microsoft Foundation Classes (MFC). MFC has been selected for our system based on multiple criteria.

- **Object Oriented**: MFC offers an object-oriented approach to the implementation of the GUI, which is a non-functional requirement of the project.
- **Easy integration**: MFC components can easily be integrated with other C++ components of the project, namely the Core and the Graphs Generator.
- **Quality of GUI**: MFC facilitates the creation of highly user-friendly and professional-looking GUIs.
- **Experience**: The team already has the required knowledge of this tool and can quickly be fully functional using it.

The graphical display of results will be implemented using OpenGL inside the MFC window, as to ensure the complete integration of the graphical output inside the input window. OpenGL has been selected for our plotting objects for multiple reasons.

- **2D/3D Features**: OpenGL allows to easily draw 3D objects, as well as creating 2D projections, which is perfect for our plotting needs.
- **Animations**: OpenGL facilitates the creation of interactive animations.
- **Experience**: The team already has the required knowledge of this tool and can quickly be fully functional using it.

13.2.3 GUI Architecture

Three classes will need to be implemented for the GUI, but the MFC AppWizard will create most of them. The Class RollSim will be created as the main application class and the class RollSimDlg will be created as the main dialog class of our program, both by the AppWizard. The other important class involved in the GUI will be the GlView class (derived from CWnd) and it will have to be implemented without the help of the AppWizard. It will encapsulate all the functionality needed for the OpenGL graphs in the MFC window. To GlView objects will be created in the program, one for each graph in the window. The following is a list of the expected functionality provided by the GlView class :

- Creation
- Initialization
- Maintaining display settings:
    - animated or still graph
    - graph type to be drawn

- Support for animation of graphs
- Plotting of the different graphs:
  - Roll Angle vs. Time
  - Angular Velocity vs. Time
  - Angular Velocity vs. Roll Angle
  - Cross-section of the ship
  - 3D Roll Angle vs. Angular Velocity vs. Time
- All plots (except cross-section) should have the following characteristics:
  - Scaled to fit sub-window
  - Relevant axis names and graph title
  - Relevant axis scales and numbering
  - Pointer to the current index in buffers (small sphere)
  - Pointer moves along curve with time
- Cross-section graph
  - Scaled to fit sub-window
  - Show proportions of actual ship dimensions
  - Show relevant position of Metacenter and Center of gravity

13.2.4 Error-Handling

In this particular program, almost all input data must be check to prevent the user from entering data of the wrong type, data that would make the program crash or data that is impossible for the problem. MFC automatically takes care of the type of data entered. For example, it will prompt the user to enter an integer value (by means of a message box) if he or she tries to enter characters in an integer edit box.

Program crashes caused by impossible data, such as division by zero, will be redundantly prevented at the core level for extra safety, but the GUI will also check input data and provide the user with relevant error messages or warnings. Impossible input values, such as a negative length of ship, will be prevented by the GUI; even tough they do not pose any stability threat.

Screenshot 3: Example Error Warning

Part IV

SOS Roll Simulator

(Implementation Guide)

14. Introduction

The purpose of this section of the document is to introduce the reader to the specifics of the program's programming.

15. Programming Style

15.1. Hungarian Notation

In order to facilitate the identification of variable types in our code, the Hungarian notation has been adopted.

This notation was introduced at Microsoft during the development of OS/2. It is called "Hungarian" since its inventor, Charles Simonyi, is Hungarian.
Hungarian variable names start with a small number of lower case letters that identify the type of the variable. These letters are followed by a descriptive variable name that starts with an upper case letter, as illustrated in Table 16.

Prefix	Type	Example
b	boolean	bool bStillGoing
c	character	char cLetterGrade
str	C++ string	string strFirstName
si	short integer	short siChairs
i	integer	int iCars
li	long integer	long liStars
d	double	double dMiles
ld	long double	long double ldLightYears
if	input file stream	ifstream ifNameFile
is	input stream	void fct(istream &isIn)
of	output file stream	ofstream ofNameFile
os	output stream	void fct(ostream &osIn)
C	declaring a class	class CPerson

Table 16: Hungarian Notation Prefixes

15.2. Code File Headers

With the intention of identifying the code easily and effectively, file headers such as the following are used at the beginning of each code-containing file. The header conveys information such as the project name, the author name, the filename, the date on which the code implementation was started and a short description of the code.

```
//-----------------------------------------------------------------------------
//               S O E N  3 4 1 ,  S e c t i o n  U ,  2 0 0 2
//-----------------------------------------------------------------------------
//
//                                  P R O J E C T:
//                   Roll Motion of a Ship in Wave Simulator
//
//-----------------------------------------------------------------------------
//
// Author:             Team U1
// Date:               January 22, 2002
// File:               CRollMotionSimulation.cpp
// Description:        Implementation of the CRollMotionSimulation class
//
//-----------------------------------------------------------------------------
```

Figure 7: Example of File Header

15.3. Function Headers

With the aim of identifying the purpose and data flow in and out of each function, headers such as the following precede each implementation. They state the function name, its actions, input parameter types and outputs.

```
//-----------------------------------------------------------------------------
//function:           CRollMotionSimulation
//action:             class constructor, initializes all data members
//input:              CShip, CEnvironment, int, double, double, double
//output:             none
//-----------------------------------------------------------------------------
```

Figure 8: Example of Function Header

15.4. Comments

In order to facilitate understanding, communication, maintenance and troubleshooting comments are embedded within each code file so as to explain the actions performed to anyone reading the code.

16. Algorithm Implementation

16.1. General Considerations

16.1.1 Data Storage

The data is held by three buffers, implemented as dynamic arrays, storing calculation results for various time, roll angle and angular roll velocity values. The first cell of the arrays stores the initial value for the time, roll angle and angular roll velocity supplied by the user.

16.1.2 Initialization

Data members for the maximum roll angle and angular velocity are initialized to the corresponding initial values, as supplied by the user.

16.1.3 Computing Interval

During the computing of values, if the calculated roll angle is greater than 90° (θ_{max}: passed this angle, the ship necessarily capsizes), the computing process stops. Otherwise, the computing process until the user-set number of points has been calculated. Comparison statements integrated in the algorithms' solving will update the maximum angle and velocity values if ever necessary.

16.1.4 Buffer Adjustment

At the end of each algorithm, the actual number of points used for plotting graphs is adjusted. For instance, the points for which the roll angle is greater than 90 degrees are discarded.

16.2. Specifics for Implementation of Explicit Euler Scheme

A loop structure (for loop) is used to calculate and store in appropriate buffers the values for the next time, roll angle and angular roll velocity.

16.3. Specifics for Implementation of Crank-Nicholson Scheme

As this algorithm requires the two previous values of roll angle in order to calculate the next one, Algorithm 1 (Explicit Euler Scheme) is used to calculate the second roll angle and angular roll velocity values stored in the result buffers.

A "for loop" is used to calculate and store in appropriate buffers the values for the next time, roll angle and angular roll velocity.

16.4. Specifics for Implementation of Predictor-Corrector Scheme

This algorithm uses the predictor-corrector scheme for calculating the roll angle and angular roll velocity. The predictor and corrector constants are stored in static arrays that simulate the tables. The predictor scheme is first used to calculate the predicted roll angle. Then the corrected value of the angular roll velocity is calculated and combined with the predicted roll angle to calculate the next corrected roll angle. A for loop is used to calculate and store in appropriate buffers the values for the next time, roll angle and angular roll velocity.

Part V

SOS Roll Simulator

(Testing)

17. Introduction

This section of the document details the strategy used by the test team in order to ensure the highest level of reliability possible. The phases of the process are detailed using a high-level view, demonstrating the solidity of the testing strategy.

18. Cross-Review

The cross-review exercise is a known way to facilitate and accelerate the testing phase of the software development.
The code is reviewed by a programmer who hasn't contributed to the development of the reviewed module.
The reviewer compares the code with what was specified in the module's documentation and ensures the validity of the code.

If any mismatch is detected at this step, it is reported to the Team Leader, who will emit a ruling on the cause of the error and will assign personnel to correct the error.

19. Mathematical Engine Unit Test

19.1. Test Plan

The Mathematical Engine is composed of a few objects (*Ship*, *Environment*, *Solver*) embedded into a *Simulation*.

The plan is to ensure that the objects are able to react correctly when provided incorrect parameters, as well as ensuring the correctness of the *Solvers*.

19.2. Test Specification

19.2.1 Criteria

An object is considered correctly failsafed when no set accepted incorrect parameters.
The solver is considered correct if the error of each computed value is under $1x10^{-5}$ from a trusted model (in our case, the Excel Prototype), without any memory leaks.
We assume that previous file tests by the developers appropriately verified the internal functioning of the parts of the mathematical engine.

19.2.2 Seeding

Six seeds were be implemented in this module

1) Replaced a '-' by a '/' for the implementation of the Explicit Euler Solver.
2) Replaced a '+' by a '-' for the implementation of the Crank Nicholson Solver.
3) Inserted an incorrect Predictor constant in the Predictor Table of the Predictor-Corrector Solver.
4) Leaved a critical assertion for error checking.

5) Removed error checking on the Length parameter in class CShip.
6) Performed an invalid static casting from double to integer in the parameter setting of the gravitation.

19.2.3 Protocol

The test plan will be in two phases:
a) Parameter control validation
b) Solver validation

For the **parameter control validation**, we built a simple driver program creating each object and trying to set invalid values. The desired parameters are compared with those inside the object. If the bad parameter(s) is rejected, then the test is successful. We will have five different incorrect values for each parameter.

The **solver validation** consists of similar driver programs: one inside our Excel model and one inside our Core. Each driver will generate files of the same format with the computation results of their engine, with a precision of $1x10^{-8}$.

A separate utility will then compare the result files for the same parameter set and verify that the results are under the acceptable error define above.

"For loops" are nested so that each parameter's options will be tested with all the combination of the other parameters. Each parameter has been tested with two valid values, and the process has been repeated for each solver, for a total of 25 165 824 tests. The rapidity of the computation combined with the automation of the process ensures that this amount of test can be run efficiently. A first run of 30 tests of each solver will first be done, so as to optimize the bug detection level.

The module has been submitted to an **endurance/test testing**, being run non-stop for 12 hours. During this period, the software calculated extremely large simulations without any stability issue.

19.3. Test Results

All seeds were discovered. All results from the comparison match. This implies that either both models are defective in a similar manner (highly improbable) or that we have correct results since our data was also checked against theoretical results.

20. GUI Unit Test

20.1. Test Plan

Since our GUI is composed of two components, we must test these two components separately.
The plan is to ensure that appropriate parameter checking is performed, as well as the response to the user is appropriate and useful.
This test will also ensure that the graphing objects render the results correctly.

20.2. Test Specification

20.2.1 Criteria

The GUI is considered satisfactory when a useful error message pops up for every invalid operation we are performing on it.

The rendering objects are to be considered satisfactory when the image displayed matches the Excel Prototype.

We assume that the developer of the MFC GUI has performed a minimal test of conformity on his code.

We also assume that the developer of the rendering objects has tested for conformity between the rendering, so that the 2D/3D renders match each other and that the crossection display is conform to the displayed graphs.

20.2.2 Seeding

Four seeds were be implemented

1) The Z-axis of the graphs was inversed, so that the negative Z have an axis, but not the positive Z values.
2) The rendering object was to voluntarily skip a random number of points, 'approximating' the results.
3) The error message for invalid ship parameter combination was left blank.
4) The dropboxes were made to display an incorrect rendering object (i.e. swap the crossection rendering with the 2D angle Vs. time plot).

20.2.3 Protocol

The testing of the Data Entry is done manually, using a checklist of bad parameter combination and incorrect parameters.

The testing of the rendering objects uses 30 output files from the Core Testing done earlier, and comparing the render with the one displayed in Excel. The results should be visually identical.

20.3. Test Results

All four seeds were detected. All error messages are functional. All renders match the result from the Excel prototype.

21. System Test

21.1. Test Plan

The integrated system is user-input-based, which means that the only part of testing left to ensure is that the system reacts correctly to the user's input.

21.2. Test Specification

21.2.1 Criteria

This test will be considered successful when all tested sequences will have yielded the expected output, whether the parameters are adequate or not.

21.2.2 Seeding

Since the integration requires no untested elements, no seeding is added.

21.2.3 Protocol

We use a Markhov model (as detailed in Figure 9 and Table 17) with what we believe is a plausible usage profile in order to determine sequences of clicks, and ensure correctness for them.
We use a random generator on the model and generate 30 sequences of clicks (of a maximum of 6 clicks per sequence), as well as adding the 10 most important sequences.

We then perform these sequences twice: once with correct parameters, the other with at least one incorrect parameter.
The sequences are performed sequentially, that is: we do not but the system in its initial mode between each sequence.

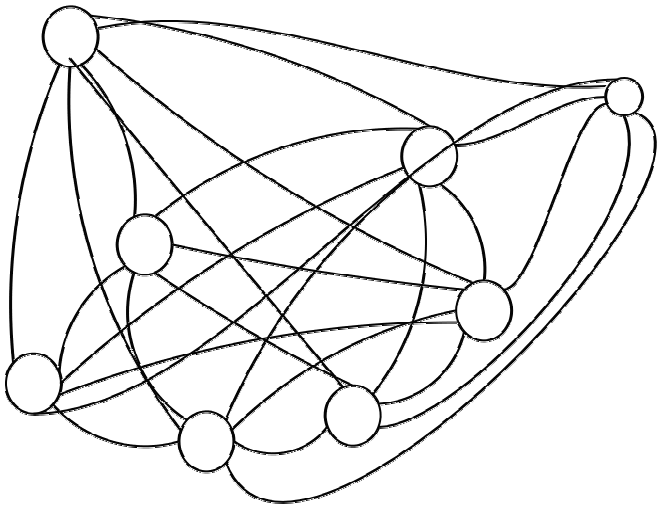


Figure 9: Simplified Markhov Model

Markhov Model of Usage and Stages (Probability of transfer)								
From / To	Start	Ship Parameter	Environment Parameter	Simulation Parameter	Calculate	Change Render	Animate	Close program
Start	-	35	10	20	30	2	2.9	0.1
Ship Parameter	-	9.5	5	5	80	0.2	0.2	0.1
Environment Parameter	-	5	5	9.5	80	0.2	0.2	0.1
Simulation Parameter	-	5	9.5	5	80	0.2	0.2	0.1
Calculate	-	14.5	15	15	0.4	30	25	0.1
Change Render	-	15	15	25	0.9	10	34	0.1
Animate	-	15	15	15	0.4	40	14.5	0.1
Close program	-	-	-	-	-	-	-	-

Table 17: State Change Probabilities of Markhov Model

21.3. Test Results

The system response to the sequences was conforming to expectation for all 40 sequences in both cases of correct and incorrect parameters supplied.

Part VI

SOS Roll Simulator

(Validation)

22. Introduction

The validation is the end of the development process for the development team. It allows the validation of the correctness of the implementation and the consistency with system requirements.

23. Validation Chart

ID	Function	Description	Implemented?
1	Mathematical Engine	Offers an object-oriented implementation of approximation algorithms for solving ODE. The Engine stops computing when the angle reaches θ_{max}.	✓
2	Data Bridge	Offers a gateway for the data to be passed from the Mathematical Engine components of the system.	Removed
3	2D Data Plotting of Roll Angle	Displays the graph of the roll angle vs. time.	✓
4	2D Data Plotting of Roll Speed	Displays the graph of the roll speed vs. time.	✓
5	Integrated Interactive GUI	Allows setting of parameters, run simulations and visualize results without restarting the program. The main window is divided as described in §3.2.1 This module is also responsible to warn the user of any error in parameters and/or general errors.	✓
6	3D Data Plotting of Roll Speed and Roll Angle	Displays a graph of roll angle • roll speed • time.	✓
6.5	2D Data Plotting of Comparison	Displays the graph of the roll speed vs. roll angle	✓
7	3D Data Plotting of Ship's Crossection	Displays an animation of the motion of the crossection of the ship trough the time of the simulation.	✓
7.5	Display of Graphs	Displays a graph selected by the user in full screen mode.	Removed
8	Playback Controls	Define controls for playback of the animation sequence yielded by (7). Playback will highlight the progression of the simulation on (7), (6), (4) and (3).	✓
9	Data Files	Allows the user to load/save data files holding simulation, ship and environment parameters.	To be implemented
10	Image Files	Allows the user to save the graphical plotting of (3), (4) and (6) into an image file.	To be implemented
11	Advanced Plot Manipulation	Allows the user to rotate and zoom the desired plot for better analysis	To be implemented

Table 18: Validation Chart

Part VII

SOS Roll Simulator

(Maintenance Recommendations)

24. Introduction

This section of the document details the suggested improvements that one could implement in further releases of the software. In addition, a change management process is suggested and could be followed by the maintainers.

25. Maintenance Recommendations

The following functions could be implemented:

ID	**9**
Name	Data Files
Description	Allows the user to load/save data files holding simulation, ship and environment parameters.
Algorithm	The data would be saved in a designated text file type. The user, trough the GUI, will be able to specify the location of the files. **Saving** The object would use standard file streams to write the contents of the parameter object to a file according to the file format specification. **Loading** Use standard file stream procedures to read and analyze the files line by line. Any violation in the file format will be considered as a corrupted file.
Data Structures	Object of related class
Source(s)	Marc-André Laverdière, Technical Writer

ID	**10**
Name	Image Files
Description	Allows the user to save the graphical plotting of (3), (4) and (6) into an image file.
Algorithm	The image map of the graph(s) would be extracted and the PNG image algorithm would then be applied to it. The data would then be written as a file. The location of the file will be set by the user trough the GUI.
Data Structures	
Source(s)	Marc-André Laverdière, Technical Writer

ID	**11**
Name	Advanced Plot Manipulation
Description	Allows the user to rotate and zoom the desired plot for better analysis.
Algorithm	Mouse clicks on the plotting areas are detected trough the GUI tools and the plot is then rotated accordingly.
Data Structures	none
Source(s)	Marc-André Laverdière, Technical Writer

Table 19: Suggested Function Addition

ID	9	
Name		Data Files
Description		Allows the user to load/save data files holding simulation, ship and environment parameters.
Algorithm		The data would be saved in a designated text file type. The user, trough the GUI, will be able to specify the location of the files. **Saving** The object would use standard file streams to write the contents of the parameter object to a file according to the file format specification. **Loading** Use standard file stream procedures to read and analyze the files line by line. Any violation in the file format will be considered as a corrupted file.
Data Structures		Object of related class
Source(s)		Marc-André Laverdière, Technical Writer

ID	10	
Name		Image Files
Description		Allows the user to save the graphical plotting of (3), (4) and (6) into an image file.
Algorithm		The image map of the graph(s) would be extracted and the PNG image algorithm would then be applied to it. The data would then be written as a file. The location of the file will be set by the user trough the GUI.
Data Structures		
Source(s)		Marc-André Laverdière, Technical Writer

ID	11	
Name		Advanced Plot Manipulation
Description		Allows the user to rotate and zoom the desired plot for better analysis.
Algorithm		Mouse clicks on the plotting areas are detected trough the GUI tools and the plot is then rotated accordingly.
Data Structures		none
Source(s)		Marc-André Laverdière, Technical Writer

Table 20: High-Level Design for Supplemental Functions

26. Change Management Process

The change management process proposed here is inspired by the NEMMCO's IT Change Management Procedures Version 6.0. It is basically constituted of four (4) basic steps:

Change Initiation: this step initiating and logging of the change request.
Change Assessment: this sub-process assesses the business and technical issues.
Change Authorization: this steps deals with authorization for the change to be progressed or the rejection of the change. Authorized changes are allocated to a particular release as part of this step.
Change Implementation: this steps plans, schedules and implements the changes to the said software.

26.1 Change Initiation

The Change Management Process is triggered by the identification of a need to improve the current software. **Anyone** in the team may initiate a change, including the customer. Once a requirement for change is identified, a Change Request is registered in the Change Request Form. Once the requester completes that form, it is given to the team leader so that it can be handed in to the appropriate person (i.e. GUI changes request to GUI people)

26.2 Change Assessment

The concerned team member(s) will performs a preliminary evaluation of the request to confirm the relevance of it. To do so, he/she will determine whether this change is within the scope of the system he/she is responsible for. This step serves two purposes, first that we don't loose time implementing extra features not asked for by the customer and second, that the Change Request was given to the appropriate team member.

Then, if the change is relevant, it is evaluated from strictly utilitarian viewpoint: that is how much money is the company going to loose if it doesn't implement that change.

Finally, the concerned team member must as well as determine the technical feasibility, risk and effect of the change within the overall software product.

A Change Assessment Form will be forwarded to the team leader for registration of the change and its further progression. This form will include all the assessments done by the concerned team member.

26.3 Change Authorization

The main concern of the Change Authorization is to give the final seal of approval on the Change assessment form. Judging by the evaluation the change in terms of cost, benefit and risk to the operation, it will authorize or reject the change to be developed.

Authorization by the Team Leader is always required for the change to progress. Then the Team Leader will notify affected team members of the outcome (including rejection of changes) of the authorization process.

The result will be a signed Change Assessment Form meaning that the change can be progressed, or if there is no signature that the change is rejected.

26.4 Change Implementation

This step takes care of designing, developing and testing the change. The objective here is to **perform** and **monitor** all relevant actions. We therefore ensure that the implementation of the proposed change is "free" of defect (thus not inserting new defect in software).

The concerned team member(s) will be verifying that all tests have been completed successfully.

In addition the programmers that will outline the action taken to implement the change will fill out a Change Implementation Form.

We should also inform the change requestor on the progress of the change implementation.

One of the more critical elements of the Change Management Process is keeping all team members advised of the status of the change. The team leader will be responsible for these notifications.

The final step is the Change Completion. Its purpose is to evaluate the completion status of the change and close the change record. The objective is to verify that the change was implemented in accordance with the specified change plan and that the desired output of the change was achieved.

The documents produced during the change completion will be the Closed Change Record.

Part VII

SOS Roll Simulator

(Appendixes)

27. Variables

Variable	Determination
T	User-Defined
Δt	User-Defined
θ	Computed from Simulation
θ'	Computed from Simulation
θ''	Roll Acceleration
θ_0	User-Defined
θ_0'	User-Defined
θ_0''	User-Defined
ω_w	Computed from Parameters
ω_e	Computed from Parameters
R	Computed from Parameters
B	Computed from Parameters
I_{xx}	Computed from Parameters
k_{xx}	Computed from Parameters
L	User-Defined
B (β)	User-Defined
D	User-Defined
T	User-Defined
KM	User-Defined
GM	User-Defined
ΔGM	User-Defined
W	User-Defined
V	User-Defined
C_0	User-Defined
C_u	User-Defined
A_d	User-Defined
H_e	User-Defined
LBP	User-Defined
G	User-Defined
θ_{max}	User-Defined

Table 21: Variables and Source

28. Values for Predictor-Corrector Scheme

B (P)	K=1	K=2	K=3	K=4	Error
B (P) 1K	1	-	-	-	½ h 2 F i
B (P) 2K	3/2	-1/2	-	-	5/12 h 3 F ii
B (P) 3K	23/12	-16/12	5/12	-	3/8 h 4 F iii
B (P) 4K	55/24	-59/24	37/24	-9/24	251/12 h 5 F iv

Table 22: Predictor Stage Coefficients and Error

B (C)	K=0	K=1	K=2	K=3	K=4
B (C) 1K	1/2	½	-	-	-
B (C) 2K	5/12	8/12	-1/12	-	-
B (C) 3K	9/24	19/24	-5/24	1/24	-
B (C) 4K	251/720	646/720	-264/720	106/720	-19/720

Table 23: Corrector Stage Coefficients