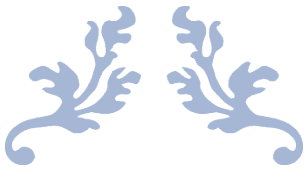


TEAM E

SOMALIAN PIRATE SIMULATOR

SOFTWARE DEVELOPMENT PLAN

SEPTEMBER 9, 2021

Team E
CS 499
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Table of Contents

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Introduction

Project Overview

The International Maritime Bureau (IMB) reported that over the period 2005–2012 nearly 1,000 ships were fired at, chased, boarded, or hijacked by pirates. According to the IMB these pirates often operate from bases in Somalia. Our team has been tasked with the creation of a featureless, gridded map to simulate interactions between navel patrols, merchant shipping, and pirate attacks. This simulation is intended to help the IMB study the likelihood of pirate attacks as a first step to eliminating future pirate attacks off of the coast of Somalia.

Scope

Somalian Pirate Simulator (SPS) will contain the following features:

- **a.** 400 (Horizontal) x 100 (Vertical) Featureless Grid Map
- **b.** Grid (0,0) in Upper Left Corner, (400,100) in Lower Right Corner
- **c.** Simulation should start at time step 1 in a paused state
- **d.** User ability to speed-up/slow-down time in 1x, 2x, 10x, 20x increments
  - **i.** 1x time steps represent one hour in real time
- **e.** Four Types of Entities with the following attributes
  - **i.** Cargo Ships
    - **(1).** Unarmed Merchant Vessels carrying materials that cannot resist pirate attacks
    - **(2).** Cargo Ships already on map move one grid directly to the east during each time step
    - **(3).** Spawn on the west side of the map
      - **(a).** Probability of spawning on to map = 0.50
      - **(b).** All western most column grids are equally likely to spawn Cargo Ship
      - **(c).** Upon entry Cargo Ships do not move—Entry is counted as that steps movement
    - **(4).** Cargo Ships on the eastern-most column of grids will be removed from the simulation on the following turn
  - **ii.** Naval Patrol Ships
    - **(1).** Armed Warships that can defeat pirate ships and rescue captured cargo ships
    - **(2).** Patrol Ships already on map move two spaces directly to the west of the map during each time step
    - **(3).** Spawn on the east side of the map
      - **(a).** Probability of spawning on to map = 0.25
      - **(b).** All eastern most column grids are equally likely to spawn Patrol Ships
      - **(c).** Upon entry Patrol Ships do not move—Entry is counted as that steps movement
    - **(4).** Patrols on the western-most column of grids will be removed from simulation on the following turn

iii. Pirate Ships
- **(1).**Small armed ships that can capture unarmed cargo ships but cannot resist naval patrol vessel attacks
- **(2).**Pirates already on map move one grid directly to the north
- **(3).**Pirates accompany (occupy the same space) Captured Cargo Ships in same grid after successful capture
- **(4).**Spawn on the south edge of the map
  - **(a).**Probability of spawning on to map = 0.40
  - **(b).**All southern-most row grids are equally likely to spawn Pirate Ships
  - **(c).**Upon entry Pirate Ships do not move—Entry is counted as that steps movement
- **(5).**Pirate Ships on the northern-most row of grids will be removed from the simulation on the following turn

iv. Captured Cargo Ships
- **(1).**Cargo ships that have been captured by Pirates
- **(2).**Captured Cargo Ships already on map move one grid directly to the south
- **(3).**Captured Cargo Ships are spawned by a successful attack by Pirate Ships
- **(4).**Captured Cargo Ships that are currently on the southern-most row of grids will be removed from the simulation on the following turn
- **(5).**Accompanied by Pirate Ship that captured the cargo ship (occupy the same space)

f.Five actions that ships can enact in either a 3x3 (adjacent) grid or 4x4 grid (only Evasion)

i.Defeat
- **(1).**If after <u>both</u> ships have moved, a Pirate and a Patrol are:
  - **(a).**on the same grid
  - **(b).**on adjacent grids (including diagonal)

  the Pirate ship is defeated by the Patrol and removed from the simulation

ii. Capture
- **(1).**If after <u>both</u> ships have moved, a Pirate and a Cargo are:
  - **(a).**on the same grid
  - **(b).**on adjacent grids (including diagonally)

    Cargo Ship is turned into a Captured Ship and the Pirate Ship will now accompany the Captured Ship (occupy the same grid space)

iii. Rescue
- **(1).**If after <u>both</u> ships have moved, a Captured Ship and a Patrol are:
  - **(a).**on the same grid
  - **(b).**on adjacent grids (including diagonally)

    The Captured Ship will return to the Cargo Ship state and the Pirate Ship will be removed from the simulation

iv. Evade

(1). If after both ships have moved, a Cargo Ship and a Pirate are within the same 4x4 grid:

(a). Cargo Ship can make a one-time evasive maneuver by moving 1 grid diagonally to the northeast

(2). Cargo Ships cannot evade the same pirate more than once

(a). Cargo Ship to be placed in lower left grid space in the middle of 4x4 grid to simulate realistic approach of pirate ships

(b). Multiple Pirate Evasion

(i). If a Cargo Ship is able to evade more than one pirate ship in the same move, it should attempt this move

(ii). If it cannot evade both ships, a ship should be chosen at random to evade

v. No Action

(1). Other than the above listed actions, if any other pair of ships are adjacent to one another in a turn, there should be no action taken

g. Event Counter

i. Cargos Entered

ii. Cargos Exited

iii. Cargos Captured

iv. Captured Cargos Rescued

v. Patrols Entered

vi. Patrols Exited

vii. Pirates Entered

viii. Pirates Exited

ix. Pirates Defeated

x. Evades Not Captured

xi. Evades Captured

h. Display Elements—all display elements must update with each time step

i. Map Display—System must have a GUI where user can view the simulation in a similar manner as Figure 1

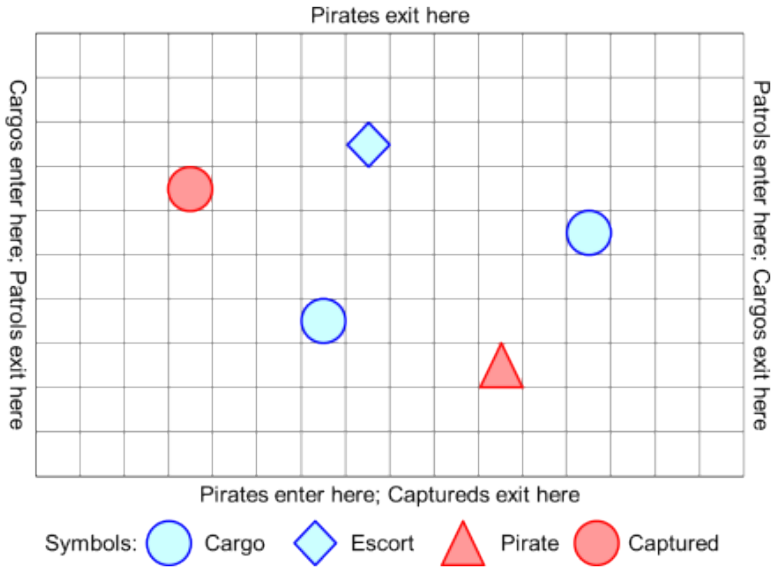


Figure 1: Example of Map Display

ii. Counts Display

(1). Display the Time Step Number, the updated counts listed in section f above

i. User Controls

i. Start—clicking this button from a paused state should cause the simulation to run continuously until the Pause or End buttons are pressed

ii. Simulation Speed Button Set

(1). 1x—Run the simulation at normal speed

(2). 2x—Run the simulation at double speed (2 time steps per update)

(3). 10x—Run the simulation at ten-time speed (10 time steps per update)

(4). 20x—Run the simulation at twenty-time speed (20 time steps per update)

iii. Pause—clicking this button from a running state should pause the simulation at the current time step and not allow updates until deactivated

iv. Single Step

(1). Has no effect if the simulation is currently running

(2). If simulation is in paused state this will allow for 1 update to run

(a). Single step should enact a single (1x) step no matter speed chosen

(b). End—ends simulation from both the running and paused state

Context Diagram

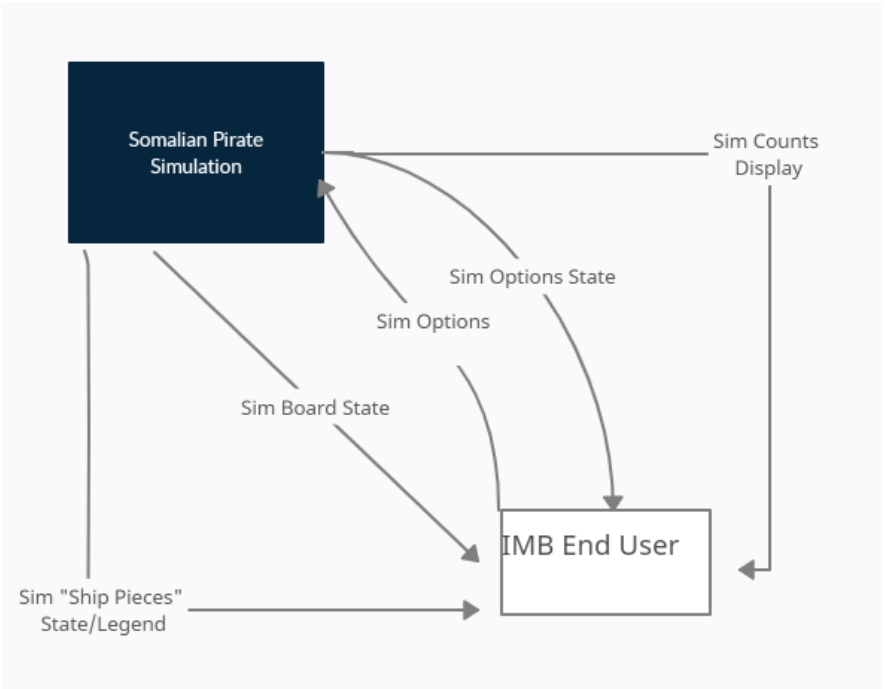


Figure 2: Context Diagram

Summary—Introduction

The introduction for SPS is made up of three main sections:

1. A brief introduction of The Somalian Pirate Simulation(Page 2) describing why Team E has been tasked with this project.
2. The scope of the application(Pages 2-5) detailing the requirements that will be implemented to achieve the project goals as outlined by the customer.
3. The context by which the user will interact with the simulation(Page 6).

References

IMB Piracy Reporting Centre. ICC Commercial Crime Services. (2017, July 7). https://www.icc-ccs.org/piracy-reporting-centre.

Software Development Plan

Personnel

a. Nathan Moore—Team Lead

Nathan is a senior Computer Science student at the University of Alabama in Huntsville; proficient in C++, C#, Java, Python, Haskell, and R languages. Nathan also has more than eight years of restaurant consulting and management experience.

b. Luke Tomlinson—Quality Lead

Luke is a senior Computer Science student at the University of Alabama in Huntsville. With a skill set mainly consisting of front-end design and development using Swift for iOS. Luke has produced and maintained two professionally released iOS applications.

c. Aaron Mendez—Technical Lead

Aaron is a senior Computer Science student at the University of Alabama in Huntsville. His skills include object-oriented design in both C++ and Java. Aaron currently holds a professional position in CS in Huntsville, Alabama.

d. Jack Bragg—Test Lead

Jack is a senior Computer Science student at the University of Alabama in Huntsville. His skills include both academic and professional software development in: Python, C#, C++, Java, Arm, and Bash. Jack is also proficient in the creation of software design documentation such as UML and sequence diagrams for professional use.

Project Schedule

a. Epics
- **i.** Deliverable Documentation Preparation—Sprints A-E
  - **(1).** Deliverables
    - **(a).** SDP—September 09, 2021
    - **(b).** SDP Presentation—September 07, 2021
    - **(c).** SRS/Backlog—September 28, 2021
    - **(d).** SRS/Backlog Presentation—September 23/28, 2021
    - **(e).** Architectural Design—October 21, 2021
    - **(f).** Architecural Design/Backlog Presentation—October 19/21, 2021
    - **(g).** UI Design—November 09, 2021
    - **(h).** UI Design Presentation—November 04/09, 2021
    - **(i).** Application Gold Build—December 07, 2021
    - **(j).** Application Delivery Presentation—November 30/December 02, 2021
  - **(2).** Milestones
    - **(a).** Delivery of SDP—September 09, 2021

(b). Delivery of SRS—September 28, 2021
(c). Delivery of Architectural Design—October 21, 2021
(d). Delivery of UI Design—November 09, 2021
(e). Delivery of Application—December 07, 2021

ii. Conceptual Design Phase—Sprints B-C
(1). Deliverables
(a). SDP—September 09,2021
(b). SDP Presentation—September 7, 2021
(c). SDP Presentation—September 07/09, 2021
(d). SRS/Backlog—September 28, 2021
(e). SRS/Backlog Presentation—September 23/28, 2021
(2). Milestones
(a). Delivery of SDP—September 09, 2021
(b). Delivery of SRS—September 28, 2021
(c). Delivery of Backlog—September 28, 2021

iii. Application Development—Sprints C–E
(1). Deliverables
(a). Delivery of Architectural Design—October 21, 2021
(b). Architectural Design Presentation—October 19/21, 2021
(c). UI Design—November 09, 2021
(d). Prelim GUI Presentation—November 04/09, 2021
(e). Application—December 07, 2021
(2). Milestones
(a). Begin Software Development—September 30, 2021
(b). Delivery of UI Design—November 09, 2021
(c). Delivery of Application—December 07, 2021

iv. Organizing Testing Effort—Sprints D–E
(1). Deliverables
(a). UI Design—November 09, 2021
(b). Prelim GUI Presentation—November 04/09, 2021
(c). Application Beta Build Delivery—November 11, 2021
(d). Application Gold Build—December 07, 2021
(2). Milestones
(a). Begin Debugging—October 12, 2021
(b). Delivery of UI Design—November 09, 2021
(c). Application Beta Build Delivery—November 11, 2021
(d). Delivery of Application—December 07, 2021

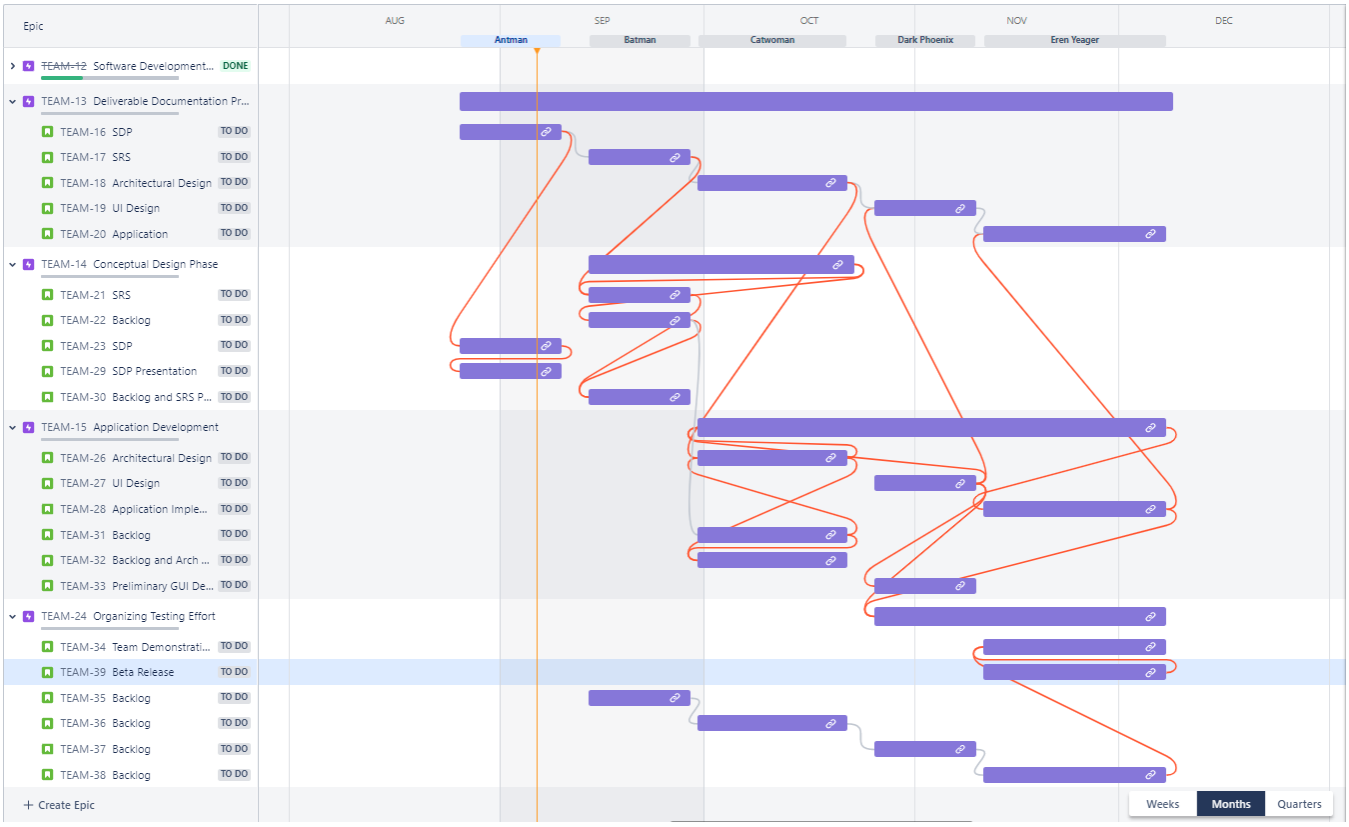


Figure 3: Weekly Software Development Plan

List of Deliverables

1. SDP Presentation—September 7, 2021
2. SDP—September 09, 2021
3. SRS—September 28, 2021
4. SRS/Backlog Presentation—September 23/28, 2021
5. Backlog—September 28, 2021
6. Architectural Design—October 21, 2021
7. Architecural Design/Backlog Presentation—October 19/21, 2021
8. UI Design—November 09, 2021
9. UI Design Presentation—November 04/09, 2021
10. Application Beta Build—November 11, 2021
11. Application Delivery Presentation—November 30/December 02
12. Application Gold Build—December 07, 2021

List of Milestones

1. Delivery of SDP—September 09, 2021
2. Delivery of SRS/Backlog—September 28, 2021
3. Begin Software Development—September 30, 2021
4. Begin Debugging—October 12, 2021

5. Delivery of Architectural Design—October 21, 2021
6. Delivery of UI Design—November 09, 2021
7. Application Beta Build Delivery—November 11, 2021
8. Delivery of Application Gold Build—December 07, 2021

Risk Management Plan

- **a.** Risks
  - **i.** Multiple Team Members with Conflicting/Dynamic Work Schedules
  - **ii.** Differentiating OS:Mac/Windows
  - **iii.** Covid-19 Exposure
  - **iv.** Multiple Team Members with 3+ Group Projects
  - **v.** Sick Family Members/Team Members requiring team members to step back from tasks
  - **vi.** Multiple Team Members taking courses at 400 level
    - **(1).** Plan—Team has created a flexible Software Development Plan which should allow for members of the development team to take the time they need to work through personal/professional issues which may arise during the semester. This plan includes early release dates for deliverables for extra testing or development if any risk evaluated above does indeed occur.

Summary—Software Development Plan

Software Development plan is organized into five main sections:

1. A list of personnel(Page 7) , and the teams risk management plan(Page 10).
2. A tentative project schedule(Page 7) which describes the teams current plan to execute long and short term tasks.
3. A list of deliverables(Page 8) showcasing all deliverables and the dates they will be completed.
4. A list of milestones(Page 9) describing when the team will reach key areas of the development process.
5. The teams tentative risk management plan(Page 10) which lays out the teams plan if something were to happen to a teammate that may affect the project development schedule.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Software Requirements Specifications

To be provided on October 1st.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Architectural Design

To be provided on October 22nd.

Detailed Designs

To be provided on <date>.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Test Plan and Reports

To be provided on October 22nd.

Conclusion

To be provided on December 7.

Appendix

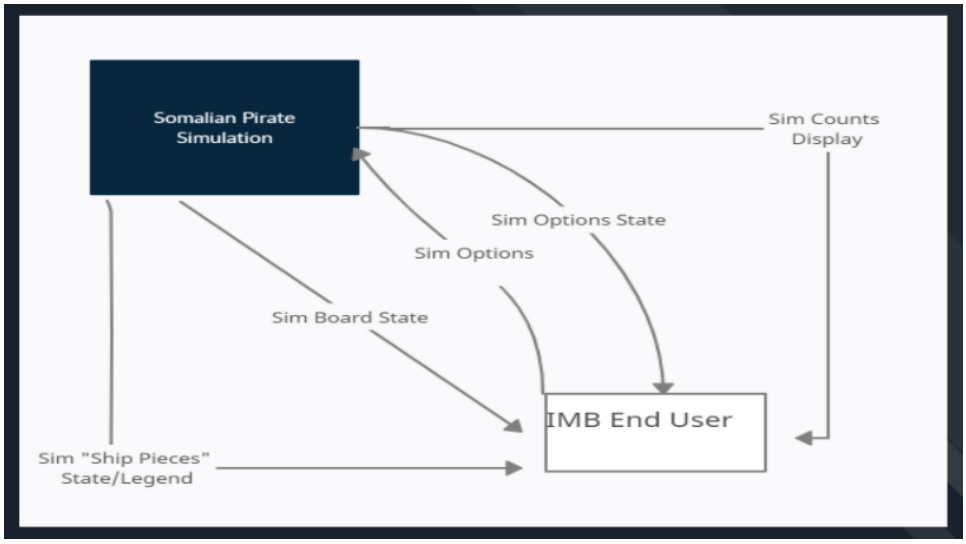
Appendix A – Statement of Work

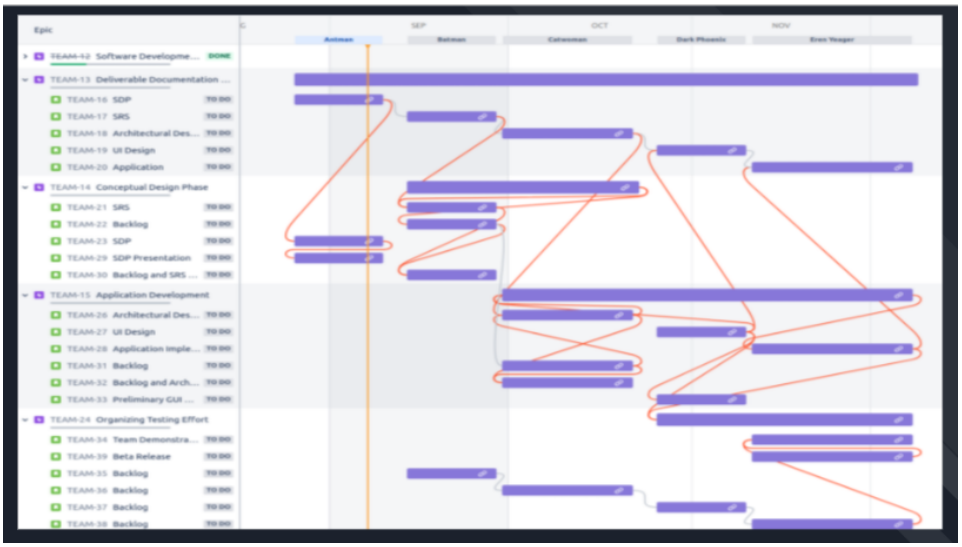
Appendix B – Meeting Minutes

Appendix C – Presentation Slides

Software Development Plan Presentation

Figure 1. An example of the simulation map display.


Somalian Pirate Simulation
Sim Counts Display
Sim Options State
Sim Options
Sim Board State
IMB End User
Sim "Ship Pieces" State/Legend




Thank you!