ENSE 623 PROJECT – Verification of Canal Systems

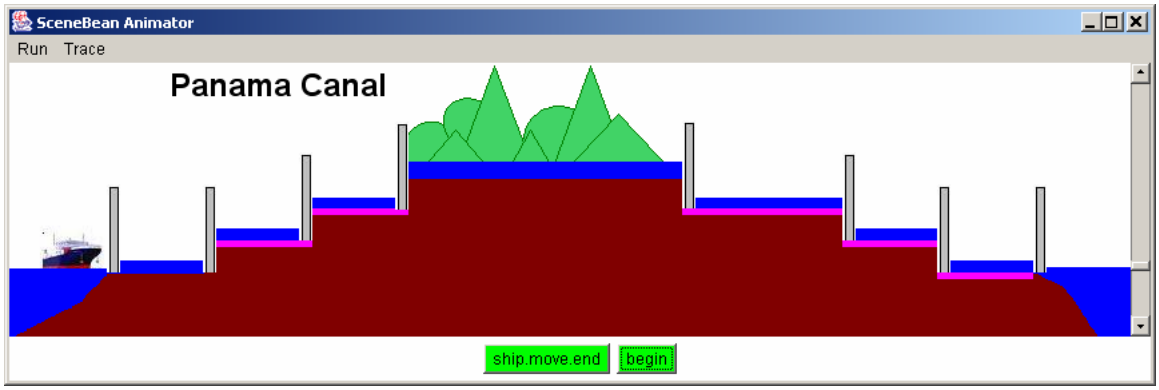
I. PROJECT DESCRIPTION

This project involves developing a model of a canal system very similar to the Panama Canal. A detailed set of requirements shall be generated. We shall then use the Unified Modeling Language (UML) to design the initial model of the system. Afterwards, the LTSA Tool shall be used to verify the UML model using its sequence diagram and state diagram capabilities. In the end the, using the Extreme Markup Language (XML) and the LTSA Tool, and animation of the canal shall be created to verify and validate the model. A brief description and analysis using the UPPAAL tool shall also be given.

Canal Description:

The canal is a two lane waterway mechanism used to allow the transport of ships. The entrance and exit of the canal contains three sets of locks that hold the ships. As a ship approaches the first chamber, valves below the compartment are released and the water level reaches that of the outside of the canal. The gates then open and the ship moves into the first chamber. After the gates lock, valves of the first and second chamber are opened to allow the water level of the first chamber to rise and match the second chamber. The gates are again opened and the ship moves to the second chamber. This process continues in the third chamber until the ship approaches the divide. The ship will then travel through the divide to the opposite side of the canal. Again, the process of raising the water level and unlocking the gates is repeated through three sets of locks/compartments until the ship exits the canal completely.

A physical description of the canal is as follows:



II. GOALS/SCENARIOS

The goal is to create a design/model that supports multiple continuous ships passing through the canal. Both directions of the canal can be active at one time.

Scenario 1 – One ship approaching canal.
n = number of current chamber (1 thru 6)

1. Ship approaches canal
2. Captain of ship requests passage
3. Operator of Control Center monitors condition
    a. If another ship in chamber 1, repeat step 3
    b. If water level in chamber 1 exceeds sea level release water valves in chamber 1 and repeat step 3
    c. If water level in chamber 1 equals sea level, close water valves
4. Open lock/gate
5. Signal to ship to pass through gate
6. Monitor ship location sensor
    a. If ship not completely in chamber *n* repeat step 6
7. Close lock/gate to chamber *n*
8. Operator of Control Center monitors condition
    a. If another ship in chamber *n+1*, repeat step 8
9. Release water valve in chamber *n* and *n+1*.
10. Monitor water sensor
    a. If water level chamber *n+1* exceeds water level in chamber *n* repeat step 10
11. Close water valves of chamber *n* and *n+1.*
12. Go to step 4 and continue until ship through to canal divide.

Scenario 2 – One ship exiting canal.
n = number of current chamber (1 thru 6)

1. Ship approaches canal
2. Captain of ship requests passage
3. Operator of Control Center monitors condition
    a. If another ship in chamber 4, repeat step 3
    b. If water level in chamber 4 exceeds divide water level release water valves in chamber 4 and repeat step 3
    c. If water level in chamber 4 equals divide water level, close water valves
4. Open lock/gate
5. Signal to ship to pass through gate
6. Monitor ship location sensor
    a. If ship not completely in chamber *n* repeat step 6
7. Close lock/gate to chamber *n*

8. Operator of Control Center monitors condition
    a. If another ship in chamber *n+1*, repeat step 8
9. Release water valve in chamber *n* and *n+1*.
10. Monitor water sensor
    a. If water level chamber *n+1* exceeds water level in chamber *n* repeat step 10
11. Close water valves of chamber *n* and *n+1.*
12. Go to step 4 and continue until ship through canal.

III. REQUIREMENTS

The requirements of the canal system are sub-divided into the two following sections: Physical Requirements and Operational/Safety Requirements.

Physical Requirements:

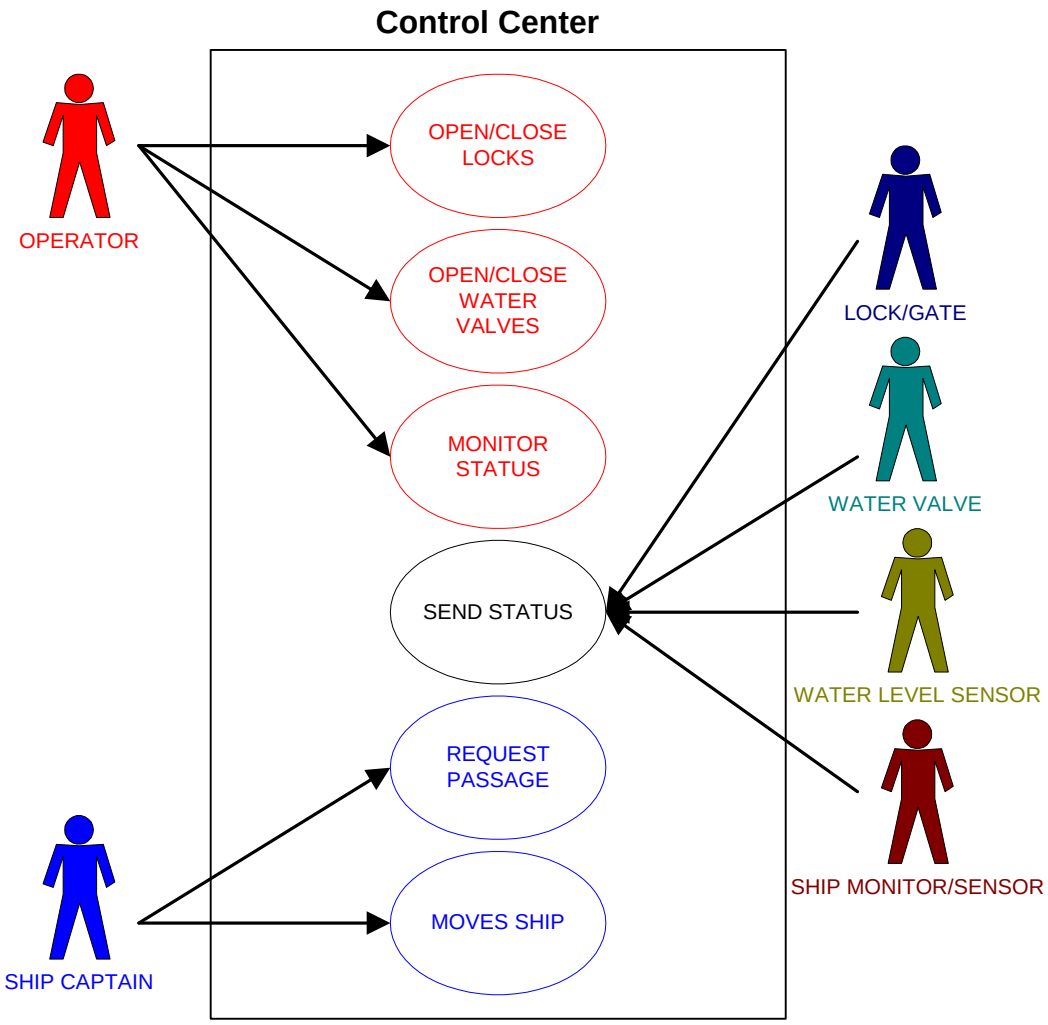
- The canal shall have two lanes for operation at each end
- Each end of the channel shall have three chambers to hold ships
- Each end of the canal shall have four gates/locks
- Each chamber shall contain five water valves to allow the increase and decrease of the water levels
- The canal dimensions shall be as follows
    - Lock chambers shall be 33.5 meters wide (110 feet)
    - Lock chambers shall be 305 meters wide (1000 feet)
    - Lock chambers shall have the depth of 26 meters (85 feet)
    - Ships no long than 294.13 meters (965 feet) in length, 32.31 meters (106 feet) in beam, and 12.04 meters (39.5 feet) in draft shall be allowed within a chamber.

Operational Requirements:

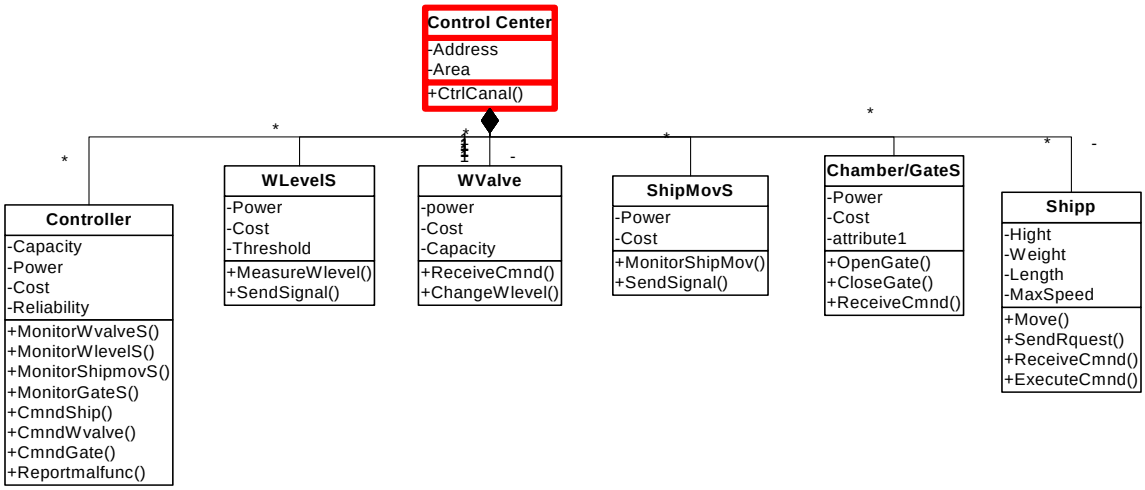
- The canal divide (known as the area of water between the three sets of locks) shall allow the traffic of two-way ships.
- No two ships shall be permitted to enter one chamber at one time
- The gate of each chamber shall open only when the water level of that chamber equals the water level of the next chamber
- The water valve of a chamber containing a ship shall not open if the next adjacent chamber contains another ship.
- A gate to a chamber shall not open if that chamber contains a ship.

IV. UML DIAGRAMS

a. USE CASES

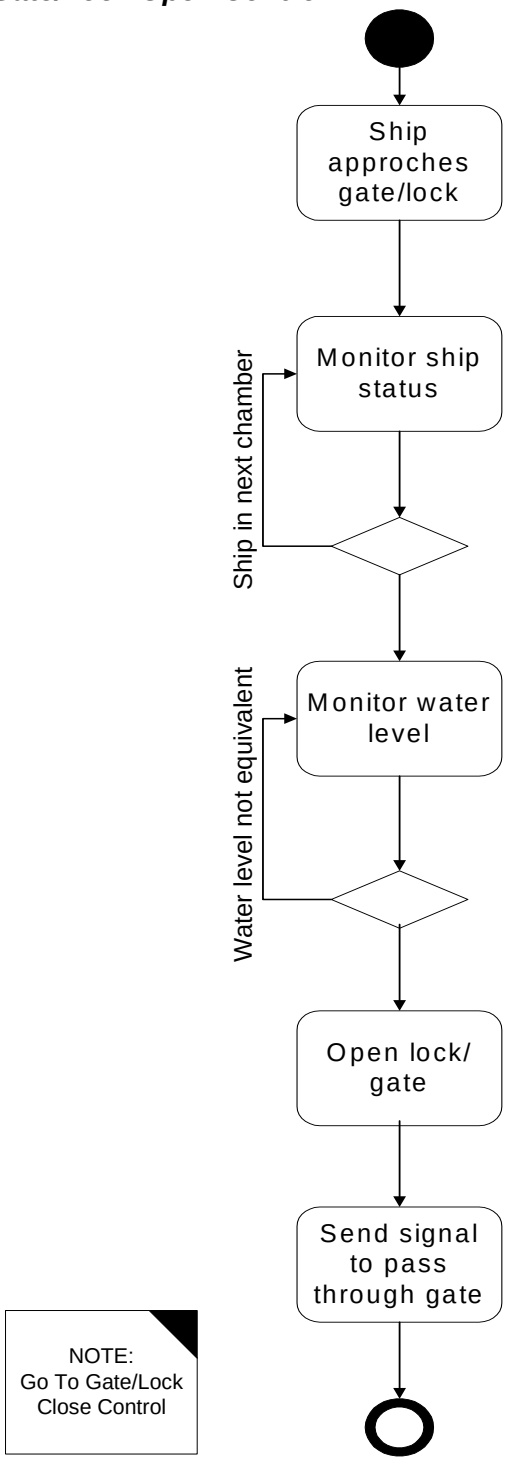


b. CLASS DIAGRAM

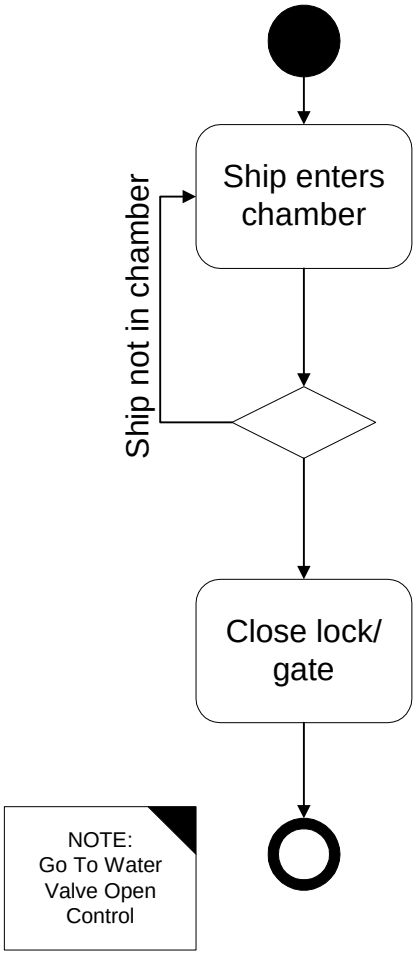


c. ACTIVITY DIAGRAMS

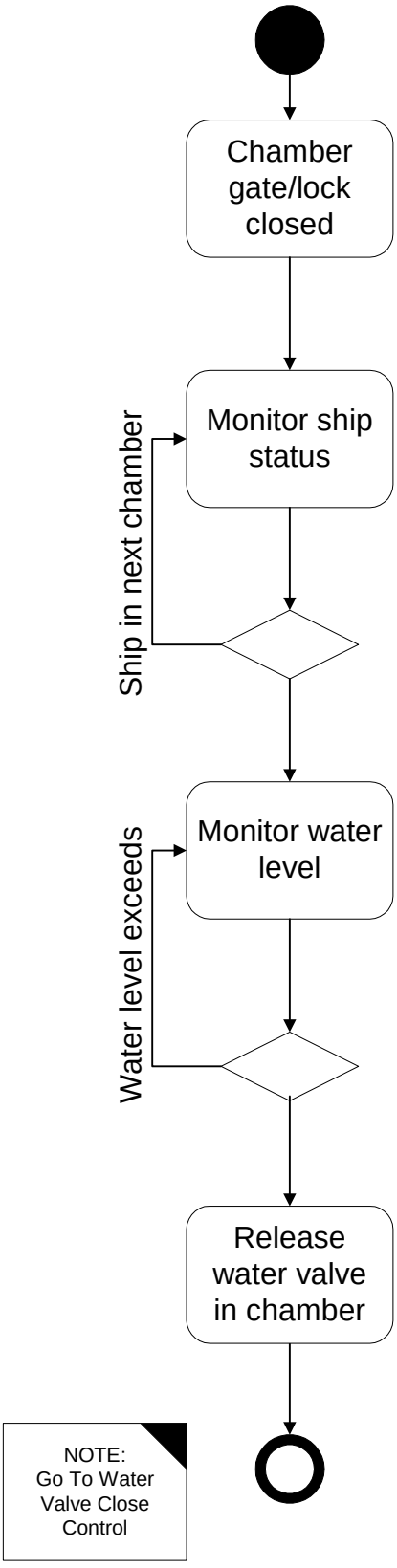
Gate/Lock Open Control:



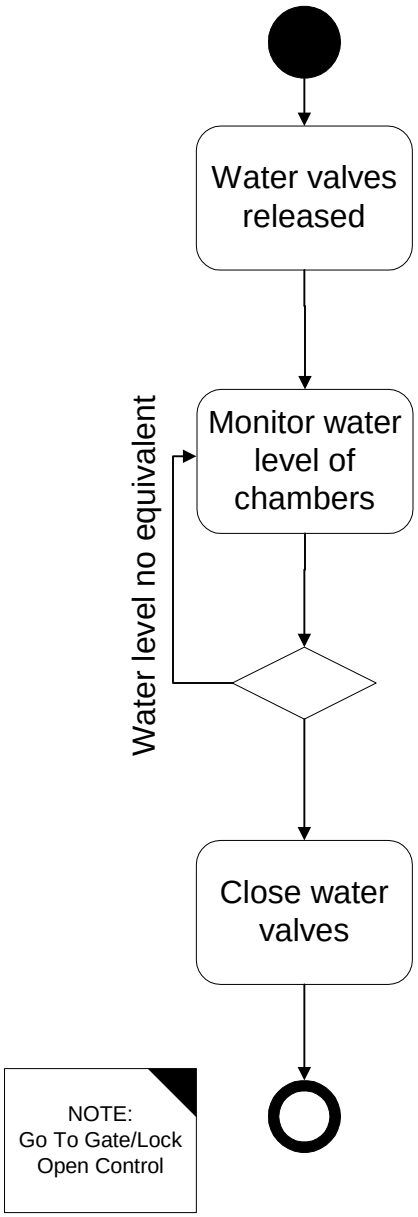
Gate/Lock Closed Control:



Water Valve Open Control:

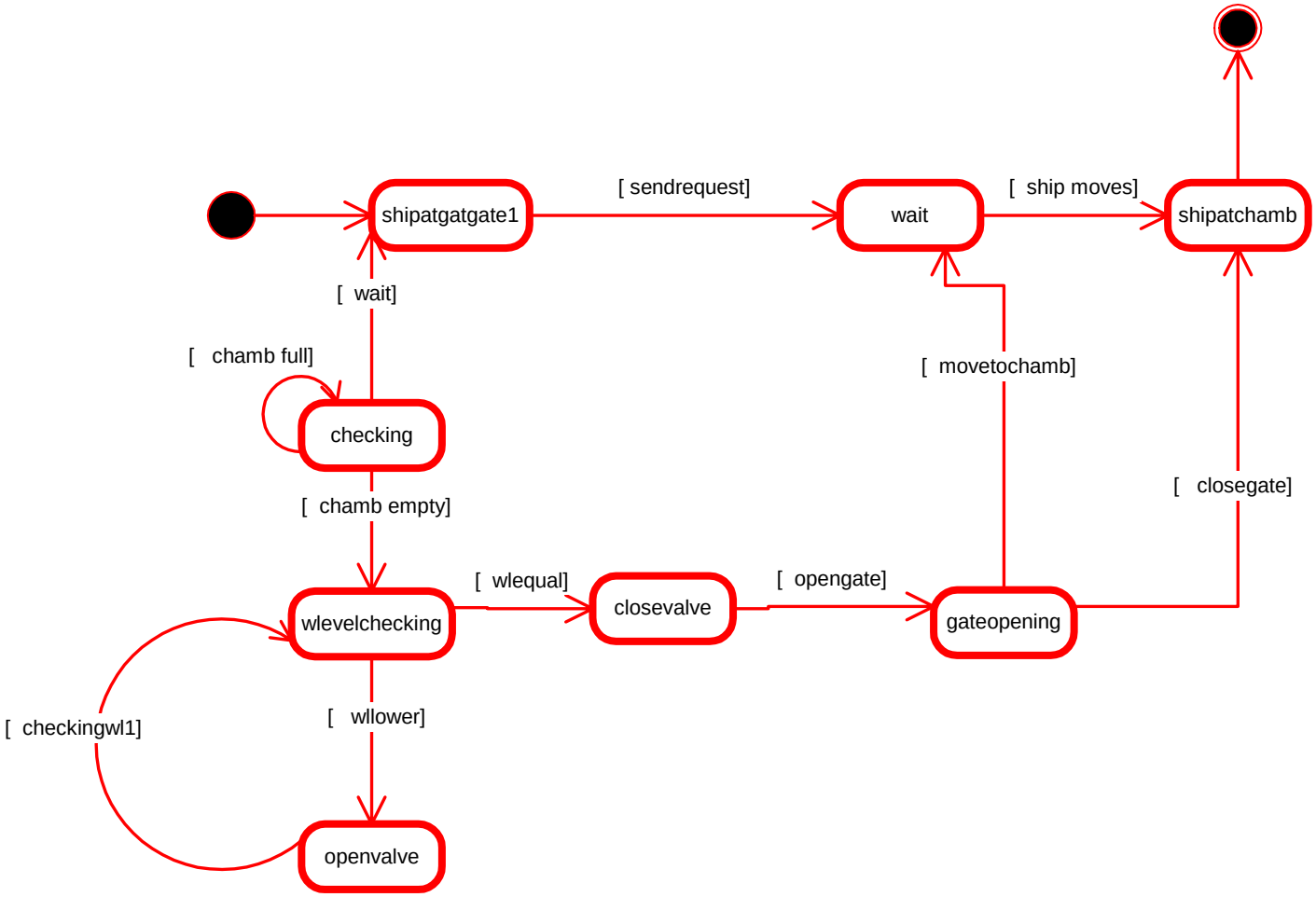


Water Valve Close Control:



d. STATE CHARTS

Before any modeling with the LTSA tool was performed, a simple state diagram of the system operation was designed by hand. Please see below.



V. LTSA Tool

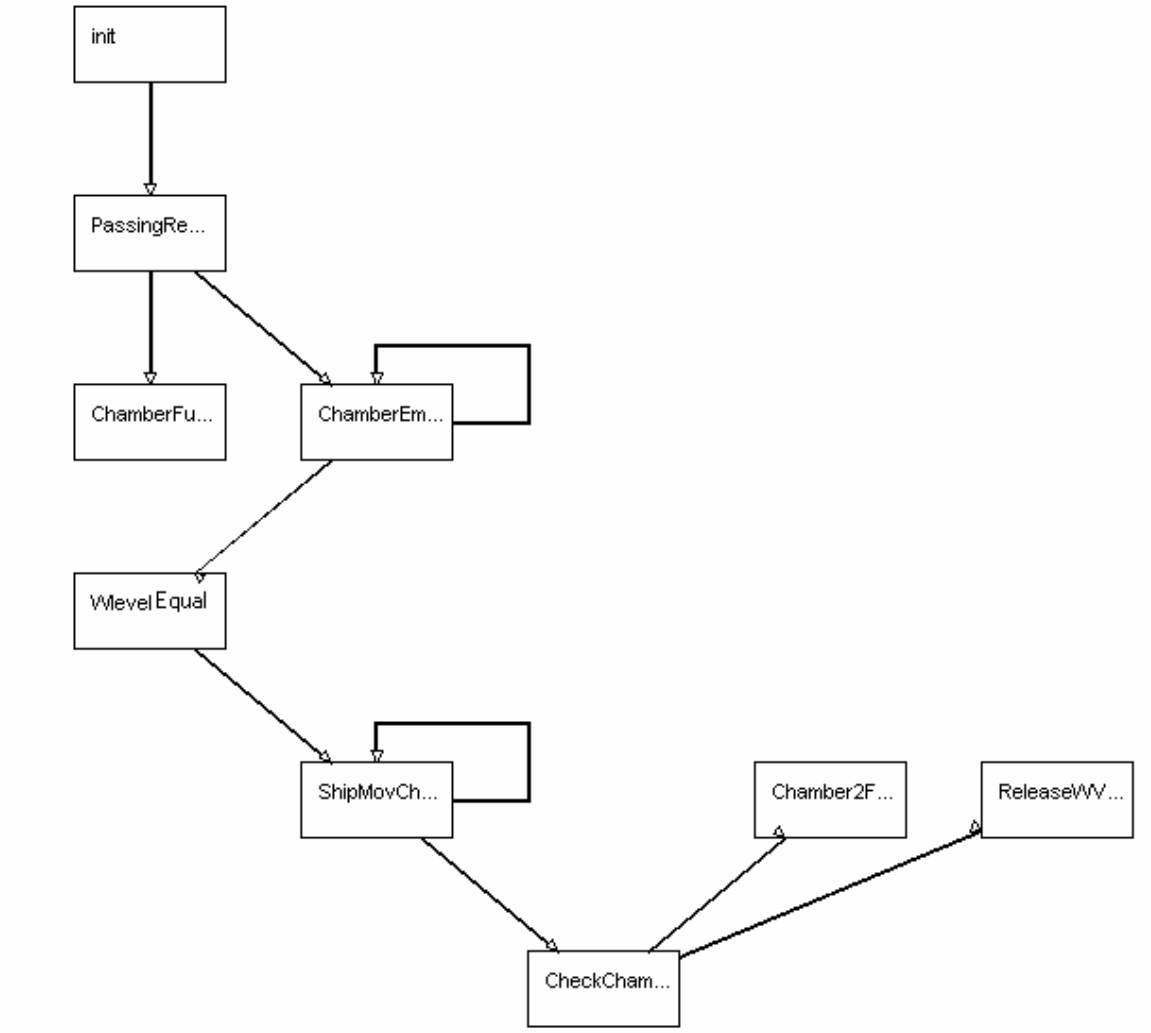
The LTSA is a tool used to verify concurrent systems. This tool runs as an applet on JAVA Runtime Environment 1.3 or higher. Finite state machines are used to model a system in LTSA. Also, LTSA supports the process algebra notation known as the Finite State Process (FSP).

There are two approaches toward validation using LTSA. One approach is to draw a high level block diagram called the high message sequence chart (hMSC). Each block in the hMSC refers to a sequence diagram known as the bMSC. These diagrams are compiled in the tool to create the machine generated FSP code of each bMSC. The code of each bMSC is then composed together using the GUI interface on LTSA to generate the behavioral model of the system known as the Architecture Model. The Architecture Model is then compared to the Trace Model to determine if there are any implied scenarios. The Trace Model is the model generated from the hMSC. This is the model which the Architecture Model needs to match. If these two models do not match, then the system contains implied scenarios. If implied scenarios exist, then each implied scenario is analyzed thoroughly and determined if it is a positive or negative implied scenario. Once all these have been analyzed then your system design is complete. Then the Animator Box found in the LTSA tool can be used to step through the behavioral model of you system for verification.

The second approach toward validation a system using LTSA involves the animation capability of the tool. This process involves writing the FSP code and XML code to model the behavior of the system and the images in the animation. Once this is complete, the animation can be ran to visually see the behavior of the system.

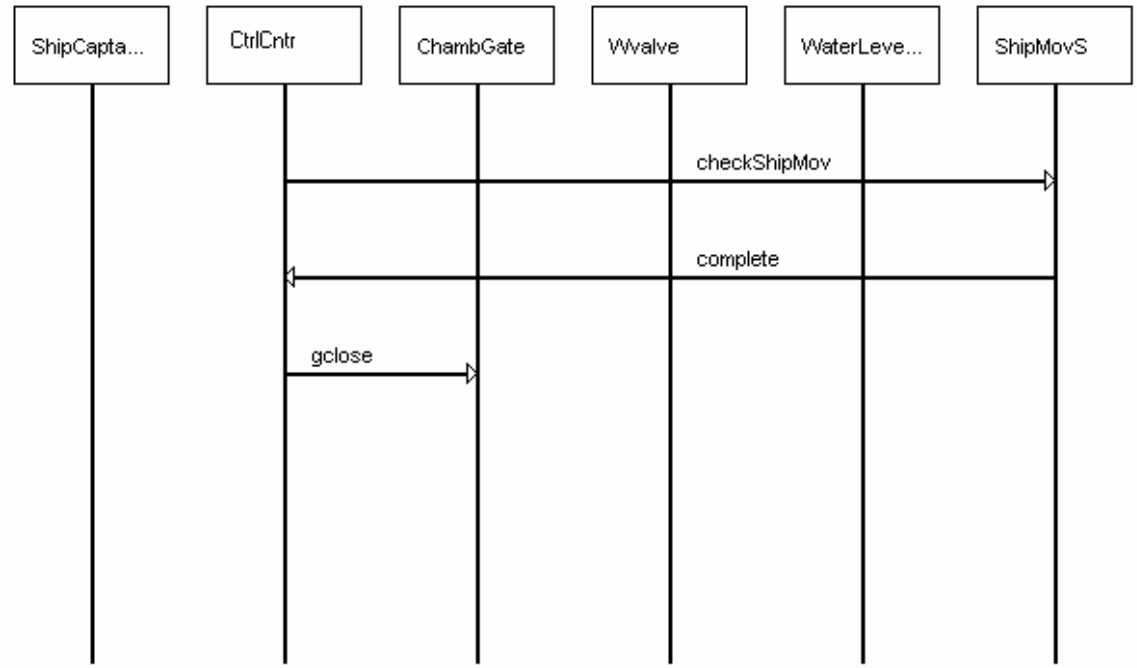
a. SEQUENCE DIAGRAMS (Using LTSA Tool)

High Level Message Sequence Charts

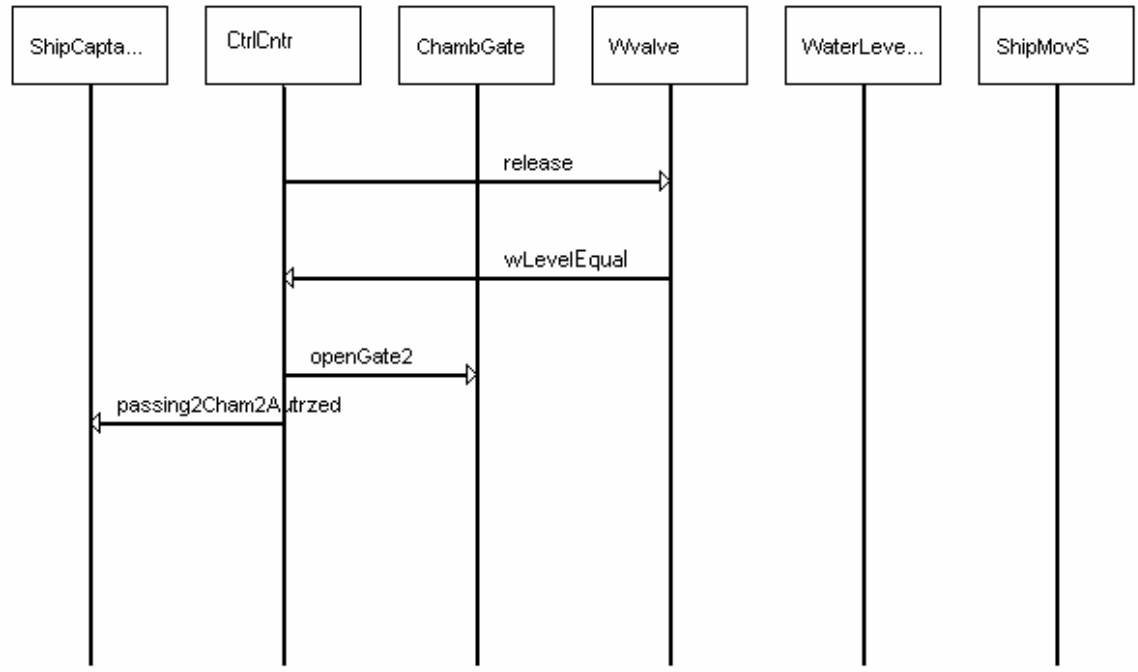


Low Level Message Sequence Charts

Ship Movement Checking

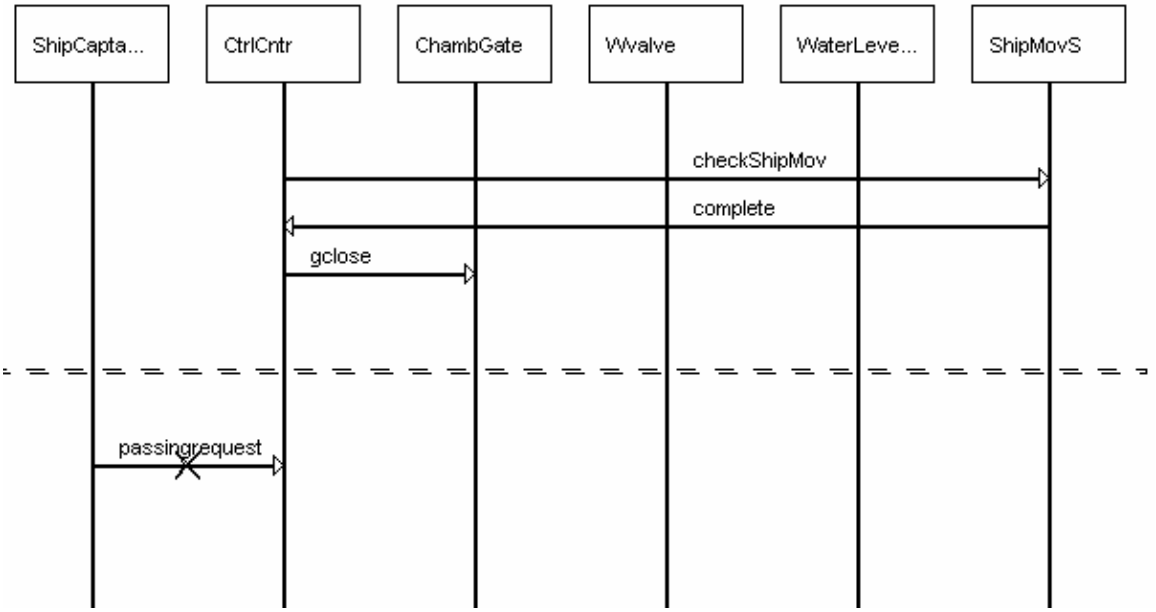


Release Water Valve



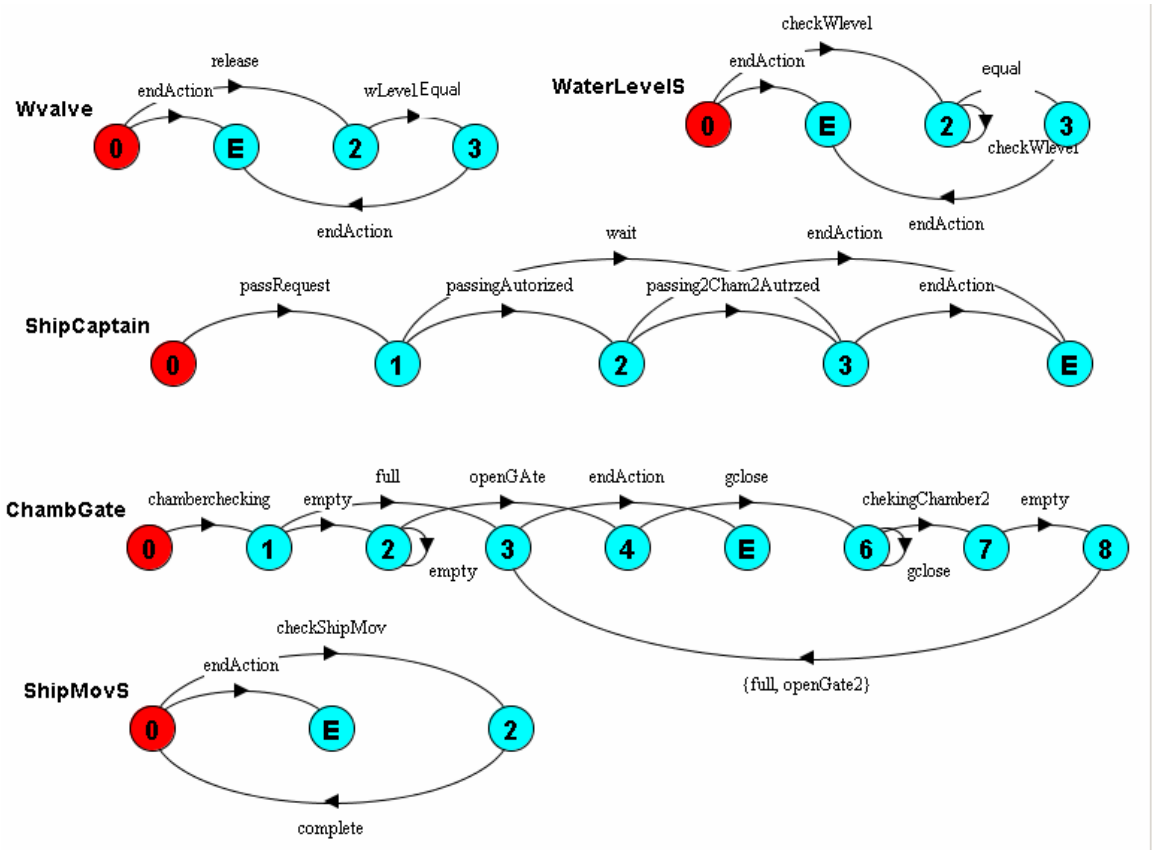
b. Negative Implied Scenario

This sequence chart below shows one implied scenario received using the LTSA Tool. This implied scenario categorized as a negative implied scenario. This is because the Control Center waits until it receives a message from the Ship Captain to request passage. Once the process of this ship is in motion the Ship Captain should not request passage another time. This is because the Control Center would confuse this as a possible request from another ship. Thus, this implied scenario is not a scenario allowed in our system. Please see below for further details.



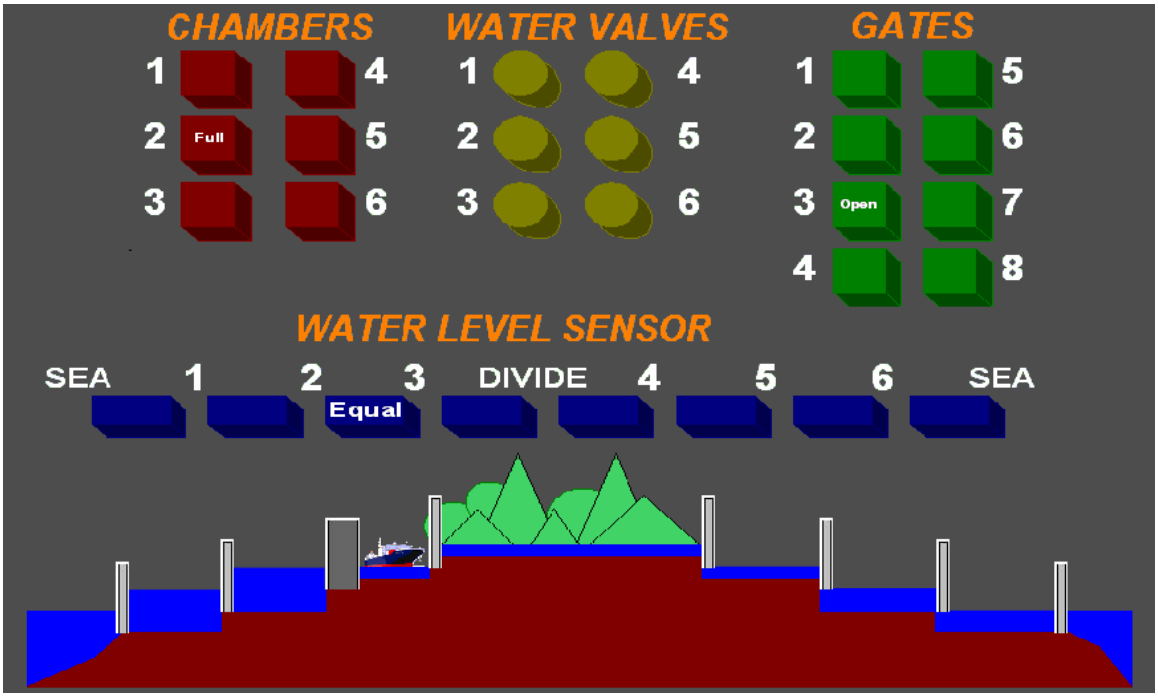
f. STATE DIAGRAMS (Using LTSA Tool)

Here are some of the LTSA FSP bubble diagrams generated from the bMSCs drawn above. Please refer to the LTSA project for more diagrams.



g. VALIDATION AND VERIFICATION (Using LTSA Tool)

Below is a screen dump of the animation using LTSA. The animation shows a sample control panel of a possible control center and a model of the ships movement throughout the canal. In this image, the ship is in the process of moving into the third chamber. Under the *Chambers* status, chamber 2 is classified as full because the ship has not completely left the chamber and the chamber gate is still open. Under the *Gates* status, the third gate is shown as open. Furthermore, the water level between chambers 2 and three are equal indicating that the ship may pass through. At this point in the animation, no water valves are open to change the water level between the chambers.



VI. UPPAAL TOOL

UPPAAL, like LTSA, is a tool used for modeling, validation and verification of real-timed systems. A real-timed system is a system which needs to respond to an input in a definite amount of time. In this design, the canal system can be considered a “soft” real-time system. There is not a definite time limit in which the ship needs to go through the canal. In this design time is introduced in terms of how long it takes for gates to open/close, ships to move completely in a chamber, water valves to open/close, and how long it takes for the water level of a chamber and its adjacent chamber to be equal. UPPAAL main feature is the use of timed automata. Timed automat is a form of modeling using state-transition diagrams with timing constrains. In other works a transition from one state to another cannot occur unless a certain amount of time has progressed. UPPAAL does use the exact concept of time in terms of minutes or seconds, but rather uses clock events that maintain a reference between multiple state diagrams.

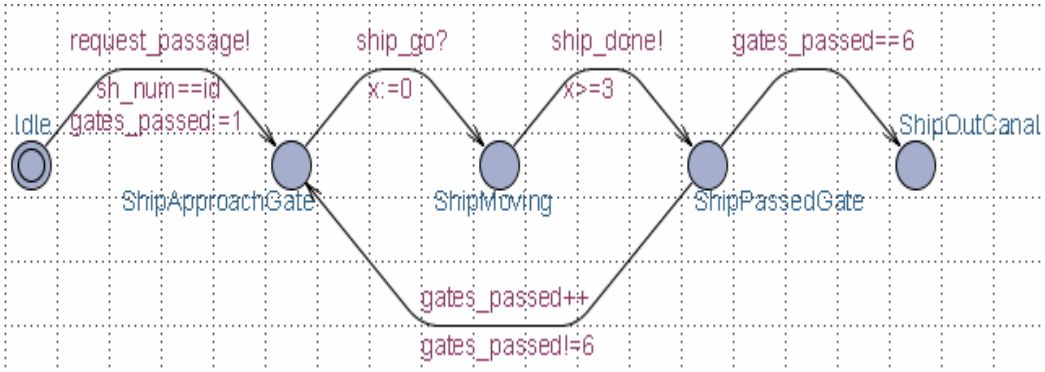
UPPAAL uses syntax very similar to C for the modeling of timed automata. Variables can be defined that are needed in the system, and state diagrams can have parameters just like a function call. There are several forms of transitions defined as guards, invariants, synchronizations, and updates in UPPAAL. Guards and invariants are very similar in that they both allow a transition based on the clock. However, guards are linked to the actual transition where are invariants are linked to states. For example, if there is a guard that states x==5, where x is the clock, then the transition to the next state cannot occur until x=5. On the other hand, an invariant that states x<5 indicates that the current state cannot be left until this inequality is no longer true. One must remain in the state as long as x<5. Syncronizations are used to link between different state diagrams, and updates are where the variables of the state machines can be assigned.

Once all the state diagrams of the system are generated, the design is then compiled for syntax errors. If no syntax errors are found, then the system can be simulated to determine the path of the behavior. The tool will show the transition between every state and the sequence diagram of every transition will be drawn. Once this is complete, then the verification tool can be used to check for deadlock and liveliness in the system.

a. State Diagrams (Using UPPAAL)

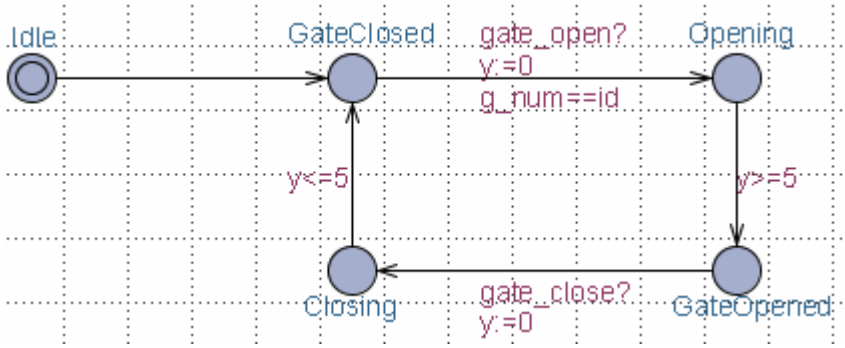
Ship Operation

The ship requests passage as it approaches the canal. Then the ship will continue to pass through 6 gates before it has reached the end of the canal. The clock variable used in this state diagrams is x. The transition between *ShipMoving* and *ShipPassedGate* will not occur until x>=3. This indicates that it takes the ship at least 3 clock events before it is completely in a chamber.



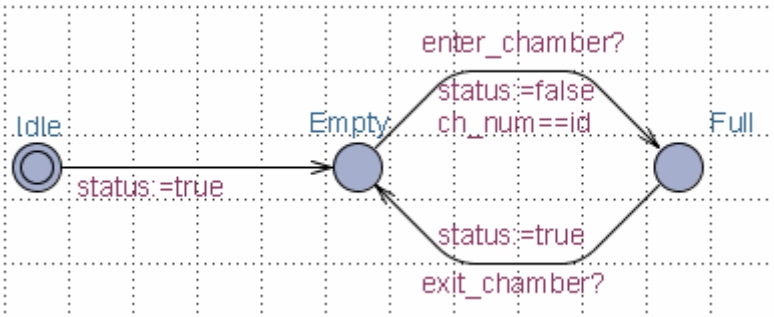
Gate Operation

There are a total of 6 gates in the canal design. Each gate is identified with the g_num variable that classifies the gate id. The clock is defined by variable y. This design shows that the gate will open and close in at most 5 clock events.



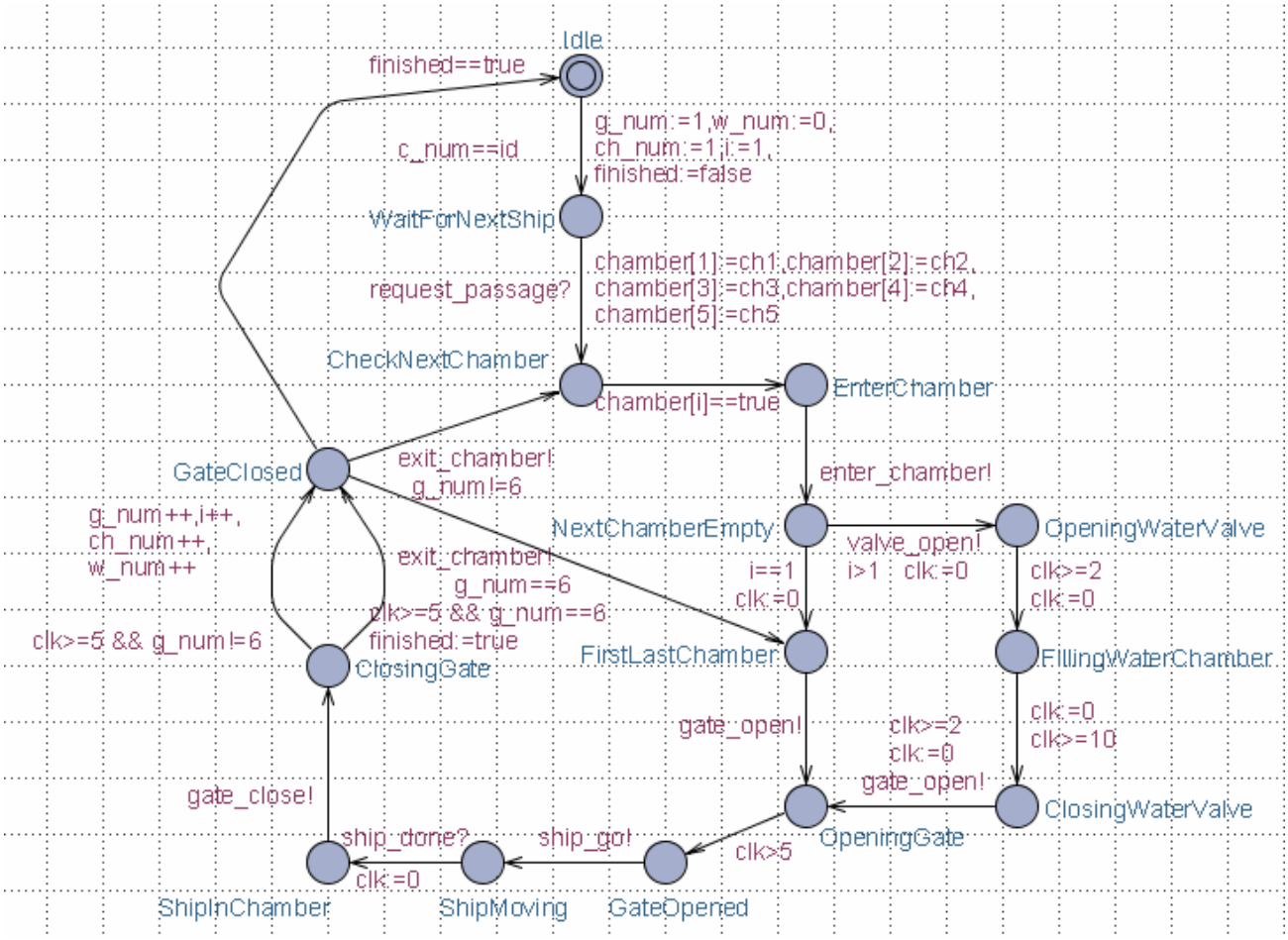
Chamber Operation

There are a total of 5 chambers in the canal design identified through the ch_num or channel number variable. This state diagram does not contain any clocks. Rather, synchronizations are used to trigger transitions between states.



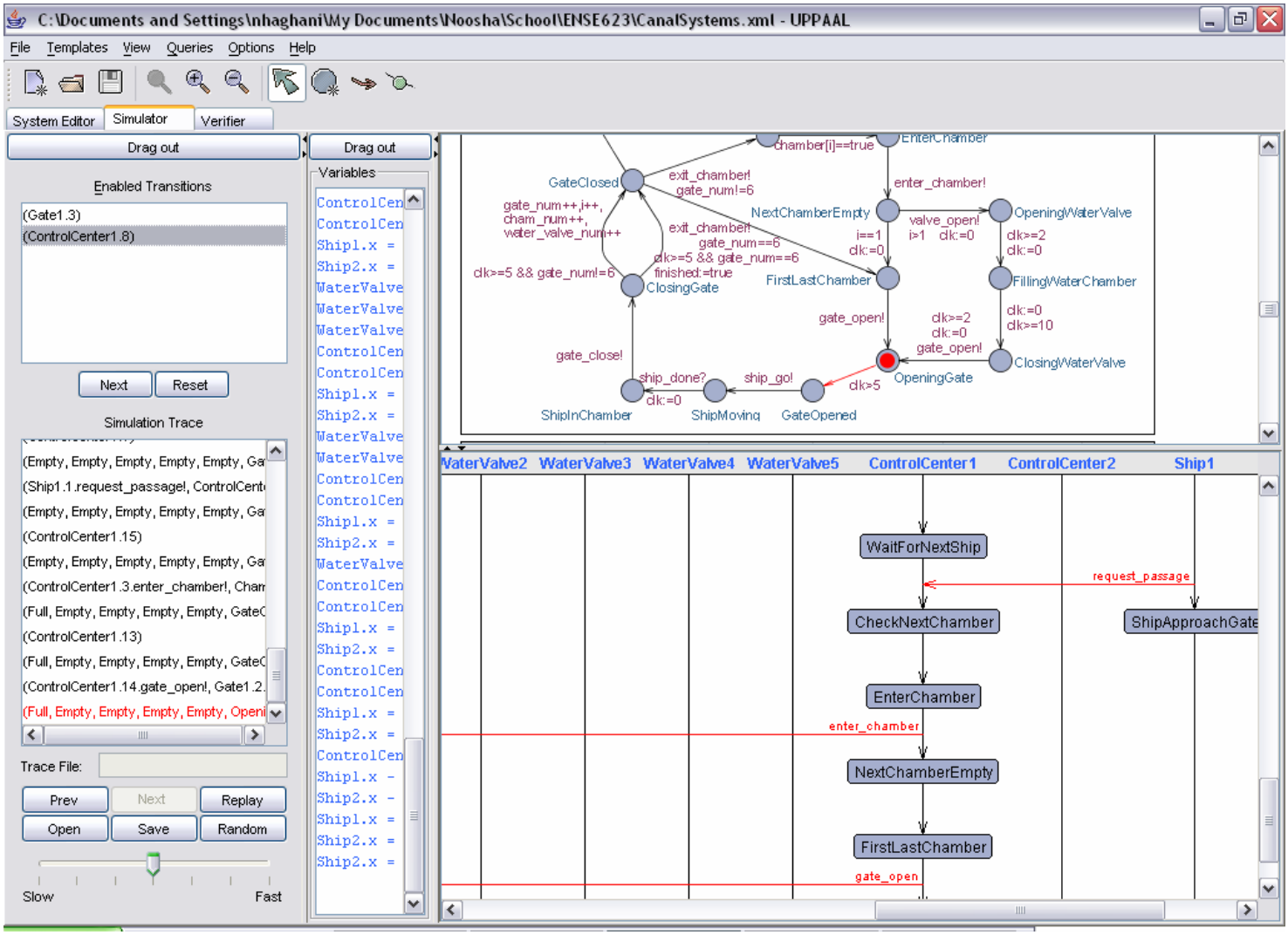
Control Center Operation

This is the state diagram of the control center. After initializations of several variables the Control Center will remain in the state *WaitForNextShip.* Once a request for passage has occurred then the states will cycle through repeatedly until 6 gates have passed. The control center gives the commands (synchronizations) to open and close gates and water valves along with commanding the ship movement.


Idle
finished==true
g_num:=1,w_num:=0,
ch_num:=1,i:=1,
finished:=false
c_num==id
WaitForNextShip
chamber[1]:=ch1,chamber[2]:=ch2,
chamber[3]:=ch3,chamber[4]:=ch4,
chamber[5]:=ch5
request_passage?
CheckNextChamber
chamber[i]==true
EnterChamber
GateClosed
exit_chamber!
g_num!=6
enter_chamber!
g_num++,i++,
ch_num++,
w_num++
NextChamberEmpty
valve_open!
i>1
clk:=0
OpeningWaterValve
clk>=2
clk:=0
i==1
clk:=0
exit_chamber!
g_num==6
clk>=5 && g_num==6
finished:=true
clk>=5 && g_num!=6
ClosingGate
FirstLastChamber
FillingWaterChamber
gate_open!
clk>=2
clk:=0
clk:=0
clk>=10
gate_open!
gate_close!
ClosingWaterValve
OpeningGate
clk>5
ship_done?
ship_go!
clk:=0
ShipInChamber
ShipMoving
GateOpened

b. Simulation (Using UPPAAL)

A screen dump of the simulation of the canal design is below.



VII. CONCLUSIONS

This project modeled the behavior of a canal system from USE Cases to Verification and Validation. The focus of the project remained on the verification of the design. Two tools known and LTSA and UPPAAL were used. Although both tools are used to validate systems, there are many differences between the tools. Both tools use state transition diagrams, but UPPAAL is designed for real-timed systems whereas LTSA focuses only on concurrent systems. UPPAAL also introduces time in its transitions between states. With LTSA one can draw the sequence message charts to obtain the state transition diagrams, and this is reversed with UPPAAL. Both tools have a simulator feature, in LTSA it is the Animator tool and in UPPAAL it's the actual simulation of the state diagrams. Both require the knowledge of a software language; however UPPAAL uses it more towards the definition and assignment of variables, whereas LTSA allows FSP to design the entire system. Overall, both tools can be used for the verification of a system. The constraints of the system, for example, any time constraints, determine which tool is more effective. In this system LTSA is adequate in modeling the system behavior.