USING AN HLA SIMULATION ENVIRONMENT FOR SAFETY CONCEPT VERIFICATION OF OFFSHORE OPERATIONS

Christoph Läsche
OFFIS – Institute for Information Technology
Escherweg 2
D-26131, Oldenburg, Germany
E-Mail: laesche@offis.de

Volker Gollücke and Axel Hahn
Carl von Ossietzky Universität Oldenburg
Ammerländer Heerstr. 114-118
D-26129, Oldenburg, Germany
E-Mail: {golluecke,hahn}@wi-ol.de

KEYWORDS

Distributed Simulation; Risk Observer; Hazards; Hazard Analysis; HLA; Failures; Physical World Simulator; Offshore; Risk Assessment; Safety Concept; Simulation;

ABSTRACT

One of the effects of the radically changing energy market is the construction of more and more offshore wind turbines. Many new companies with different levels of experience are entering the market to meet the demand of renewable energy. To this end, we introduce a distributed simulative approach for verifying a safety concept for offshore missions to support the involved companies. In this paper, we present our vision of the emerging simulation environment and introduce the components that we currently work on. More precisely, we introduce a Physical World Simulator (PWS) that allows the simulation of the environment, persons, and resources of offshore operations. The simulator can be interconnected to other simulators and to a Failure and Hazard Observer (FHO) which we also present in the paper. The observer tracks the occurrence of hazards and thus checks if they have been considered while creating a safety concept for an offshore mission. We use the High Level Architecture (HLA) as the simulator communication interface for which we developed a helper library which also extends its features while maintaining compatibility with the standard. The simulation framework enables us to automatically verify the safety concept by performing simulation runs and injecting identified failures.

I. INTRODUCTION

With merely twelve years of experience in the commercial installation of offshore wind farms, the industry is still in an early development stage. A huge change towards renewable energy in a short time can only be realized by a large amount of companies constructing multiple facilities concurrently. The development and implementation of the necessary practices and processes is a highly complex task for the new companies entering the market. Recent events have shown that profound assessments of risks are essential to protect personnel as well as the environment.This is why we aim to support the planing and the execution of safe offshore operations for construction and maintenance of offshore wind turbines.

In this paper, we will describe how we plan to verify the completeness of identified hazards in a safety concept using a simulative approach. This is a novel approach as a safety concept is normally manually verified and thus might have not considered all possible failures that lead to a hazard. Our approach is still a work in process, nevertheless we will introduce our vision of the simulation environment which is used to perform the verification. Afterward, we describe the components that have already been developed or are currently under development. We especially describe our *HLA Helper Library*, the used *Physical World Simulator* (PWS), and the *Failure and Hazard observer* (FHO). We conclude with the next steps that are to be performed to realize our vision.

II. RELATED WORK

Numerous works deal with co-simulation and distributed simulation. For example, [Alexander, 2007] addresses a similar problem in his thesis. He wants to analyze hazards when simulating systems of systems. However his approach is focused on machine learning and he does not use distributed simulation. [Vinnem, 2007] addresses the methods of risk assessment currently performed for oil and gas offshore platforms. However, all analyses are performed in a non-model-based manner in the described approaches. [Lemessi et al., 2010] and [Raab et al., 2011] introduce a similar approach for distributed simulation which also includes observers. The observers are not used for safety analysis as in our approach, but for controlling the simulation and for quality measurement.

III. VISION

Our goal is to improve the safety of offshore operations. We want to achieve this by creating a safety concept similar to HSE (Health Safety Environment) plans (cf. [Sobiech et al., 2012] for HSE plans). However, these concepts are planned to contain more information than the HSE plans. In order to make the safety concept verifiable, we want to use a model-based approach that allows to execute and analyze the processes under investigation using simulation.

A. *SCENARIO AND SAFETY CONCEPT CREATION*

A scenario that will be analyzed has to be modeled in beforehand. For this, we want to provide an easy modeling

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

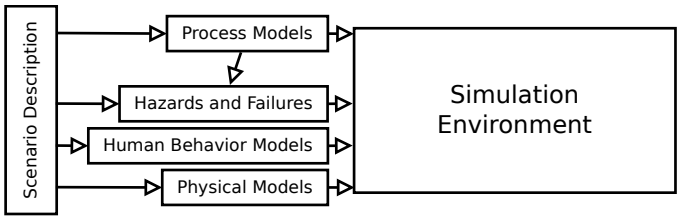

Fig. 1: Inputs for the simulation environment

tool that can be used by maritime experts without extended modeling background. The tool is also intended to be able to assist in creating a safety concept for the scenario.

The safety concept consists, following ISO 26262[ISO, 2011] and IEC 61508[IEC, 2010], of all identified potential hazards and the failures that might lead to these hazards. A hazard is defined by the IEC 61508 as "potential source of harm", a failure as the "termination of the ability of a functional unit to provide a required function or operation of a functional unit in any way other than as required". It has to be ensured that no single failure might lead to a hazard and that the hazards and their impacts are sufficiently considered. This is done by assessing the frequency, consequence, and controllability as described in previous papers ([Droste et al., 2012] and [Läsche et al., 2012].

The novelty in our approach is that we support the verification of the safety concept by using a distributed simulation. This allows, to a certain confidence level, to verify if all failures leading to a hazard have been considered. Formerly, this verification had to take place in a manual manner which is more time-consuming and also prone for oversights.

B. *SIMULATION ENVIRONMENT*

To ensure that all failures for a hazard are considered correctly and no hazard occurs unexpectedly, we want to use a distributed simulation to verify the safety concept. We use several kinds of models that can be found in figure 1. All of them originate from the *Scenario Description* of the offshore operation and are used as inputs for the various simulation participants. The proposed structure of the simulation environment is depicted in figure 2. The central component of

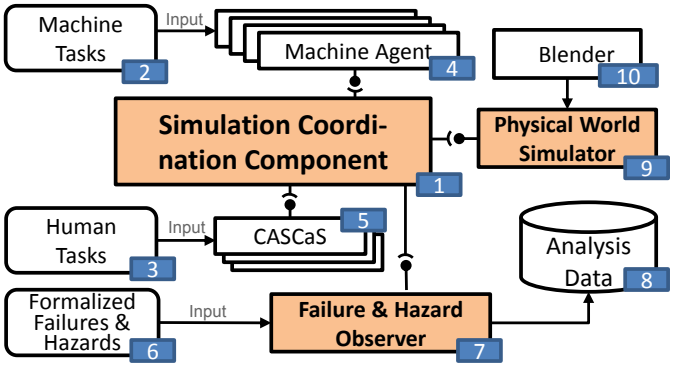

Fig. 2: Structure of our proposed simulation environment

the simulation is the *Simulation Coordination Component* **(1)** which manages the communication between the simulators and the time synchronization among them.

The simulation is controlled by agents. Those are differentiated in *Machine Agents* **(4)** that reflect the behavior of machines and *Human Agents*. Human agents are implemented using the *CASCaS* framework **(5)** (cf. [Lüdtke et al., 2009]) that allows to simulate the non-deterministic behavior of humans. Both execute *Tasks*, the *Machine Agents* execute the *Machine Tasks* **(2)** and *CASCaS* those that are performed by human actors **(3)**. The PWS **(9)** is used to represent the environment and its physics. Further simulators for physics for a special component might be added to the simulation environment if the PWS is not adequate enough for displaying its behavior. A certain kind of this is the *CASCaS* framework **(5)** (cf. [Lüdtke et al., 2009]) that allows to simulate the non-deterministic behavior of humans. The *Blender* tool **(10)** is used to model simulated *Objects* and to design the rough scenario environment.

To determine if the safety concept is correct, the *Failure and Hazard Observer* **(7)** is used. It uses *Formalized Failures and Hazards* **(6)** that originate from the identified failures and hazards of the safety concept. The output of the observer **(8)** is used as a verification of the concept and also as a source for further concept input, e.g. if there is a failure that has not been considered.

C. *CASE STUDY*

We want to use our approach in a scenario in which a *Cargo* is transported on a *Ship Deck* and followed by a *Cargo Supervisor*. It is designed to cover a test case for all components of the simulation environment. It addresses human and environmental behavior, the execution of an operation plan, and hazards and failures that occur during the execution.

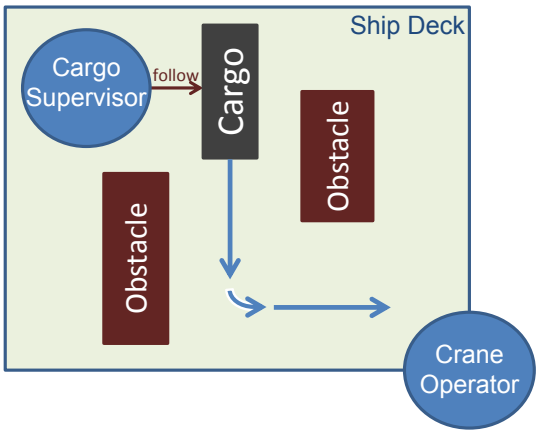

Fig. 3: Overview of the Case Study

Figure 3 outlines the scenario. The *Cargo* is transported along the path marked by arrows whereas the *Cargo Supervisor* follows the *Cargo* to be able to observe it, keeping a distance of at least two meters. The obstacles on the ship deck might cut off the line of sight between the *Cargo Supervisor* and the *Crane Operator*. Further, they might hinder the *Cargo Supervisor* to maintain sufficient distance to the *Cargo*. A failure that has to be detected is a distance lower than two

meters between the *Cargo Supervisor* to the *Cargo*. As the distance drops, the failure has to be detected and indicated.

The case study is a first approach of evaluating our concept and will be extended to a larger one in the future.

IV. IMPLEMENTATION STATUS

This section gives an overview of the current concepts and implementation status of our simulation environment. We already have developed a way to identify potential failures and hazards of a scenario using a Generic Hazard List as described in our previous works ([Droste et al., 2012] and [Läsche et al., 2012]). By the means of this list, we are able to create a preliminary safety concept which contains all potential hazards and the failures that might lead to the hazards. It is created by an offshore safety expert and contains a quantitative rating for each hazard. We also have introduced an approach for modeling scenarios in a previous paper [Sobiech et al., 2012].

The safety concept is verified using the proposed simulation framework. In the following, we want to introduce which of its components have already been developed.

A. *SIMULATION FRAMEWORK*

In order to exchange objects and messages between the various simulators, a communication interface had to be realized. There exist numerous communication frameworks, like *ZeroC IceStorm*[1], but many of these frameworks do not support clock synchronization. Therefore, we chose to use the *High Level Architecture* (HLA) which besides communication using objects and messages also supports timed synchronization of a simulation. Other standards for this purpose are, for example, DIS or TENA (cf. [Henninger et al., 2008]). However, DIS is a limited and superseded standard. TENA offers much of the same capability as HLA but is not as widely spread and supported. We choose to use HLA because of its wide distribution as well as of our previous good experience using it.

1) ***HIGH LEVEL ARCHITECTURE****:* HLA is a standard for implementing a simulation framework that allows the exchange of object instances and interaction messages. The communication takes place over a central component, the *Run-Time Infrastructure* (RTI). All simulators, called *Federates* in HLA vocabulary, register to the RTI. They communicate which objects and interactions they want to update and for which they want to be notified about updates. To support this, a common definition of all object classes and interactions exists.The base for this is called *Object Model Template* (OMT) and is used as *Federation Object Model* (FOM) by the RTI and as *Simulation Object Model* (SOM) by all Federates.

A simulation environment consisting of multiple Federates and an RTI is called *Federation* in the HLA terms. Every Federate in the Federation has an own SOM. The SOM specifies the entities relevant for the single Federate. By this, the Federate can indicate for which instance changes it wants to receive notifications and for which it provides attribute updates itself. The RTI has an aggregated version of all SOMs, the FOM. By this, the RTI knows about all communicated entities and thus can coordinate the communication by providing unambiguous handles for each of them and their instances.

As HLA is just a specification, no reference implementation exists. There are several implementations, both commercial and non-commercial ones. We chose the open source implementation *CERTI*[2] as we already had some experience with it and we found in previous works that its performance is similar to that of a commercial RTI implementation (cf. [Puch et al., 2012]). Thus, there is no significant speed or reliability impact when using the open source implementation.

2) ***COMMUNICATING VIA HLA****:* Communication in the HLA takes place via instances of *Objects* and *Interactions*. *Object Instances* can be used to update the representation of actors or resources within the simulation, i.e. they are used for persistent entities. In contrast, *Interactions* are used for communicating temporary information, like for example an instruction for a simulator.

The reception of data takes place using callback functions. If a Federate wants to be informed about *Object Instance* updates, it subscribes to the *Object Class*. From now on, all changes of *Object Instances* of the *Object Class* performed by any Federate result in a function call at the subscribing Federate. The update might include the time stamp of the update. Similar applies to *Interactions*. If a Federate has updated an *Object Instance* in its simulation, the new attributes have to be manually communicated by invoking a function of the HLA library and thus transmitting the new attributes to the RTI which manages the callback invocations of all subscribing Federates. After all *Object Instances* are transmitted and the simulator proceeds to the next time step, the new simulation time has to be requested using HLA. The moment all other Federates have reached the time, the step is granted and the simulator may continue with the simulation.

3) ***HLA HELPER LIBRARY****:* As mentioned before HLA does not have a reference implementation, it is just a definition that describes how the simulation framework has to be implemented. Three versions of the definition are used (1.3, cf. [U.S. Department of Defense, 1998], 1516-2000, cf. [IEEE, 2000], 1516-2010, cf. [IEEE, 2010]), which are not fully compatible among each other. Actually, even the version 1516-2000 is incompatible among different implementations, as the standard is faulty and two different interpretations of it exist (cf. [Granowetter, 2004]).

This is one of the reasons why we choose to implement a helper library for the HLA implementation. It allows to switch the internally used HLA implementation without a change of the interface to the simulators. Just the library itself has to be adjusted, as well as it might support several implementations that can be exchanged in an easy way. A possible solution for this problem is SimArch (cf. [Gianni et al., 2008]). However, SimArch provides a further abstraction to also use other frameworks than HLA and cannot be used in our software because it is licensed under the terms of the GPL. TrickHLA (cf. [NASA, 2011]) also provides an abstraction from the used HLA implementation but is not freely available and thus could not be used by us.

[1] http://www.zeroc.com/

[2] http://savannah.nongnu.org/projects/certi/

Another reason is the missing representation of relations between *Objects Instances* in the HLA framework. We want to be able to define and update relations between *Object Instances*, for example the distance. Therefore, we added this feature to the library. Internally, we wrapped those relations in special *Object Classes* and let the library interpret them. This allows to maintain full compatibility to Federates that do not use our library. The library is implemented in C++, as the PWS and the FHO are. We also provide a "wrapper for the wrapper" using SWIG[3] to support Java and other programming languages. The concept of callback functions is preserved, although we tried to minimize the amount of different functions performing the same action.

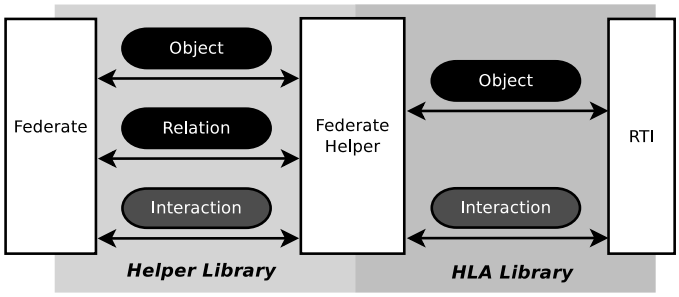

Fig. 4: Interaction between HLA helper library and the specific HLA implementation

Figure 4 depicts the structure of our HLA helper library. Only the library is linked to the Federate, not the specific HLA implementation that is used. Thus, replacing the HLA implementation is easily possible. Further, the library maps the relations to HLA *Objects* which allows the Federate to use relations without having to extend HLA to support them. We also outsourced the time request and grant calls to the library to make the usage of HLA easier. Thus, Federates just have to request a time step. The size of the time step is constant during the simulation and configured in beforehand. Federates are informed as soon as the time is granted.

B. *PHYSICAL WORLD SIMULATOR*

The PWS is used to provide a 3D model of the scenario and of the physical and environmental conditions. Physical effects are, for example, the collision of objects, the buoyancy of the simulated ship or soft body effects which are used to simulate the swinging of a crane rope. Environmental conditions are, in our case, particle effects like rain or snow, blinding by the sun or lamps, or the appearance of fog. This simulator is based on the *GameKit*[4] game engine which contains *Ogre*[5] as visualization and *Bullet*[6] as physics engine. Both components have a huge range of functions and a large, active community. Enhancements can easily be added because of the good documentation and the strongly pronounced object orientated approach. The engine itself is open source software, thus freely customizable. It can load models and scenarios created using Blender[7], an open source 3D content creation

[3]http://www.swig.org/
[4]http://code.google.com/p/gamekit/
[5]http://www.ogre3d.org/
[6]http://bulletphysics.org/wordpress/
[7]http://www.blender.com/

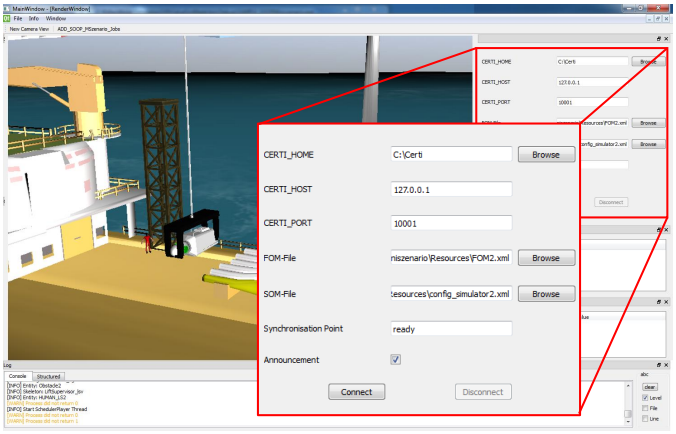

Fig. 5: PWS GUI with marked HLA connection interface

suite, which is another huge advantage, as Blender models of some of the used ships already exist. The PWS has an integrated visualization which is handy for testing purposes, manual observations, or to demonstrate the simulation. But the PWS can also be executed without graphical output to achieve a simulation speed up. A detailed description of the PWS and its components can be found in [Schweigert et al., 2012].

On the basis of the scenario model, our PWS creates the initial configuration for the simulation runs. This includes the initial poses of the used objects, i.e. their position and orientation, as well as the initial environment conditions. There are three general tasks the PWS has to take care of:

- Informing the other simulation participants about object and environmental changes,
- Giving the possibility to control simulation objects and environmental conditions, and
- Injecting failures throughout a simulation run.

For the PWS, the HLA helper library as described in section IV-A3 was fully integrated and was targeted to be an easy usable tool. Thus, the triggering of the attribute updates and the next step calls are performed by the PWS. This was achieved by connecting the PWS scheduler to the HLA helper library update mechanism. The PWS can be connected to any HLA Federation from the simulator interface by entering host, port, and the path to the SOM file (cf. figure 5). We also allow to use a local RTI to test components without requiring the complete simulation environment to be configured and running.

*1) **UPDATING ATTRIBUTES**:* So called *HLASIMElements* are used to update attributes in the PWS. These elements connect HLA Attributes with simulation objects like the mentioned Cargo Supervisor or environmental conditions like wind strength. *HLASIMElements* consist of *single elements* and *related elements*. *Single elements* are used to describe attributes (*HLASIMAttribute*) of a single simulation object or environmental condition. *Single elements* can also be automatically created for every simulation object in the used scenario, in which standard attributes like position, orientation, and the scale factor are observed and updated. *Related elements* are

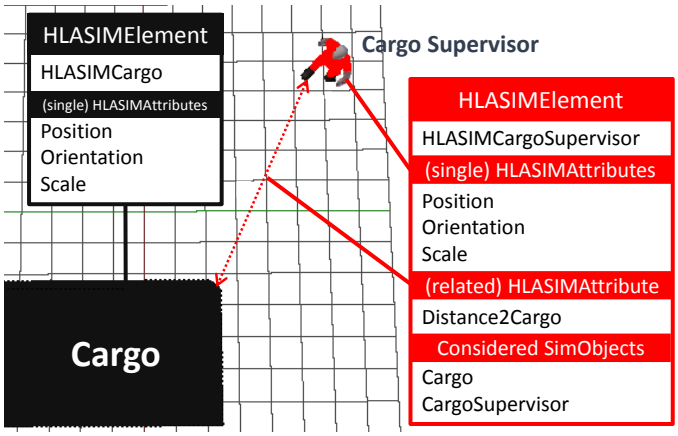

Fig. 6: Example for HLASIMElements with single and related HLASIMAttributes

updated attributes which display relations between two or more simulation objects. An example for this is the distance between two objects in the PWS. In this case, the *related element* observes the two objects and triggers an update of the connected HLA Attribute when one or both objects change their position, orientation, or scale.

An example for *HLASIMElements* as well as single and related *HLASIMAttributes* is shown in figure 6. The Cargo and the Cargo Supervisor have single attributes like the position while the Cargo Supervisor has an additional related attribute which represents the distance to the Cargo. Furthermore, the created *HLASIMElements* are automatically registered at the *RTI* when the simulation starts.

2) ***CONTROLLING THE PHYSICAL WORLD SIMULATOR****:* To control objects within the PWS, like moving to a given position, the *Interaction* concept of the HLA architecture is used. The PWS provides *HLASIMInteractions* which consist of *standard interactions* and *own interactions*. *Standard interactions* are already present in the simulator and comprise the possibilities to translate, rotate, and scale every controllable simulation object. *Own interactions* can be extended *standard interactions*, like moving to a waypoint, or completely new integrated operations. The *HLASIMInteractions* are implemented as PWS jobs which can be started at any given simulation time and have to return a `done` statement when the *Interaction* has been processed. In the case of a *Move to waypoint Interaction*, the pose of the object is interpolated between the start and target pose while looking at the speed or duration parameters of the *Interaction*.

The controlling of environmental conditions, like the maximum wave height, the fog density, or the sun position, is done by listening for *Interaction* calls which change the simulation properties to the received parameters from the responsible simulation participant.

The PWS uses an extension of the controlling solution to give the possibility to inject manual or automatic failures into a simulation run. The difference to the `normal` controlling is that *Failure Injection Components* are attributes connected to hazards or failures which can be defined in beforehand. This can for example be the appearance of fog to disturb the vision or an obstacle at a given position to block an usually used path of the Cargo Supervisor.

The *Failure Injection Components* are visible as special GUI parts of the PWS for a manual injection as well as properties which can be set by a separated control unit to allow automatic injections in the later implementation phase. Failure Injection Components are also memorized to optimize the running times of a simulation while omitting already tested scenarios.

C. FAILURE AND HAZARD OBSERVER

The simulation has to be observed to determine if a hazard occurs during the simulation of the scenario. We are also interested in all occurring failures, as we want to verify the dependencies of the hazards. This information is used for the verification of the safety concept.

We use an FHO framework which joins the Federation and subscribes to the relevant *Objects* and *Interactions*. To be able to observe hazards and failures, the ones identified in the safety concept have to be formalized. Possible ways of formalization include LTL (Linear Temporal Logic) with past operators (cf. [Latvala et al., 2005] for details on PLTL). The challenge is to create executable FHO clients from the hazards and failures that are described in LTL. For now, the FHO clients are manually coded based on the hazard specification. Another aspect regarding the simulation framework is, that it has to be determined which *Objects*, *Interactions*, and *Relations* are necessary to be observed and thus have to be subscribed to. A quite simple approach is to create one FHO for all FHO clients and to subscribe to all *Objects*, *Interactions*, and *Relations* that are mentioned in the LTL formalizations of the failures and hazards. However, this might render into poor performance. One could think of distributing FHO clients among multiple FHOs in order to minimize the amount of subscribed data for each FHO and thus optimizing the performance. This is a further optimization challenge.

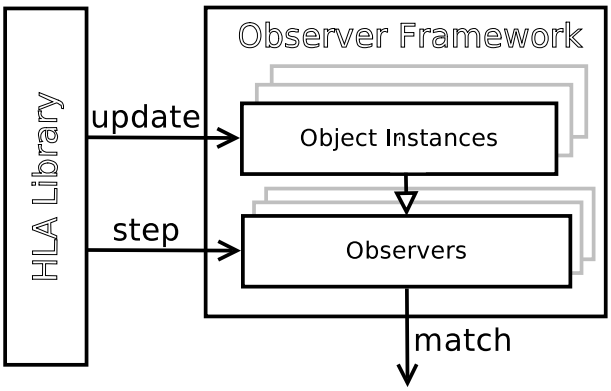

Fig. 7: Outline of the functionality of the Failure and Hazard Observer

Figure 7 outlines the functionality of a possible FHO. It receives updates whenever a simulator updates the simulation *Objects* and stores the relevant *Objects* as well as their history, if this is required. In every simulation step, the FHO clients are evaluated, regarding the *Objects* received in this time step and possibly saved *Objects* from previous steps. If a match is detected, it is indicated by writing to a file, including the trace of all received *Objects* and *Interactions* that lead to the match.

Using the output of the FHO, it is possible to assess if all failures have been correctly assigned to the hazards when creating the safety concept. The safety concept is faulty if a hazard occurs without all of its identified depending failures having occurred. It has to be corrected in this case. To check the safety concept for all identified hazards, all failures have to be injected in all possible combinations during all possible time steps of the simulation. However, there has to be an abstraction of this method because of the feasibility. Again, this is a step that currently is performed manually. But in the future, failures might be automatically injected by taking them from the formalized hazards. The analysis if a hazard has occurred without the required preconditions is also a step manually performed for now. We will try to automate this using the formalized hazards as well.

D. FURTHER SIMULATION COMPONENTS

As described in section III, we also want to include the CASCaS framework for representing human behavior. The Component itself already exists (cf. [Lüdtke et al., 2009]) and will adjusted to be used within the simulation environment. Further information can be found in a previous work [Lenk et al., 2012].

A second component that already exists in a preliminary version is a framework to execute the Machine Agents. Those agents are modeled using the *Business Process Model and Notation* (BPMN2)[8] and define the steps that have to be performed during an operation. The agents will be addressed in a future paper.

V. CONCLUSION AND NEXT STEPS

We have introduced our simulation environment for the verification of a safety concept for offshore operations in this paper. It interconnects several simulators using the standardized HLA for which we added a helper library that allows us to depict relations among actors and resources. The physical simulation is performed in a Physical World Simulator (PWS) which we extended to be usable with HLA. The Failure and Hazards Observer (FHO) allows to determine if a failure or a hazard has occurred.

We successfully implemented the case study as described in section III-C using our simulation concept using the Machine Agents, the PWS, and the FHO. The communication took place over the HLA interface. The Machine Agent sends a commands to the PWS and the PWS executes actions. When an action has been performed, the Machine Agent the next command, and so on. Figure 8 shows the output of the PWS while executing the study. The FHO correctly determined that the distance between Lift Supervisor and Cargo dropped below two meters at the end of the scenario and indicated this. This meas that the case study has been executed as planned and all its components worked as intended. The scenario will be extended by the missing simulators and the already used components will be extended, e.g. by further failures and hazards.

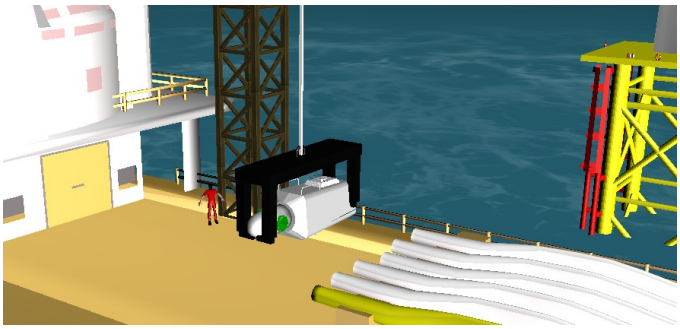

Fig. 8: 3D Output of the Case Study

A next step is to add the CASCaS framework to the simulation environment in order to add human behavior. Further, a common meta model will be developed to create a common understanding of communicated objects between all simulators. A preliminary version already exists.

The next step regarding the observers is to allow the automatic generation of observers. We want to propose a hazard specification language that allows to formalize hazards and failures and automatically generate code out of the formalizations which can be used within the observer framework.

Considerations regarding the automatic injection of possible failures at suitable time steps have to be taken to allow a more systematic and automated investigation of the system. Preparations for this have been taken as described in section IV-B2. A simple, abstracted approach is to randomly inject failures and perform a lot of simulation runs. Of course, this can be further optimized and thus further concepts will be developed.

In the future, we might also use the observers to determine the frequency of occurring hazards. This can for example be done by extending the models by a probability for failures and by performing a lot of simulation runs to gain a confident frequency value.

ACKNOWLEDGMENTS

This work was partially supported by the European Regional Development Fund (ERDF) within the project *S*afe *O*ffshore *OP*erations (SOOP), http://soop.offis.de/.

REFERENCES

[IEEE, 2000] (2000). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules*. The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

[IEEE, 2010] (2010). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules*. The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

[8] http://www.bpmn.org/

[Alexander, 2007] Alexander, R. D. (2007). *Using simulation for systems of systems hazard analysis*. PhD thesis, University of York.

[Droste et al., 2012] Droste, R., Lässche, C., Sobiech, C., Böde, E., and Hahn, A. (2012). Model-Based Risk Assessment Supporting Development of HSE Plans for Safe Offshore Operations. In Stoelinga, M. and Pinger, R., editors, *Formal Methods for Industrial Critical Systems*, volume 7437 of *Lecture Notes in Computer Science*, pages 146–161. Springer.

[Gianni et al., 2008] Gianni, D., D'Ambrogio, A., and Iazeolla, G. (2008). A layered architecture for the model-driven development of distributed simulators. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 61. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Granowetter, 2004] Granowetter, L. (2004). IEEE 1516 Compliance – Will the Real C++ API Please Stand Up? *MÄK Technologies*.

[Henninger et al., 2008] Henninger, A. E., Cutts, D., Loper, M., Lutz, R., Richbourg, R., Saunders, R., and Swenson, S. (2008). Live virtual constructive architecture roadmap (lvcar) final report. Technical report, Institute for Defense Analyses.

[IEC, 2010] IEC (2010). *IEC 61508*. International Electrotechnical Commission .

[ISO, 2011] ISO (2011). *ISO/DIS 26262 - Road vehicles - Functional safety*. International Organization for Standardization.

[Lässche et al., 2012] Lässche, C., Böde, E., and Peikenkamp, T. (2012). A Method for Guided Hazard Identification and Risk Mitigation for Offshore Operations. volume 7612 of *Lecture Notes in Computer Science*, pages 37–48. Springer.

[Latvala et al., 2005] Latvala, T., Biere, A., Heljanko, K., and Junttila, T. (2005). Simple is better: Efficient bounded model checking for past LTL. In *VMCAI. Volume 3385 of LNCS*, pages 380–395. Springer.

[Lemessi et al., 2010] Lemessi, M., Rehn, G., Raab, M., and Schulze, T. (2010). Unterstützungssystem zur Verteilten Simulation. In Zülch, G. and Stock, P., editors, *Integrationsaspekte der Simulation: Technik, Organisation und Personal*, number 14 in Fachtagung der Arbeitsgemeinschaft Simulation; ASIM-Fachtagung "Simulation in Produktion und Logistik", pages 485–492. KIT Scientific Publishing.

[Lenk et al., 2012] Lenk, J. C., Droste, R., Sobiech, C., Lüdtke, A., and Hahn, A. (2012). Towards Cooperative Cognitive Models in Multi-Agent Systems. In *COGNITIVE 2012, The Fourth International Conference on Advanced Cognitive Technologies and Applications*, pages 67–70. ISBN: 978-1-61208-218-9.

[Lüdtke et al., 2009] Lüdtke, A., Weber, L., Osterloh, J.-P., and Wortelen, B. (2009). Modeling Pilot and Driver Behavior for Human Error Simulation. In Duffy, V. G., editor, *Digital Human Modeling. Second International Conference, ICDHM 2009, Held as Part of HCI International 2009*, volume 5620/2009 of *Lecture Notes in Computer Science*, pages 403–412. Springer.

[NASA, 2011] NASA (2011). TrickHLA Framework Facilitates IEEE 1516 Simulation Integration. *http://www.nasa.gov/centers/johnson/techtransfer/technology/MSC-24544-11-trickhla.html*. Last visit: 24. January 2013.

[Puch et al., 2012] Puch, S., Osterloh, J.-P., Fränzle, M., and Lässche, C. (2012). Rapid Virtual–Human–in–the–Loop Simulation with the High Level Architecture. In *Proceedings of Summer Computer Simulation Conference 2012 (SCSC 2012)*, number 10 in Simulation Series Vol, pages 44–50. Bruzzone, A., Curran Associates, Inc.

[Raab et al., 2011] Raab, M., Masik, S., and Schulze, T. (2011). Support System for Distributed HLA Simulations in Industrial Applications. In *Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on*, pages 1 –7.

[Schweigert et al., 2012] Schweigert, S., Droste, R., and Hahn, A. (2012). Multi-Agenten basierte 3D Simulation für die Evaluierung von Offshore Operationen. In *Go-3D*.

[Sobiech et al., 2012] Sobiech, C., Droste, R., Hahn, A., and Korte, H. (2012). Model based Development of Health, Safety, and Environment Plans and Risk Assessment for Offshore Operations. In *MCMC - 9th IFAC Conference on Manoeuvring and Control of Marine Craft*. in print.

[U.S. Department of Defense, 1998] U.S. Department of Defense (1998). *High-Level Architecture Rules, Version 1.3*. The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

[Vinnem, 2007] Vinnem, J. E. (2007). *Offshore Risk Assessment : Principles, Modelling and Applications of QRA Studies*. Springer, London, 2nd edition.

AUTHOR BIOGRAPHIES

Christoph Lässche received his M.Sc. in Computer Science (focus: Embedded Systems and Microrobotics) in 2011 at the University of Oldenburg and started to work at OFFIS in the research group SAV (Safety Analysis and Verification) afterwards. Currently, he is working within the project SOOP (Safe Offshore OPerations), which focuses on risk assessment in the offshore wind sector. is E-Mail address is *laesche@offis.de* and his group can be found at http://www.offis.de/en/start.html.

Volker Gollücke received his M.Sc. in Computer Science in 2012 at the University of Oldenburg and started to work at the University of Oldenburg in the Business Engineering Group afterwards. Currently, he is working within the project SOOP (Safe Offshore OPerations), which focuses on risk assessment in the offshore wind sector. His E-Mail address is: *golluecke@wi-ol.de* and the homepage of his group is http://be.wi-ol.de/.

Axel Hahn is full professor at the University of Oldenburg and leads the working group Business Engineering and board member of the division Transportation at the research institute OFFIS. His research topics are safety and efficiency in marine transportation systems. His E-Mail address is: *hahn@wi-ol.de* and the homepage of his group is http://be.wi-ol.de/.