SHIP DAMAGE CONTROL ACTION SIMULATION USING HLA

Paulo T. Martins [(a)], Rosaldo J. F. Rossetti [(b)], António Carvalho Brito [(c)],

[(a) (b) (c)] Faculty of Engineering of the University of Porto
[(a)]Centro de Investigação Naval, Portuguese Navy

[(a)]deg11010@fe.up.pt, [(b)] rossetti@fe.up.pt, [(c)]acbrito@fe.up.pt

ABSTRACT
This work discusses on the simulation of a ship crew reaction in controlling damage in a single watertight compartment, taking into account ship's hydrostatics and equipment limitation. Three different simulators were built which exchange data using High Level Architecture concepts, implemented with an open source RTI called CERTI and a MatlabHLA Tollbox.

Keywords: Ship Simulation, High Level Architecture, Discrete Event-Oriented Model, Damaged Ship, Ship Human Factors

1. INTRODUCTION

Large ships are complex "hard systems" that rely on their crew and their interactions with machinery and equipment, to navigate independently through water, while performing the roles for which they were built, such as: fishing, freight transport, entertaining passengers, scientific research, and force projection.

In any case, Human Factors (meaning all issues related to personnel) have large impact on the fulfillment of any vessel operational roles. In particular, personnel movement on board has a major influence on operability, time reaction and efficiency, which in turn is strongly related to the ship's architecture and the knowledge one has (Andrews, Galea, et al. 2008). Nonetheless, crew performance is also affected by the ship's hydrostatic behavior and the way the ship responds to wave excitations (seaworthiness) (Rawson and Tupper 2001, O'Halon and McCauley 1974). In case of an accident, there are other variables that may affect crew's performance, namely smoke spread, ship inclination and many others (Gwyne, Galea, et al. 2003, Gwyne, Galea, et al. 2001, Ginnis, Kostas, et al. 2010).

Ship's operation and survivability in case of damage are also affected by her equipment layout, equipment redundancy, and the paths of the auxiliary systems (electrical cabling, ventilation conduits, water refrigeration pipes, etc). Yet, some of these equipment are vital to the ship's operation and/ or survivability, such as radars, propulsion plants, communication systems, and if broken they must be fixed, whatever the situation, normal operation or damage (Simões-Marques and Pires 2003).

The aim of this work is to present a first approach to crew's performance simulation in case a single watertight compartment is damaged, and analyzing how the ship's architecture may affect it. We are interested in simulating the damage, checking the ship's hydrostatic response in case of flood (inclination), and crew's movement on getting to the damaged area. In fact, these are two very different models and therefore we used High Level Architecture (HLA) concept (Defense Modelling and Simulation Office 1998, IEEE Standard 1516 2010) in building a federation formed up of three different federates: i) damage generation; ii) ship hydrostatic response; iii) damage control teams movement and equipment analysis. To implement it we used an open source Run Time Infrastructure (RTI) called CERTI (Siron, Noulard and Rousselot, 2009) to serve as middleware. To build up federates, we used the MatlabHLA toolbox developed at the Wismar University (Stenzel and Pawletta 2008).

Throughout this paper, we first present the state of the art on the subject of study, and then we describe how the simulation was built following the phases of a simulation project, as much as possible. Finally, some conclusions are drawn and lines for future work are proposed.

2. STATE OF THE ART

Simulation is widely used in ship design to check whether all owner requirements, operational capabilities and international standards have been fulfilled or not.

In opposition, crew and passenger behavior simulations are not that common. Human factors considerations are introduced in new concept designs by applying construction standards, rules derived from experience, designers' common sense, and very specific ergonomic studies of man-machine interface. An exception is made to the study of evacuation routes, since there are standards that oblige passenger ships to verify evacuation behavior (International Maritime Organization MSC Circ 1/ Circ 1238), and several studies were published about the subject in the past (Gwyne, Galea, et al. 2003, Gwyne, Galea, et al. 2001, Ginnis, Kostas, et al. 2010).

To the best of our knowledge, the first study on developing a tool to explore the interaction of personnel movement with the ship design at an early stage was

carried out by the University College London and the Greenwich College (Andrews, Galea, et al. 2008). In this study, ship's compartments, equipment and auxiliary systems (connectivity items) were represented using the design building block approach (Andrews and Dicks 1997, Andrews 2003, Andrews, Burger and Zhang 2005) implemented under the Paramarine-Surfcon software suite (produced by QinetiQ Graphics Research Corporation) and for people movement the Maritime Exodus was used (Gwyne, Galea, et al. 2003). To enable both software to work together an interface toolset built in C++, with several spreadsheets and macros was used.

However, in this work there is no explicit mention to two different problems that we believe affect crew's performance: i) how hull geometry affects ship's seaworthiness and crew's wellbeing (O'Halon and McCauley 1974); ii) how equipment layout and auxiliary systems affect routes, as not only compartment layout may increase crew's workload (Simões-Marques and Pires 2003).

On the subject of the software interface, previous works such as (Stenzel and Pawletta 2006) have used distributed simulation in ship design, applying the High Level Architecture (HLA) concept. HLA evolved from the Aggregate-Level Simulation Protocol (ALSP) and Distributed Interactive Simulations (DIS). The earlier versions of the HLA standard were developed for military applications by the United Sates Department of Defence (Defense Modelling and Simulation Office 1998). Nevertheless, today, HLA is a wide-spread open IEEE standard (IEEE Standard 1516 2010).

In the next sections, both our own approach to analyze crew's performance and the means to implement an HLA instance to simulate and evaluate it are explained and discussed.

3. PROBLEM FORMULATION AND STUDY OBJECTIVES

The main objective of this study is to evaluate a ship's crew performance when a single watertight compartment is damaged, and how it is affected by the ship's architecture and equipment layout. In case of damage the crew carries out several actions in order to limit damage effects and to solve any equipment malfunction, aiming to increase operational capability of the vessel.

Explaining, generically, the problem in hand, when a ship is damaged by flood or fire, she and her crew are affected in several ways, both in her structural integrity, seaworthiness and operational capability. Simplifying the subject, we may consider that in case damage occurs:

- The structural integrity may be affected due to impact, flooding (Vassalos and Jasionowski 2011), or by the effects of fire and high temperature on structure (International Maritime Organization 2010);
- Flood will incline the vessel, affect her ability to float, and her capacity to come back to the upright position when any disturbing moment is applied by wind seas or any other reason (Biran 2003);
- Both previous effects will change her seaworthiness (Gaillard 2011);
- Fire and smoke will spread in case of fire onboard (Gwyne, Galea, et al. 2003, Gwyne, Galea, et al. 2001);
- There will be equipment and auxiliary systems that will be damaged (Simões-Marques and Pires, 2003).
- All above will affect passengers' evacuation and crew performance while trying to reduce the damage effects (Gwyne, Galea, et al. 2003, International Maritime Organization MSC Circ1/ CIRC 1238 2007, International Maritime Organization SOLAS 2009);
- For all of these reasons the ship's capacities to navigate and to proceed with different operations are going to be affected.

All these different issues are very difficult to simulate, and therefore in our first approach to the problem, we are simplifying them even further. At this stage, we are interested in simulating damage of a single compartment and, for each one, finding:

- what is the ship hydrostatic response, i.e. how much she inclines;
- what are the equipment affected by proximity or by means of damage of its electrical power cables;
- how long does a "damage control team" spend to reach the damaged compartment from their normal waiting position.

As for the latter point, we consider that the team selected to reach the incident is the closest one and that it will use the shortest path to reach it. The travelling speed is different while is travelling through a corridor or through stairs (International Maritime Organization MSC Circ1/ CIRC 1238 2007) and it is reduced when the ship inclines due to flooding (Gwyne, Galea, et al. 2003, Ginnis, Kostas, et al. 2010).

All in all, in order to evaluate crew's performance and how the ship's internal layout affects it, which is the ultimate goal of the study, we intend to use three different types of metrics related to the previous points, being the two last ones (2 and 3) used as performance measures of the architecture:

1. ship's draught, hell and trim angle;
2. list of equipment affected;
3. time interval to reach the damaged compartment by a team.

4. MODEL CONCEPTUALIZATION

The nature of the problem in hand suggests the use of a discrete event-oriented model for its simulation, allowing for each time the starting event occurs to verify the evolution over time (dynamic simulation) of the overall system (ship-compartments-crew-equipments-auxiliary systems).

To build the conceptual model, we already know the simulation's objectives and performance measures, and it is easy to identify the starting event (compartment damage). The next step is to identify the components of the system, namely, the other events, activities and entities along with their states and attributes.

We can identify several entities: i) Ship; ii) Compartments; iii) Equipment; iv) Auxiliary systems; v) Damage control teams. All of them, except for the damage control teams, have two states "fully operational" and "damaged", meaning the first does not require repair, and in the second it requires repair. As far as the team is concerned, it can be "waiting" in its initial position, "in action" when moving, combating damage or repairing equipment, and "nonoperational" if the damage occurs in the compartment coincident with their initial position.

The behavior of each entity in the model is understandable, though uncommon in this case is that it is necessary to identify from all entities the ones that change their state in case of a "compartment damage" event or not. While the ship operation capability is always affected, the influence upon the other entities depends upon their relative position in space to the damaged compartment. Other compartments (problem formulation states a single compartment is damaged), as well as equipments that are not in the compartment and whose vital auxiliary systems do not cross the compartment will not be affected. Further, only the damage control team closer to the incident is going to intervene, except in the case its state is "nonoperational".

Undamaged compartments and equipment, as well as the team that is not "in action", do not change their state during the model run and do not contribute for the final evaluation. Therefore, Figure 1 shows the Activity Cycle Diagram that takes only into account the entities that will affect the performance measures.

We can see that after a "compartment damage" event, flood or fire starts to spread, the compartment becomes useless ("damaged" state), the equipment inside is lost and the equipment whose electrical cables go across it also are considered malfunctioning ("damaged" state). The selected damage control team goes towards the incident ("in action" state) and after a while the ship inclines if there is a flood (damaged compartment under the waterline) affecting the team's moving speed. When the team arrives to the damaged compartment, first the fire must be extinguished or the reason for the flood must be repaired. After that, equipment and auxiliary systems may be repaired. Only when all repairs are finished the ship may be considered fully operational once again, and the team can go back to its initial position.

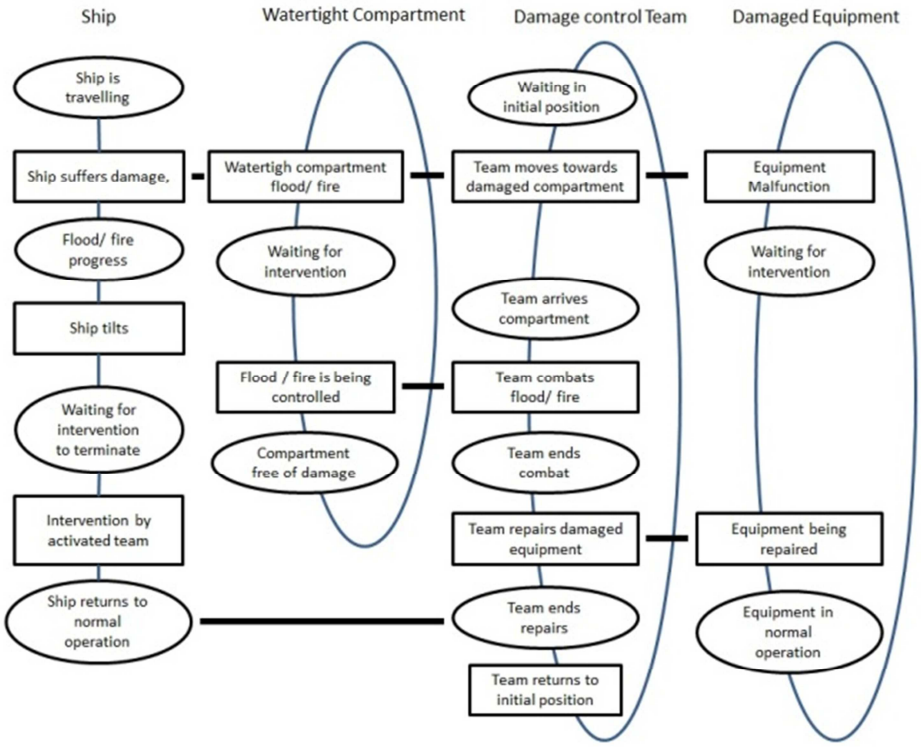

Figure 1: Activity Cycle Diagram

5. MODEL TRANSLATION

First, in order to translate the conceptual model into a computer form, we have to decide which is the best available option. In other words, what the most adequate implementation language to use is, if we may use commercial simulators, and what the most proper software architecture is.

Because the performance measures are quite different from each other, we believe it is best to build different simulators, allowing exchanging data between them, using software architecture such as the "High Level Architecture" (HLA). HLA is based on the concept of federation (collection of federates) and federates (interacting models) that require a middleware, the "Run Time Infrastructure" (RTI), through which data is exchanged and time may be controlled. These concepts are developed further on.

The initial decision on using HLA limited our next selections. Though there are commercial packages that implement HLA, our intention is also to be able to understand and be able to control all software we use. Hence, we use an open-source RTI called CERTI, which was developed by the French Aerospace Lab ONERA (Siron, Noulard and Rousselot 2009), and to build up federates we used Matlab R2010a, making use of the MatlabHLA toolbox (Stenzel and Pawletta 2008, Grades 2001), since previous work has already been done using it. Both run under the Ubuntu operating system (open source operating system built around the Linux kernel) (Ubuntu open source).

In the next paragraphs, we first present the program design options and how entities and their attributes are defined; next each one of the federates is analyzed; and finally we verify how everything comes together checking out the federation and how the HLA works making use of CERTI and the MatlabHLA toolbox.

5.1. Program design

The option of using procedural programming instead of using object-oriented programing is justified due to the fact that defining classes and objects in Matlab is not straightforward and that existing MatlabHLA toolbox work uses this approach, though the toolbox itself requires object definition, since HLA is object-oriented.

Nevertheless, the entities mentioned before have their own attributes and relations, though they do not have their own class operations. The entities are implemented using a structure (struct) that contains other structs. Figure 2, presents the rationale followed to define entities' attributes using a Class Diagram that we entitled Relations Diagram. Though, as mentioned before, for the sake of simplicity this was not exactly the implementation used, serving us just as a conceptual model.

5.1.1. Ship attributes

Attributes incorporate all characteristics that are unique of the ship, such as dimensions, hydrostatic tables and cross curves (variables calculated from the hull geometry) and light ship weight distribution (initial). Also, there are others that may change in case of damage (flood), such as the specific cargo condition, heel, trim and draught.

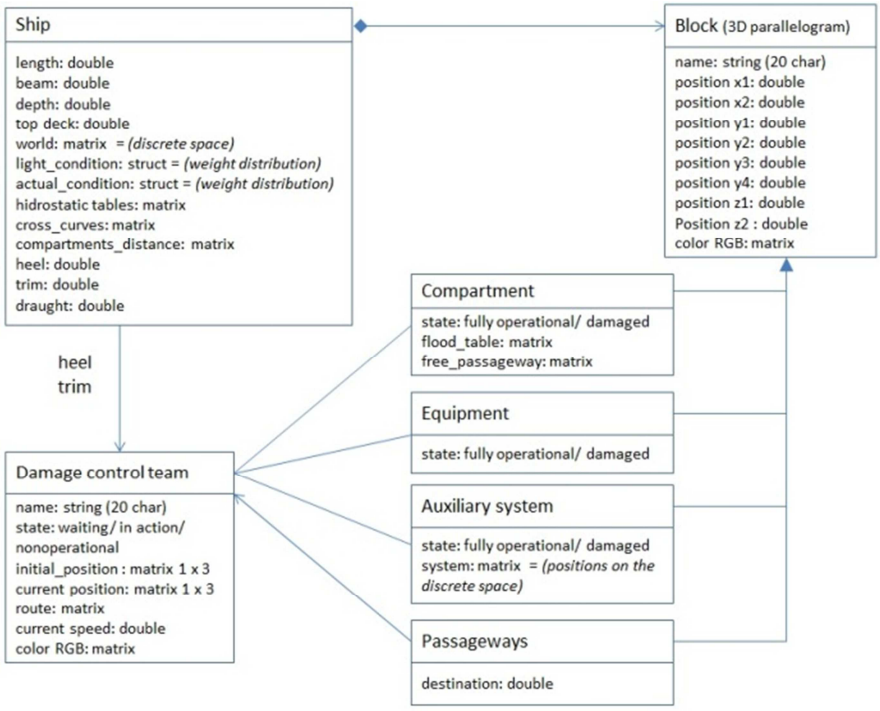

Figure 2: Relation Diagram

There are still another two attributes that are related to how space is defined: a) a large matrix (world) of 0.5 x 0.5 x 0.5 m cells that describes the hole ship using a simple code of cells occupied by numbers indicating equipment, auxiliary system and other entities position, or 0 otherwise; b) a compartments distance matrix that allows for knowing whether there is a passageway between them and that is used to determine the shortest path from one compartment to another using Dijkstra's algorithm (Dijkstra 1959).

5.1.2. Block attributes

This is similar to a superclass, which would be implemented if object-oriented programming were used. This is equivalent to the Block concept in ship design (Andrews and Dicks 1997), which conceives the ship as an arrangement of blocks that can be either compartments or equipment, or any other significant item. All blocks are regular 3D parallelograms defined by their vertices coordinates, and they have their own color which is related to their function (such as technical, rest and service areas).

5.1.3. Compartment attributes

Compartment is derived from Block. It is defined by a state, a matrix that specifies the variables required for flood calculations, and another matrix that specifies internal free corridors for crew movement limiting their occupation (this is going to be used to determine the 0 filled cells in the ship's attribute: *world*).

5.1.4. Equipment attributes

It is also derived from Block that differs from the compartment because it is only defined by its state attribute besides the parent attributes.

5.1.5. Auxiliary systems attributes

It as well derives from Block and is further defined by its state, and by a matrix which provides positioning information. This information is necessary because this is not truly a 3D parallelogram but an addition of 3D parallelograms that form a route between two or more equipment with change on directions in all axes.

5.1.6. Passageways attributes

This derived structure is a 3D parallelogram with dimensions corresponding to only one cell. This is specified within each compartment to represent a door a hatchway or a stair that leads to another compartment, and therefore requires an attribute to identify the compartment it leads to.

5.1.7. Damage Control team attributes

The Team is defined by its name, state and color, as well as its speed and three other attributes related to its positioning: the initial position and position (current position) which are self-explained; and the routing matrix. As far as the route is considered, it stores the coordinates of each step to go from its initial position to its destination (damaged compartment). The speed changes depending it is moving in the horizontal plane or it is going up or down, following IMO's guidance (International Maritime Organization MSC Circ1/ CIRC 1238 2007) and as a function of the ships inclination (Gwyne, Galea, et al. 2003, Ginnis, Kostas, et al. 2010) estimated from ship attributes: heel and trim. This classifier relates to Compartment, Equipment, and Auxiliary System since it may change their state and to Passageway, because it needs its attributes to define its route.

5.2. Ship and Block models

To help defining some of the classifiers default attribute values, it was necessary to build up an application software using Matlab called *DamageHLA*.

The aim of such an application is to, knowing the positioning attributes of the Compartments, Equipments, Passageways and most the ships attributes (dimensions, light condition, tables, and compartment distance), find out the default values for the auxiliary system attribute: system, and the ship attribute: *world*.

DamageHLA allows both to visualize the ship and the blocks, and also to insert all entities into the matrix *world*, which defines the discrete space (0.5 x 0.5 x 0.5 m) that then is used for the damaged teams' movement. The *world* definition and insertion is done following a pre-defined sequence internal to the software:

1. The *world* matrix is all filled with one only obstacle (no clear space);
2. Compartment attributes are read. Within the limits of their positions the cells of the *world* matrix are clear (clear values, i.e. changed to 0);
3. The corridors and free-space within the Compartment are then added, whose data is stored in the *free_passageway* matrix, marking all cells with a different number (e.g. -6000);
4. Passageways are included changing the values of the *world* matrix cells with the value of the destination attribute;
5. Equipment are included, by changing the values of the *world* matrix cells within the 3D parallelogram to each equipment identification number;
6. Auxiliary systems path are defined (*system*) finding out the optimized route between two equipment avoiding obstacles (equipment and free passageways), using Lee maze solving algorithm (Lee 1961).

5.2.1. Auxiliary systems routing

This is made by a standalone routine which implements a modified 3D Lee maze solving algorithm (Lee 1961). This algorithm consist on flooding the discrete space cells with sequential numbers if they are not already filled with an obstacle, in all directions (north, south, east, west, up, down). This is repeated until the goal is reached, beginning than to backtrack to the origin while storing the optimum path positions. The implementation

consists in connecting two pieces of equipment. First, it is necessary to shorten the search space limiting it to the limits of the compartments where the equipments are. Next we define one piece of the equipment as the origin and another as the goal. The path is then defined avoiding other equipment, free passageways and other previously defined auxiliary system. The world matrix cells are marked with the identification number of the destination equipment making possible to identify which equipment is affected when its required auxiliary system is affected.

Figure 3 presents the interface window and Figure 4 two images of the ship representation with and without auxiliary systems representation.

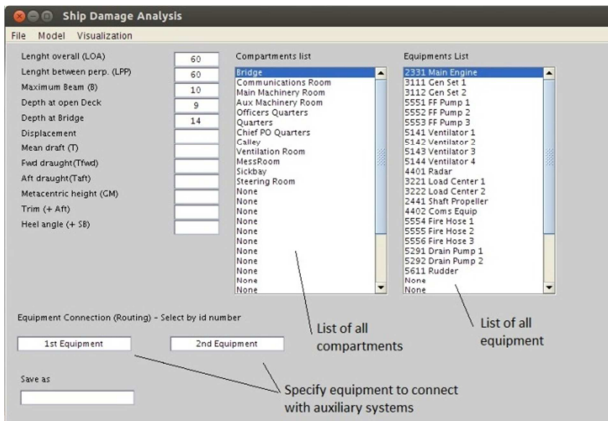

Figure 3: User interface window of *DamageHLA.m*

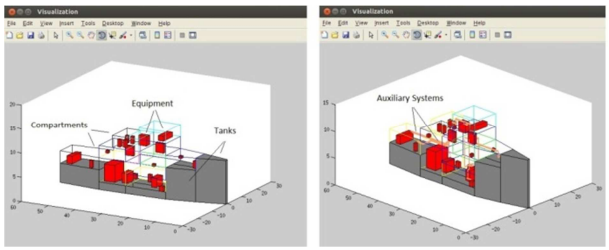

Figure 4: Ship visualization (left) without auxiliary systems; (right) with auxiliary system

5.3. Accident federate

The "*Damage_rt*" federate has the task of originating the starting event (compartment damage), and to control time synchronization with the Run Time infrastructure. The federate communicates with all other federates, that previously declared to receive its attributes every time an event occurs, which may be "hit" and "solved".

Explaining the process, the first event to occur is "solved", which means there is no damage. We identify that after 5 seconds a new event will occur, this time it will be a "hit", meaning that one of the compartments is going to be damaged.

The damaged compartment can either be selected by the user or pseudo-randomly selected by the software, allowing for a continuous cycle of "hit" - "solved" events without user's intervention. After 60 seconds of the initial time of the "hit" event, a "solved" event occurs allowing the user to check the time interval that the damaged control team spent to reach the damaged compartment (performance measure).

Figure 5 shows the interface window of this federate.

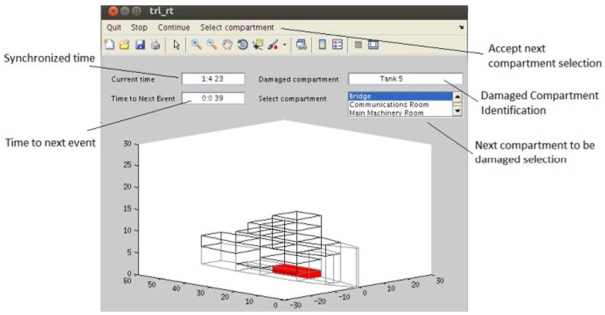

Figure 5: User interface window of the accident federate (*Damage_rt.m*).

5.4. Hydrostatics federate

When the ship is damaged under the waterline, it is most probable that there is a flood (water ingress), and therefore the ship will increase its draught and most probably incline. In case the ship is damaged above the waterline, there will be no flood, though for reasons out of the scope of this work hydrostatics could also change.

What happens can be explained by the added weight approach to damaged stability (Rawson and Tupper 2001, Biran 2003). As water enters the damaged compartment the ships total weight increases, and so her draught also increases. Consequently, even more water enters on board, and the level of fluid will increase, and so on until the level of fluid equalizes with the new draught caused by water ingress.

Additionally, only if the compartment is in the ship transverse and longitudinal center, the flood will not cause a disturbing moment due to the added weight of water. Au contraire, if the compartment is away from the center longitudinally it will produce a trim angle, and if the compartment is away from the transverse axes to port or starboard the ship will heel towards the side of the flood.

The federate uses an external routine to calculate the hydrostatic behavior of the ship after damage which has already been used and validated against commercial software in previous works done by the author (Martins and Lobo 2011). In this case, after the damaged compartment information has been received, the data stored in the values of the compartment attribute: *flood_table* and ship attributes: *light_condition*; *cross_curves*; *hydrostatic_tables* are used to determine the new load condition (ship attribute: *actual_condition*) and the angles of trim and heel. These two last ship attributes are going to be declared and published to all other federates that previously declared to receive them, every time they change. Figure 6 presents the output window of the hydrostatic behavior

of the ship when damaged in compartment "Tank 2" heels the ship.

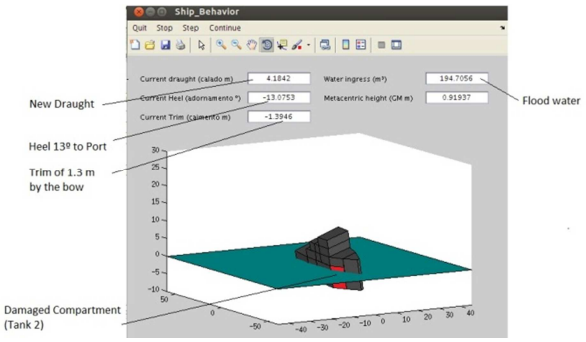

Figure 6: Output window of the Hydrostatics federate (*Actual_ts.m*)

5.5. Damage Control federate

This federate enables the user to gather the performance measures when any of the ship's compartments are damaged; verifying the list of equipment in a "damaged" state and the time the closest Damage Control Team spends to arrive the damaged compartment from its initial position. The identification of the damaged compartment is received from the accident federate, and the values of the trim and heel after damage are received from the hydrostatics federate.

5.5.1. List of damaged equipment

The damaged equipments are gathered and presented in a list, by making a partial sweep of the world matrix, checking all equipment or their auxiliary systems identifications in the cells. The reduction of the search space is done for the particular damaged compartment whose identification has been obtained from the damage federate, in a similar form as in the *DamageHLA* case and the sweep has no specificity.

5.5.2. Damage Control team movement

The movement of the team raises two distinct questions: the first is what the team's route is; and the second at what speed is the team moving. While the teams are in motion, the teams change their state from "waiting" to "in action".

As far as the first issue is considered, all teams' routes across compartments are estimated using the Dijkstra's algorithm (Dijkstra 1959) applied to the ship attribute: *compartment_distances*. In this case, the starting point for each team is its *initial_position* and its goal is the damaged compartment. The team with the lowest "cost" is the selected one. However this does not specify the path within the compartments. In order to do this, we used the Lee algorithm (Lee 1961) to cross from passageway to passageway avoiding any obstacle. The implemented procedure is:

1. The route across compartments is estimated for all teams using Dijkstra algorithm applied to the *compartment_distances* matrix;
2. The team with the lowest "cost" is selected;
3. To find the path within each compartment, the search space matrix used is a reduction of the world matrix, limited to the compartment boundaries;
4. Within each one of these reduced matrices the passageways from the previous compartment and the next compartment in the route matrix are identified;
5. The shortest path within the compartment is found using Lee's algorithm taking the passageways as the start and goal points, avoiding any obstacle.

The movement speed is another relevant issue. IMO distinguishes velocity values for passenger evacuation while walking in the same deck, going down stairs and going up (International Maritime Organization MSC Circ1/ CIRC 1238 2007). Further we assumed a 5% reduction value for each heel angle, taking into consideration there are measures that specify a reduction of 35% in speed with a 7% deck inclination (Ginnis, Kostas, et al. 2010), nevertheless it is our believe that not only the inclination should be considered but also the roll period of the ship from port to starboard due to the seasickness state it can cause (O'Halon and McCauley 1974).

Some implementation particularities are:

- we consider no equipment or auxiliary systems inside tanks;
- we assume that if any tank is damaged then the team will go to the nearest compartment;
- if the damaged compartment is the team's initial position, then its state becomes "nonoperational" and therefore another team must come to rescue.

Figure 7 shows the output window where the list of damaged equipment can be seen and the motion of the damage control team can be followed.

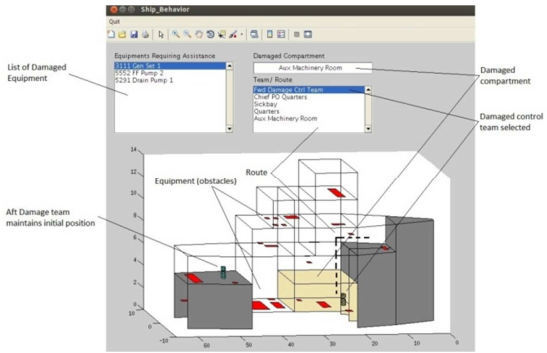

Figure 7: Output window of the Damage Control Action federate (*Observ.m*)

5.6. HLA Implementation

The previous paragraphs discuss on two quite different models (hydrostatics and damage control action) and a third one whose aim is to generate events.

HLA is a software architecture that enables distributed simulation. The basic principle of the HLA, is that several applications, called federates, are brought together to form up a federation, being able to exchange information in accordance to a specification called federation object model (FOM), through a run time infrastructure (RTI) which serves as a middleware and provides common services to the simulation system. The communication between federates and RTI is done trough a RTI ambassador, while communication in the opposite direction is made through a federate ambassador. HLA interface specification to the HLA 1.3 standard includes six management areas (Defense Modelling and Simulation Office 1998, Grades 2001):

1. **Federation Management**: this includes creating/ destroying federations; joining/ resigning federates to/ from federations; federation-wide synchronization; etc.
2. **Declaration Management**: specifies the data that each federate will send and receive by means of publication, subscription and supporting control functions;
3. **Object Management**: provides the means to register and distribute objects, including registration and updates (sender), and discovery and reflection (receiver).
4. **Ownership Management**: since the RTI allows two or more federates to shear the responsibility of updating objects, these procedures manage this information.
5. **Time Management**: time advance policies selection (real time, step-by-step, etc).
6. **Data distribution Management**: provides mechanisms to filter the transmission and reception of undesired data.

Figure 8 presents our HLA implementation using the open-source RTI called CERTI, and the MatlabHLA Toolbox, both compliant with the HLA 1.3 standard (Defense Modelling and Simulation Office 1998). This was done using Ubuntu operating system.

CERTI, is a local distributed system, made of three components: i) the RTI Ambassador; ii) the RTI Gateway; and iii) the library libRTI. The RTIA interacts locally with the federate, and RTIG manages the creation and destruction of federations, as well as, the publication/ subscription of data.

The MatlabHLA Toolbox is used to connect federates to the RTIA in one direction and the Federate Ambassador to federates in the other. As described in reference (Stentzel and Pawletta 2008) the Matlab external interface (MEX) allows access between set of m-functions that provide RTI and federation services and the C++ wrapper function (*rti.cpp*) that provides all necessary conversions.

The three existing m-file federates include the necessary routines for communicating via the RTI. All of them hold federation and declaration management capabilities. As far as the object management capabilities, federates have built in routines that allow for exchange data making use of two different procedures: Send/ Receive Interaction and Update/ Reflect Attribute values. The first is used by the Accident federate (sender) to inform the other two of the event (damaged compartment). The second is used by the Hydrostatic federate that sends regular updates of the attribute values, which are received by the Damage Control Action federate (object management capabilities).

Finally, for the architecture to work, it is only missing the federation declaration and specification. This is done making use of a FED file (FOM) where all management capabilities available to federates and all shared information must be declared.

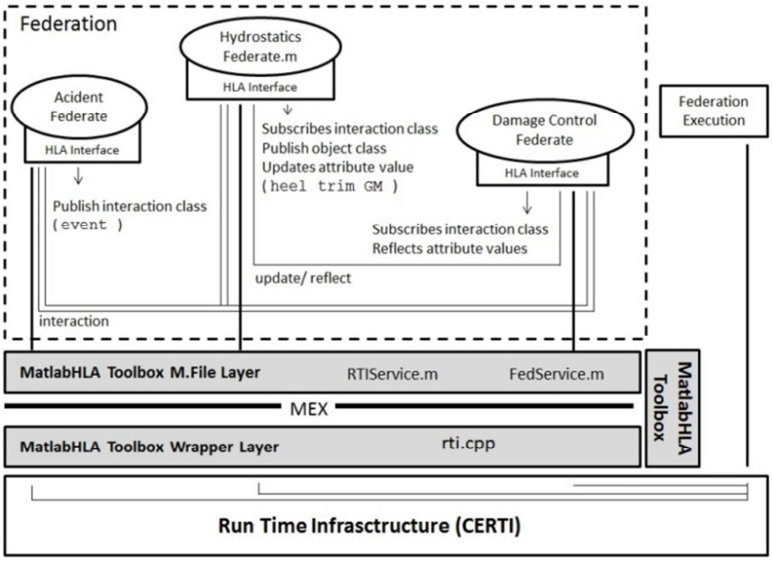

Figure 8: Functional view of simulation under HLA using CERTI and the MatlabHLA toolbox

6. VERIFICATION

The verification of the work has been done running all possible damage compartments one by one, and checking the reasonability of the results.

As far as the hydrostatic response of the ship is concerned, we used routines previously verified and validated against commercial software (Martins and Lobo 2011), and also partially validated against real inclining tests. Similarly, Lee algorithm has also been tested in the past and verified against several limited problems (Martins and Lobo 2009).

The performance measures were verified, checking the list of equipment against the ship and block models (paragraph 5.2), as well as verifying the teams' routes across compartments and if the movement speed would correspond to the direction of movement and to the ship's heel.

7. CALIBRATION

The speed of movement is an issue that would require calibration. As mentioned before, there is no confidence in the data used to reduce speed due to inclination and we could find no reference to the speed reduction due to roll period, though we know, from seakeeping studies that this will be a major factor of seasickness (O'Halon and McCauley 1974) and therefore most probably of speed reduction.

8. CONCLUSION

Through this work we present step-by-step the creation of a simulator that aims to analyze crews' performance and ship interoperability when she suffers a single water-tight compartment damage.

The conceptual model was built assuming that any damage would affect the ship's operational capacity. The crew would act to return the ship to her initial status, both restraining the damage impact and solving equipment malfunctions. Yet, the actions of the crew are affected by the inclination the ship caused by flood, if that is the case.

To implement this concept, first we had to create the ship's model building dedicated software to define her in a discrete space, locating compartments, passageways, equipment, and auxiliary systems. In the auxiliary systems case we used Lee's maze solving algorithm to find the optimum route connecting different equipment (Lee 1961).

Next we built three different simulators using Matlab. The first generates events (damages), the second calculates the hydrostatic behavior of the ship, and the third simulates a damage control team moving towards the damaged compartment. The team's route selection was done using Dijkstra's algorithm (Dijkstra 1959) and the motion through each compartment cell-by-cell we used the same Lee's algorithm.

In order to allow communication between the simulators we used the High Level Architecture standard (HLA 1.3 standard) (Defense Modelling and Simulation Office 1998), allowing data exchange with each other through CERTI (open source RTI) (Siron, Noulard and Rousselot, 2009), using the MatlabHLA toolbox as an interface (Stentzel and Pawletta 2008).

9. FUTURE WORK

We expect this work to be part of a larger study about optimum ship design concept over the Pareto frontier, taking into account several concurrent requirements.
HLA is supposed to have a crucial role in this work to allow the designs' evaluation using multiple simulators, different scenarios, while including crew related issues.

REFERENCES

Andrews, D., Dicks, C., 1997. The Building Block Design Methodology Applied to Advanced Naval Ship Design. *Proceedings of the International Maritime Design Conference*, pp. 3–19. June, New Castle University (United Kingdom).

Andrews, D., 2003. A Creative Approach to Ship Architecture. *International Journal of Maritime Engineering*, 145 (3), 229-252.

Andrews, D., Burger, D., Zhang, J., 2005. Design for production using the Building Block approach. *International Journal of Maritime Engineering*, 147.

Andrews, D., Galea, E, Casarosa, Pawling, Deere, Lawrance, 2008. Integrating Personnel Movement Simulation into Preliminary Ship Design. *International Journal of Maritime Engineering*, 150.

Biran, A., 2003. *Ships Hydrostatics and Stability*. 1st ed. Oxford: Butterworth-Heinemann.

Defense Modeling and Simulation Office, 1998. *High Level Architecture Run-time infrastructure Programmer's guide 1.3*, *Version 5*. United States of America: Department of defence.

Dijkstra, E. W., 1959. A note on two problems in connection with graphs. *Numerische Mathematik*, 1: 269–271.

Gaillard, A. C., Wu, G. X., Wrobel, P., 2011. Simulations of motions of a damaged ship in regular waves. *Proceedings of The Damaged Ship Conference*, pp. 115 -121. January 26-27, London (United Kingdom).

Ginnis, A.I., Kostas, K.V., Politis, C.G., Kaklis, P.D., 2010. VELOS: A VR platform for ship-evacuation analysis. *Computer-Aided Design* 42 (11), 1045-1058.

Grades, 2001. *Distributed Simulation Based on the High Level Architecture in Civilian applications*. Thesis (PhD). Madgberg University.

Gwynne, S., Galea, E.R., Lawrance, P. J., Filippidis, L., 2001. Modelling occupant interaction with fire conditions using the building EXODUS Evacuation Model. *Fire Safety Journal*, 36, 327-357.

Gwynne, S., Galea, E.R., Lyster, C., Glen, I., 2003. Analysing the evacuation procedures employed on a Thames passenger boat using the maritime

EXODUS evacuation model. *Fire technology*, 39, 225-246.

IEEE, 2010. *Standard 1516: Standard for Modeling and Simulation High Level Architecture – Framework and Rules*. New Jersey: IEEE Standards Association

International Maritime Organization, 2007. *MSC Circ 1/ Circ 1238, 2007. Guidelines for evacuation analysis for new and existing passengers ships*. London: International Maritime Organization.

International Maritime Organization, 2009. *Consolidated text of the international Convention for the Safety of Life at Sea (SOLAS)*. London: International Maritime Organization.

International Maritime Organization, 2010. *International code for application of fire test procedures (FTP)*. London: International Maritime Organization.

Lee, C. Y., 1961. An Algorithm for Path Connections and Its Applications. *IRE Transactions on Electronic Computers*, EC-10 (3), 364-365.

Martins, T., Lobo V., 2009. A tool for automatic routing of auxiliary circuits in ships. *Proceedings of the 14th Portuguese Conference on Artificial Intelligence*, pp. 77-86. October 12-15, Aveiro (Portugal).

Martins, T., Lobo V., 2011. A GA based decision support tool for stability and structural viability under damage, *Proceedings of The Damaged Ship Conference*, pp. 133 -139. January 26-27, London (United Kingdom).

O'Hanlon, J. F., McCauley, M.E., 1974. Motion sickness incidence as a function of frequency and acceleration of vertical sinusoidal motion. *Aerospace Medicine*, 45 (4), 366-9.

Rawson and Tupper, 2001. *Basic Ship Theory, Volume 2*. 5th ed. Oxford: Butterworth-Heinemann.

Simões-Marques, M, Pires, J., 2003. SINGRAR – A fuzzy distributed expert system to assist command and control activities in naval environment. *European Journal of Operational Research*, 145, 343–362.

Noulard, E, Rousselot, J-Y, Siron, P., 2009. CERTI: An open-source RTI, Why and How?. *Spring Simulation Interoperability Workshop FSIW-2009*. March 27-23, San Diego (United States of America).

Stenzel, C., Pawletta, S., 2006. HLA Applied to military ship design process. *Simulation News Europe*, 16 (2), 51-56.

Stenzel, C., Pawletta, S., 2008. *Matlab ® High Level Architecture Toolbox*. Available from: http://www.mb.hs-wismar.de [accessed 1 May 2012].

Vassalos, D, Jasionowski, 2011. The Damaged Ship – The steepest curve yet. *Proceedings of The Damaged Ship Conference*. January 26-27, London (United Kingdom).

Ubuntu, 2012. *Ubuntu open source official site*. Available from: http://www.ubuntu.com [1 May 2012]

AUTHORS BIOGRAPHY

Paulo Martins began his carrier in 1996 as a Portuguese Navy crew officer on board several ships. After time at sea he enrolled in the MSc in Naval Architecture at University College London finishing by 2002. Returning to Portuguese Navy he spent time in the Directorate of ships where he worked in refits of different kinds of ships. From 2004 he has been working in new vessels' constructions programs and been engaged in navy's research. Currently he is attending the PhD Program in Industrial engineering and management at the Faculty of Engineering of the University of Porto.

Rosaldo Rossetti joined University of Porto in 2006, where he is an Assistant Professor with the Department of Informatics Engineering, and a Research Fellow with the Artificial Intelligence and Computer Science Lab. Dr. Rossetti received the B.Eng.(Hons) degree in Civil Engineering from UFC, in 2005, and both the M.Sc. and the Ph.D. degrees in Computer Science from UFRGS, Brazil, in 1998 and in 2002, respectively. He carried out most of his doctoral research as a full-time Ph.D. research student at Leeds University's Institute for Transport Studies, Leeds, U.K. His areas of interest generally include complex systems analysis, systems optimization, computer simulation, Artificial Intelligence, and multi-agent systems. Dr. Rossetti is an active member of IEEE, ACM, APPIA, and AISTI.

António Carvalho Brito studied mechanical engineering at Universidade do Porto and obtained his degree in 1981, staying in the same institution as Lecturer where he did his M.S. in 1985. He obtained his Ph.D. in Simulation from the School of Management at the University of Cranfield (U.K.) in 1993. Currently he is an Assistant Professor at Universidade do Porto, where he leads research in the field of Simulation and Information systems.