Concordia University
Department of Computer Science
and Software Engineering
Software Engineering

COMP 354/1 --- Summer 2013 --- Section BB

Software Requirement Specification
Team C

Team members information	
Name	**SID**
Ashwath George	9733469
Nicholas Gignac	9128964
Robert Jakubowicz	6045707
Caroline Labbe	6320945
Brian Lam	1696785

Table of Contents

1. Introduction

The C-Ker program, from C-Labs, is a Vessel Monitoring System, delivered during the course of COMP 354, where the teacher will be referred to as the 'customer'. As a first delivery, this document exposes the various items of the project, from its description and required items, to a complete breakdown of its goals, constrains and functionalities. It will also give an overview of the resources anticipated to be needed by this project, as well as its scoping and projected scheduling.

2. Project Description

Stated in the document COMP354_Project2013-1.pdf provided by the teacher, regarding the Term Project for Comp 354 – Introduction to Software Engineering, Section 2, Part 2 (Title of Project), the current development regards a Vessel Monitoring System (VMS). According to Section 2, Part 3 (Project Description):

*"This project is a very simplified version of real Vessel Monitoring System (VMS). As part of the project you are required to develop a simple radar simulator to feed data to your VMS. The vessel scenario definition has been defined and a sample file is given to you (**comp354_vessel.vsf**). Some information has been given in this vessel scenario file (vsf)."*

As requested in Section 2, Part 4, 5 and 6, there are three main components to the VMS: the Radar Simulator, the VMS itself and a GUI/Display. The radar simulator will be responsible for taking the text file input provided for the simulation purpose, and will be computed into data the VMS can display into the GUI. The VMS will compute trajectory, movement (speed + direction) and will also compute two (2) radius alerts for each vessel: Low Risk from 50 to 200 meters and High Risk from 50 meters and under. VMS specification requirements also state that it has to be able to handle up to 100 vessels at a time, as well as having a refresh rate of 5 seconds.

The GUI will include two (2) main parts: a log in screen and the VMS display system. The log in screen will discriminate between users with operator credentials and administrative credentials. Under current customer (the professor) request, the authorisation level and available function for each class of users has not been defined yet and is awaiting further information.

The VMS/display system will come with a built-in amount of vessels of specified types. In accordance with Section 2, Part 5, the C-Ker VMS will be able to add additional types if requested by the customer. Also, the GUI/Display system will come with two (2) built-in displays: a table format and a 2D visual graph. The table format display will contain the following sections and will be able to be sorted according to each of them:

"Vessel ID / Type / X Position / Y Position / Speed / Course / Distance"

Again, as required in Section 2, Part 5, the table display will be updated every five (5) seconds. As the 2D graphic display, it will represent the vessels as dots and direction with vectors. If during the production of this project the customer's requirements change regarding the graphical display, C-Labs will be able to adjust in consequence.

In the event the customer requests additional feature and/or modification to existing features, C-Lab will comply with request as long as a reasonable time table can be provided for the delivery of the said request.

3. Goals and Constraints

Currently, the project goals are restricted to the original functional requirements described in section 3.1, and therefore will mirror any changes, be they additions or reductions, made to those requirements by the client. Time is the sole major constraint; if these goals can be accomplished ahead of schedule; further incremental upgrades could potentially be made to the GUI in order to visually emulate signal processing methodologies currently being utilized in real-world radar systems.

3.1 Functional Requirements

The overall C-Ker VMS will comprise of the following subsystems:

- A Radar Simulator that is capable of reading Vessel Scenario Files (.vsf) and performing calculations based in the information parsed from those files.

- A Vessel Monitoring System that receives positional data on the vessels and calculates the distances between them, generating alarms when certain conditions are met.

- A GUI that is capable of displaying vessel information in tabular form as well as representing data from the Radar Simulator in the form of a Cartesian grid, with individual vessels depicted as blips.

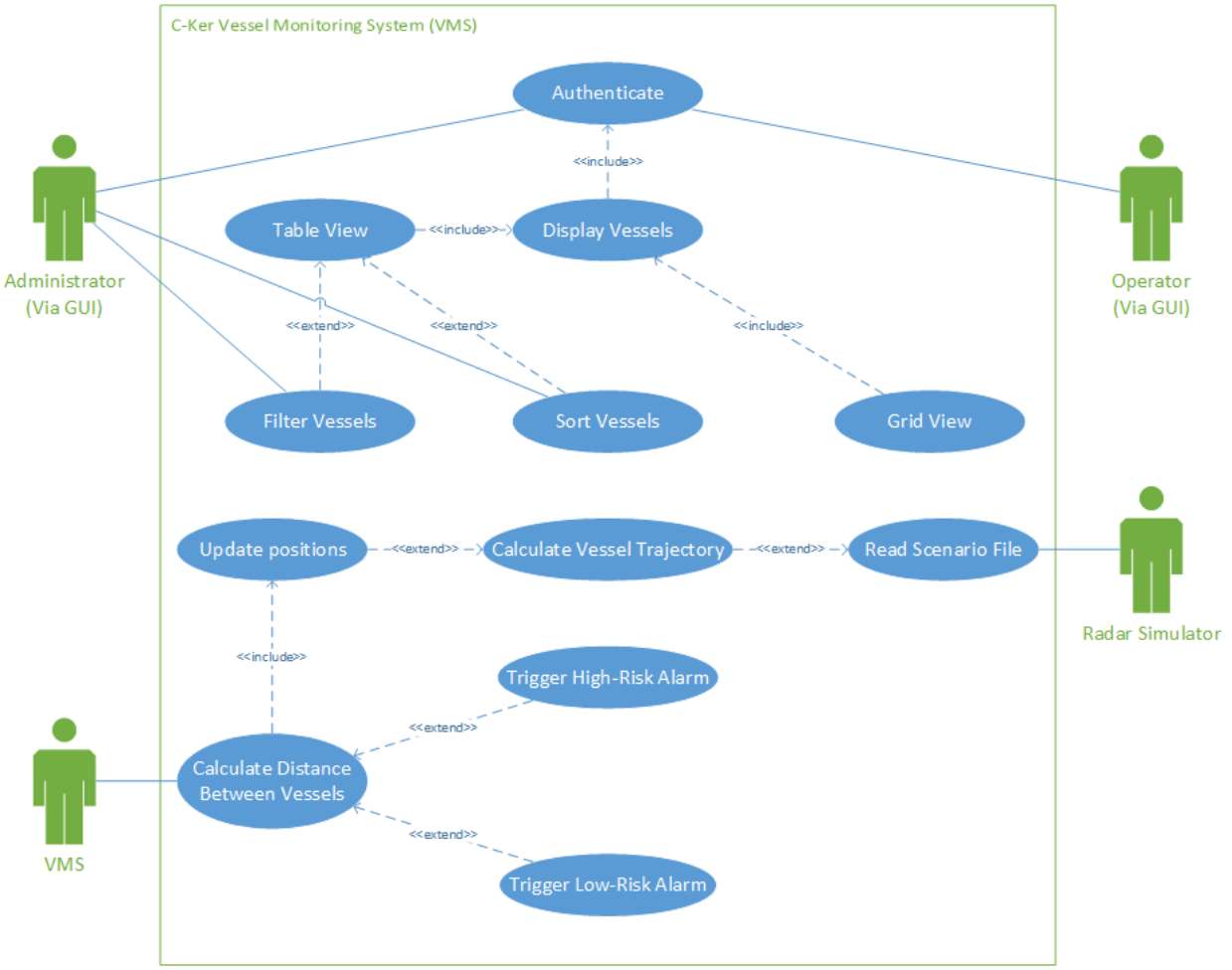

Figure 1 – Use Case Diagram (UCD)

3.1.1 Listing Vessels in a Table

Information about each vessel is displayed in a table format, allowing users to inspect vessel data.

Related Use Cases:

- Updating Vessel Positions

Preconditions:

- Vessel information must be loaded and updated correctly.

Steps:

- The application displays vessel information in a table on screen.
- The user views vessel information as desired.

Difficulty: Low
Importance: High

3.1.2 Filtering Vessel Listing by Type

The displayed vessel information can be filtered by vessel type, allowing users to view or hide certain vessel types only.

Related Use Cases:
- Listing Vessels in a Table

Preconditions:
- Vessel information must be loaded and listed correctly.

Steps:
- The user selects vessel types to be filtered through the application interface.
- The application updates the listing accordingly to only show wanted vessel types.
- The user views vessel information as desired.

Difficulty: Low
Importance: High

3.1.3 Sorting Vessel Listing by Data

The displayed vessel information can be sorted according to selected data, such as position, speed, type, etc., allowing users to organize and view vessels with more flexibility.

Related Use Cases:
- Listing Vessels in a Table

Preconditions:
- Vessel information must be loaded and listed correctly.

Steps:
- The user selects a data type through the application interface.
- The application sorts the listing according to the selected data.
- The user views vessel information as desired.

Difficulty: Low
Importance: Medium

3.1.4 Displaying Vessels on a Grid

Vessels are represented graphically through blips on a Cartesian grid, giving users a top-down visualization of the simulation in two-dimensional space. Vessel positions are updated in real-time.

Related Use Cases:
- Updating Vessel Positions

Preconditions:
- Vessel information must be updated continuously and correctly.

Steps:
- The application draws the Cartesian grid, with blips representing vessel positions.
- The grid is redrawn at regular intervals to reflect updates in vessel positions.
- The user observes the map and vessels as desired.

Difficulty: Medium
Importance: High

3.1.5 Loading Vessel Information

The Radar Simulator reads information on vessel IDs, initial coordinates and velocities from a scenario file.

Preconditions:
- Scenario files must exist and be formatted correctly.

Steps:
- A specific scenario file is selected.
- The simulator reads the file, parses the relevant information and stores it.

Difficulty: Low
Importance: High

3.1.6 Updating Vessel Positions

The Radar Simulator plots vessel trajectories by performing the necessary physics calculations using the information parsed from the corresponding scenario file.

Related Use Cases:
- Loading Vessel Information

Preconditions:
- Vessel information must be loaded correctly.

Steps:
- The simulator reads the initial position of a vessel.
- The vessel's subsequent positions are calculated over time, based on its velocity along the coordinate axes.

Difficulty: Low
Importance: High

3.1.7 Triggering Risk Alarms

Trajectories calculated by the Radar Simulator are fed to the Vessel Monitoring System (VMS), which calculates distances between vessels, and triggers specific alarms under certain conditions.

Related Use Cases:
- Updating Vessel Positions

Preconditions:
- Vessel information must be updated continuously and correctly.

Steps:
- The VMS continuously calculates distances between vessels based on their coordinates.
- If the distance between any two vessels is between 200m and 50m, the VMS triggers a Low Risk Alarm for those vessels (can be reflected on the grid by a yellow line between the two blips).
- If the distance between any two vessels is less than 50m, the VMS triggers a High Risk Alarm for those vessels (similarly reflected on the grid by a red line between the two blips).

Difficulty: Low
Importance: High

3.2 Domain Model

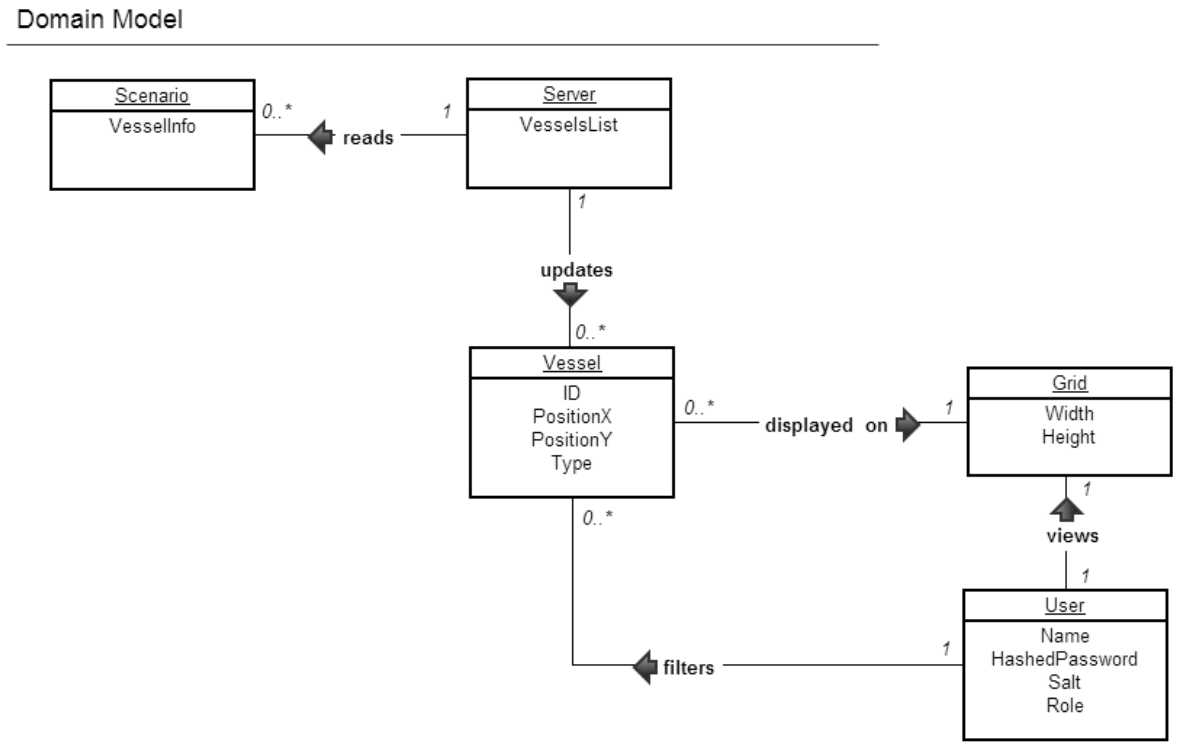

Figure 2 – Domain Model

A *Server* reads the information about *Vessels* from a *Scenario* file. It then calculates the new positions of the *Vessels* and updates them accordingly. Each *Vessel* is displayed on a *Grid* to show its location more intuitively. An authenticated *User* can view the *Vessels* on the *Grid*. If he has the permissions to do so, he can also filter the *Vessels* he wants to see.

3.3 Constraints and Qualities

The following qualities are what we consider quality standards and non-functional requirements that the C-Ker should attain:

Usability:

The application must allow a smooth experience for the user. The learning curved should be minimized and the interface design must follow Windows's interface best practices. The visual design must be appealing and modern.

Reliability:

The system must be tested under different conditions to make sure that it will not fall into an infinite loop or crash.

Security:

Only authorized users are allowed to access the system and view the vessels. Only administrators can filter the vessels. Passwords must not be stored in plain text instead they are stored as a hash.

Scalability:

The system must maintain the same performance speed even if the number of vessels increases up to 100 vessels.

Constraints:

- The application must run on a Windows operating system or mono on Linux.
- The application will not be optimized for multicore processors.
- The application can only have English language support.
- Only one user at a time can be logged in.

4. Resource Evaluation

The resources for this Project will be broken down into two (2) sections: Human Resources and Technical resources. Due to the nature of this project, Physical Resources (e.g.: Machinery, Vehicule, etc), Real Estate Resources (e.g.: Building, Rent, etc) and Financial Resources (e.g.: Financing, Budgeting, etc.) will be discarded.

4.1 Human Resources

Ashwath George
Developer

Availability:
- Works full time
- Available most weekends, on demand.

Education:
Bachelor Degree, Computer Applications w/ Major Computation Arts

Skills:
- **Programming**:
- Java - Advanced
- C++ - Advanced
- C# - Intermediary to Advanced
- Python – Advanced

Nicholas Gignac
Team Coordinator

Availability:
- Works three (3) days a week
- Flexible/Adaptive weekday schedule
- Unavailable on Tuesdays and Thursday nights due to classes for the duration of the course
- Available most weekends, on demand.

Education:
Bachelor Degree, Computer Science, Information System

Skills:
- **Programming**:
- Java – Intermediary to advanced
- C++ - Introductory
- **Other**:
- Communication
- Planning
- Documenting

Robert Jakubowicz
Lead Developer

Availability:
- Studying full time.
- Available most days.

Education:
Bachelor Degree, Computer Science, Computer Systems

Skills:
- **Programming**:
- Java – Intermediary to Advanced
- Ruby - Advanced
- C++ - Advanced
- C# - Advanced
- **Other**:
- Git – Advanced
- NUnit - Intermediary

Caroline Labbe

Documenter and Tester

Availability:

- Works full time.
- Available most weekends, Wednesdays, and on demand.

Education:

Bachelor Degree, Computer Science, Information System

Skills:

- **Programming**:
- Java – Intermediary to Advanced
- C++ - Introductory
- **Other**:
- Communication
- Documenting

Brian Lam

Developer

Availability:

- Available most evenings and weekends; will dedicate time as necessary

Education:

3rd year Computer Science student, Computer Games option

Skills:

- Programming Languages: C++, C#, Java
- 12 months internship experience as software developer
- Familiar with development for both console and GUI applications
- Familiar with development processes: designing, coding, testing, etc.

Roles of every member have been decided according to their skills and quality in order to insure the best performance possible from every one of them.

Because most of the human resources have other obligations (e.g. full-time or part-time job), this will be very important to evaluate the time each individual is able to spend on the project. Since meeting in person requires everyone to spend travel time, most of the meeting will be done through conference calls using Skype, online communication software.

4.2 Technical Resources

The Technical Resources required for this project are based on C-Lab's team personal equipment. From the team's personal computer, either laptop, desktop, PC or Mac, to communication device and technology such as phones or emails, most of the equipment will be provided individually by each team members.

The program will be built using Visual Studio 2012, acquired through Concordia University's ENCS portal and will be develop on and for Windows 7. Due to the team residing in various locations, we will be using Github as repository for our coding and development files, and use Git as version control system. In order to facilitate development of the GUI, the Windows Presentation Foundation library (WPF) will be integrated to VS2012.

Finally, the testing framework will be done using NUnit test framework, which will allow us to automate some section of the testing framework.

5. Scoping

As stated in section 3, time is the major constraint for the realization of the project. In the event of the team not being able to meet the schedule, less crucial and with lower priority requirements/features will be left out. Is has been determined that the appealing and modern design is the first quality to be eliminated if needed. Developing a more basic and simple GUI can be really less time consuming. If this is left out, time will also not allow creating the upgrade for the real-time radar input. If the situation is worst, the next quality that will be partially left out is the reliability. If time is missing, it can happen that the team will lack time for testing the VMS and this will increase the possibility of the program crashing.

In order to not have to get to this point, C-Lab ensures that everything in its capacity will be executed for the benefit of the customer.

6. Plan

In order to bring the C-Ker project to its end, scheduling and planning must be executed. The following parts are only tentative and anticipation of what will need to be delivered, considering that exact due date for the following delivery along with their content was never divulged precisely or directly. C-Lab will adapt to the changing environment and requirements as given by the customer. For practicality, and since Delivery 1 is the item at hand, we will begin scheduling and planning for Delivery 2 and 3 only, beginning at the first week of June.

6.1 Activities

According to the document 'Project Information', Section 1, Part 2 (Teams and Roles), each delivery had specific goals to achieve:

- Delivery 1: "*design of the interface for the core application*."
- Delivery 2: "*creating the design*."
- Delivery 3: "*designing UI*"

Assuming that our version of Delivery 1 meets the required standards of the customer, it is safe to assuming that we are left with two items on the deliverable list: Creating the design and designing (and implementing) the UI. Considering the customer's request of using Agile Software Process, C-Lab will work on a SCRUM-type activity planning. Unlike SCRUM, we will have weekly face-to-face meeting, either before or after lecture, but will keep in touch on a daily basis using email and Skype.

Unless further delays and extension happen, we expect Delivery 2 to be due 3 weeks after the handing-in of Delivery 1 (Week 6 to Week 8 + Mid-Term Break). Since we do not yet have the exact delivery requirement for Delivery 2 and 3, we can only anticipate the Activities Breakdown.

For Delivery 2, we expect the Radar Simulator and VMS, fully functional, to be the required items to be delivered, both on a raw, but testable display and GUI. As far as Delivery 2 is concerned, we expect it to be the delivery of the fully functional VMS, along with any additional requirement the customer may have requested between now and then. The full breakdown of each activity is to follow.

6.2 Project Estimates

According to our current progress, if requirements from the customer remain as it is, the following estimates can be made regarding the activities needing completion in order to deliver the C-Ker program on time:

<u>VMS and Radar Simulator:</u>

Currently stated as been main component of delivery 2, there is 19 days available between the handing in of delivery 1, and the projected due date of delivery 2 (20 actual days minus Quebec Civil Holiday St. Jean, June 24th). We expect the initial development of the VMS and Radar Simulator to take approximately 5 days, with hours spread on availability of programmers and work divided between programmers in accordance with Lead Programmer Robert J.'s planning. The GUI programmers will share task in order to generate a basic, yet testable, GUI for the purpose of testing the VMS and Radar Simulator, and simultaneously provide programming assistance to the VMS programmer.

We expect to need approximately 3 days to complete the initial testing of the first version of VMS and Radar Simulator. As much as we expect our program to meet requirements, no one is perfect, so we are anticipating a required 3 days of post-testing programming in order to correct for the flaws and error found during the initial testing phase. An additional 3 days of testing post-correction will be done to ensure full quality of the VMS and Radar Simulator phase.

Thanks to this tight scheduling plan, we anticipate to have 5 days as buffer. We expect these days to be either consumed for additional testing and programming in the event the initial and secondary test phase requires additional time. Otherwise, we expect this time to be used for R&D purpose in the event the customer requires new features for his system.

GUI/Display:

Considering that some elements of the GUI will need to be functional in order to test the previous phase, we expect the final development of the GUI, including both type of display, ordering controls and log in screen, to take about 3 days. Same as for the VMS, the usage of the time will be left for the programmers to handle on their own. Once completed, we expect the initial phase to take 3 days. Same as before, a 3 day period will be allowed post-testing to bring the corrective to the GUI and yet another 3 days of testing afterward. Once the GUI will be fully completed, we expect to need 2 days to assemble both parts of the program into one symbioses system. Again, the remaining 4 days will be reserved either for buffering time in the event one of the phases takes additional time. If not, this time will be reserved for the final R&D phase, if there are, again, new requirements from the customer.

Documenting:

Documenting will be done as the phase's advance, parts done by the programmer themselves, parts done by the testing and documenting team, in order to keep track of every elements of the project. A full, tentative, schedule including expected documenting time is located in Section 6, Part 4.

6.3 Activities Assignments

In accordance with Section 4.1 of this document, each Activity will be assigned to the teammate with the skills best suited for it.

Radar System:

- Main Programmer: Robert J.
- Assisting Programmers: Ash G. & Brian L.
- Tester: Nicholas G.

VMS:

- Main Programmer: Robert J.
- Assisting Programmers: Ash G. & Brian L.
- Tester: Caroline L.

GUI/Display:

- Main Programmers: Ash G. & Brian L.
- Tester: Robert J.

Documentation & One-Minute Meeting:

- Caroline L. & Nicholas G.

6.4 Schedule

C-Lab | C-Ker Project

June 2013

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
2	**3**	**4**	**5** **Delivery 1 Due.**	**6** Completion of VMS & Radar Simulator *Documenting	**7** Completion of VMS & Radar Simulator	**8** Completion of VMS & Radar Simulator
9 Completion of VMS & Radar Simulator	**10** Completion of VMS & Radar Simulator	**11** Testing: VMS & Radar *Documenting	**12** Testing: VMS & Radar	**13** Testing: VMS & Radar *Documenting	**14** Correction from test result: VMS & Radar	**15** Correction from test result: VMS & Radar *Documenting
16 Mid-Term Break Week Correction from test result: VMS & Radar	**17** Testing: Corrected VMS & Radar *Documenting	**18** Testing: Corrected VMS & Radar	**19** Testing: Corrected VMS & Radar	**20** Available R&D Time or Additional Testing Time *Dedicated Documenting Time Begins	**21** Available R&D Time or Additional Testing Time	**22** Available R&D Time or Additional Testing Time
23 Mid-Term Break Ends Available R&D Time or Additional Testing Time	**24** St. Jean Holiday	**25** Available R&D Time or Additional Testing Time *Dedicated Documenting Time Ends	**26** **Expected Delivery 2 Due Date**	**27** Development of GUI/Display *Documenting	**28** Development of GUI/Display	**29** Development of GUI/Display
30 Development of GUI/Display						

C-Lab | C-Ker Project

July 2013

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1 Canada Day	**2** Testing: GUI/Display *Documenting	**3** Testing: GUI/Display	**4** Testing: GUI/Display *Documenting	**5** Correction from test result: GUI/Display *Documenting	**6** Correction from test result: GUI/Display
7 Correction from test result: GUI/Display	**8** Testing: Corrected GUI/Display *Documenting	**9** Testing: Corrected GUI/Display	**10** Testing: Corrected GUI/Display	**11** Union of VMS with GUI *Dedicated Documenting Time Begins	**12** Union of VMS with GUI	**13** Available R&D Time or Additional Testing Time
14 Available R&D Time or Additional Testing Time	**15** Available R&D Time or Additional Testing Time	**16** Available R&D Time or Additional Testing Time *Dedicated Documenting Time Ends	**17** **Expected Delivery 3 Due Date**	**18**	**19**	**20**
21	**22**	**23**	**24**	**25**	**26**	**27**
28	**29**	**30**	**31**			

C-Lab | C-Ker Project

August 2013

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
4	5	6	7	8	9	10
11 **Classes Ended**	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 3, 4 & 5 – Calendar

6.5 Risks

Due to the scale of this project, we will keep the risk factor to a minimalistic amount, and will only focus on 4 types of risk: External Events, Unrealistic Estimates, Poor Communication and Technical.

External Events:

Events such as:

- Injury & sickness
- Death within a team member's family
- Course drop

This is the most serious source of risk from our point of view, due to its unpredictability and heavy impact in case of occurrence. As a matter of fact, C-Lab lost a team member due to family sickness overseas even before the project began, as we were originally set as a team of six members. Considering every team member's commitment and dedication to this course, we can rule out, at this point in time, the course drop factor, but injury, sickness, and death is still an element to be considered. We consider this element to be a high risk factor.

Unrealistic Estimates:

Events such as:

- Unfixed Deadlines
- Modified Deadlines
- Miscalculation due to Scope of the project and inexperience of the team working on a VMS.

Second most serious source of risk, yet more likely to happen, the time estimates related to this project are hard to plan due to the vagueness of the information feed to the team, as well as the unpredictability of the customer with his requests. The last element from the list above, is the most likely to happen, and while it may not jeopardize the successful delivery of the program on time, it could definitively pose additional stress factor on the team. We consider this element to be a medium-high risk factor, with the notation that it is the most likely to happen.

Poor Communication:

Events such as:

- Misinterpretation of an electronic communication (email, text)
- Unsuccessful delivery of an electronic communication (email, text)
- Unreachable team member.

The risk from this element is definitively less than those previously, but on the contrary, it is even more likely to happen also. As observed earlier during the completion of delivery 1, some items failed to deliver to the required mailbox via email. It had no impact on the handing in of the delivery because it was planned ahead accordingly, but it proves that

this risk is still present, even if it has a lesser impact. In the event of an unreachable team member, we can cope to this issue by using more than one communication system, ranging from emails, phone/text, Skype, Facebook and LinkedIn. We consider this issue to be from medium to low, but with a higher certainty of happening to some extent.

Technical:

Events such as:

- Team member not familiarized with C#
- Computer malfunction
- lost/corruption of data
- lost internet connection
- Technical issue

The risk level from this element is classified as lowest for us. As far as most of the traditional technical issue is concerned, the fact that the entire C-Lab team is from a computer science background neutralizes this treat to a minimum, as we are aware of every contingency possible. As for the C# issue is concerned, we do not believe this to become an issue, since one of the team member is more than specialized at it, and that the rest of the team as at least a base in C++ and the minimal Object Oriented Programming classes from ENCS. We classify this issue as minimal risk, with even lower level of happening.