Automatic Creation of a HLA Simulation Infrastructure for Simulation-Based UI Evaluation in Rapid UI Prototyping Processes

Bertram Wortelen and Christian van Göns

OFFIS – Institute for Information Technology
26121 Oldenburg, Germany
Email: {wortelen|vangoens}@offis.de

***Abstract*—Evaluating user interfaces with virtual user models is a means for rapid prototyping. Setting up a simulation environment for virtual user models often requires high effort due to the heterogeneous simulation tools. Furthermore, the frequent reconfigurations of the simulation due to the rapid changes of the user interface prototypes impose a high amount of workload upon the user. In particular, the manual reconfiguration of the communication between the simulation components is very complex and error prone. Small changes to the user interface often result in changes in the communication of several components. Our solution is the automatic generation of the communication data description for all simulation components. This paper presents the implemented solution and illustrates it with two scenarios from the maritime domain. These scenarios deal with collision avoidance strategies and new concepts for route exchanges between ships and vessel traffic service centres. The automated generation process facilitates handling the emerging changes, which are required in the complex simulation configurations. The evaluation of how well this supports the rapid prototyping process in these scenarios is not addressed in this paper, but is the topic of ongoing research.**

Keywords–rapid prototyping; virtual user models; co-simulation; user interface evaluation.

I. Introduction

The dynamic simulation of virtual user models (VUMs) in interaction with the user interfaces (UIs) is a means to evaluate UI designs. Examples can be found in various domains. In the automotive domain Salvucci [1] uses a virtual driver model to predict the distractive effects of in-vehicle interfaces on the driving performance. Wortelen et al. [2] use a virtual driver model to simulate attention distribution among in-vehicle interfaces and other information sources. In usability research on desktop applications, the Cogtool approach focusses on the prediction of execution times by simulating VUMs [3]. Virtual seafarer models have been used in the maritime domain by Sobiech et al. [4] to test an adaptive bridge system. Lüdtke et al. [5] predicted pilot errors in commercial aircraft cockpits on a virtual pilot model.

The effort to set up a simulation environment for VUMs and UIs can be quite high. Cogtool [3] and ACT-CV [6] are two independent applications that strongly reduce this effort, but are limited to the UIs of desktop application. Both tools enable VUMs created with the cognitive architecture ACT–R to interact with a real UI (ACT-CV) or a UI mock-up (Cogtool) to predict for example task execution times. However, the UI of desktop application is often restricted to one screen, keyboard and mouse. In more complex environments like cars, cockpits, ship bridges or control rooms, it requires far more manual tasks to connect the VUMs to the UIs. Main reasons for this are the very different physical UIs and the heterogeneous tools used for simulation. In order to simulate a realistic scenario in a car or on a ship bridge, a simulation of the environment is required. Another reason can be found in the cooperative nature of many of the above mentioned scenarios, where a task is not executed by just one but multiple human operators, like in aircraft cockpits or on ship bridges. Thus, multiple VUMs might be required.

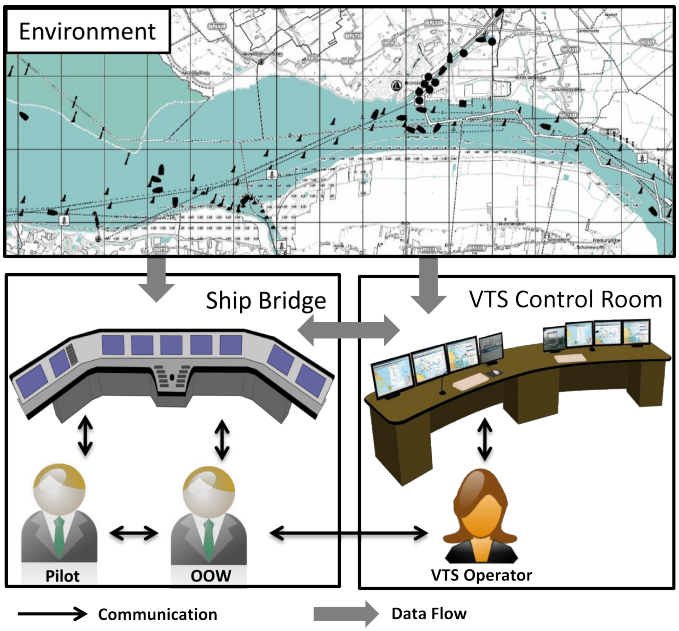

Figure 1. The use case scenario: A VTS operator cooperates with a ship master on a ship bridge, e.g., to inform him about a course conflict and negotiate a new course for the ship.

Puch et al. [7] provide a software framework that aims to reduce the technical challenges of connecting the heterogeneous tools involved in the simulation. They implemented parts of the IEEE 1516 standard on the High Level Architecture (HLA) framework [8], and tailored it to the needs of virtual user model simulations. This paper builds upon their work and shows how parts of their simulation framework can be automatically configured based on a model of a UI prototype.

The work is demonstrated with two case study scenarios from the maritime domain. The involved components can be seen in Figure 1. These scenarios involve a UI system used

 ISBN: 978-1-61208-382-7

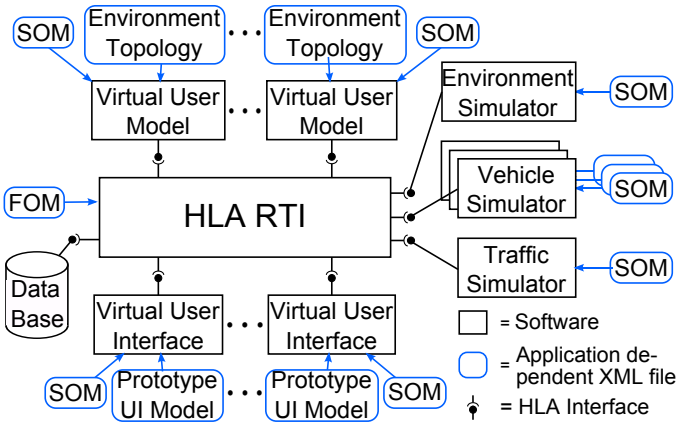

Figure 2. General HLA setup for virtual user simulations in scenarios from the transportation domain.

on a ship bridge operated by two users: The ship master and the pilot. It furthermore contains a Vessel Traffic Service (VTS) control room as a second UI system. A VTS operator is interacting with this UI, while communicating with the pilot and master on the ship bridge.

The remainder of this paper starts with a description of the typical settings for which the presented approach applies and a description of the components involved in the simulations. Afterwards in Section II, it is shown which parts of the simulation configuration can be automatically created. In the same section the algorithm for the automatic creation of the simulation configuration is described. Finally, the approach is illustrated based on maritime case study scenarios.

II. Components of the Simulation

This work builds upon the HLA simulation software provided by Puch et al. [7]. The general setting that is considered can be seen in Figure 2. It shows the different kinds of components that are connected to the HLA simulation.

The core of the simulation is the HLA Runtime Infrastructure (RTI), to which all other components (*federates*) are connected. The HLA standard defines an Object Model Template (OMT), which describes the data that is exchanged between the federates. Each federate defines the data objects it sends or receives in a Simulation Object Model (SOM). The SOM is specified in a dedicated XML file for each federate (see Figure 2). The Federation Object Model (FOM) is a superset of all SOMs and defines all objects that can be exchanged in the simulation.

In cooperative scenarios there can be multiple users interacting with multiple user interfaces. A user interface might even be shared by multiple users, e.g., in aircraft cockpits or on ship bridges. Besides the user and interface simulation the dynamics of the environment is an important aspect of the simulation. In scenarios from the transportation domain, this typically includes the behaviour of the vehicles steered by the user models, the surrounding traffic and further aspects of the environment like weather conditions, road topology or any kind of events.

In the following the data exchange between the environment simulation and the simulation of the UIs is not addressed. It is expected that the data exchange between these components

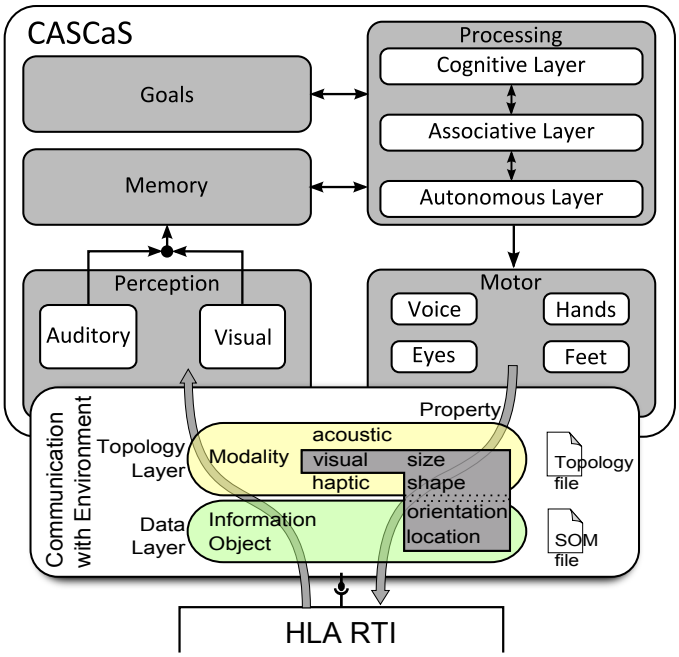

Figure 3. High-Level structure of the Cognitive Architecture CASCaS. At the bottom the two layers of the communication with the environment are shown.

changes infrequently in the scenarios we intend to support. The focus of design is the UI of the workstations that the human operators interact with. Changes in the UI design lead to changes in the communication between the workstation UI and the operator, but not necessarily between the workstation and the environment. In order to analyse multiple UI design proposals with VUMs, it should be easy to change the communication structure between VUM and UI (marked as black arrows in Figure 1).

A. Virtual User Models

There are several ways to create virtual user models. Cognitive Architectures are a good approach to create VUMs, which are capable of simulating human behaviour. A cognitive architecture consists of an integrated set of general models about different aspects of human cognition, like memory or attention processes. These models are defined independently of the task that the user is working on. Human behaviour is shaped by these task-independent aspects. However, the current task is also an important driver of human behaviour.

Cognitive architectures load a formal model of the user's task procedures, typically as a set of production rules. A cognitive architecture interprets and applies these rules based on its internal state and the information that it perceives from the environment. Figure 3 shows the structure of the main components of the Cognitive Architecture for Safety Critical Task Simulation (CASCaS) [9]. This is the architecture used in the case study to create the virtual user models, i. e., cognitive seafarer models. Like many other cognitive architectures it contains components to simulate perceptual and motoric processes. Information perceived from the environment and knowledge stored in a dedicated memory component is processed in an central information processing component. The aim of CASCaS is to simulate goal-directed human behaviour.

 ISBN: 978-1-61208-382-7

Therefore, a goal component is defined that simulates how humans divide their attention between multiple task goals.

In order to interact with the UI, the cognitive architecture needs to know which information objects are displayed on the UI and which actions are available on the UI. Furthermore, the information modality of the information and actions needs to be specified. Is information displayed as text, as graphical icon, as voice message or acoustic signal? Is an action executed on a touch screen, a wheel or lever? This kind of information defines the *environment* that the VUM can interact with.

The data exchange with the environment of a VUM created with CASCaS differentiates between the two aspects of information mentioned above (information objects and information modality). The information modality also includes the 3-dimensional location of the information objects in order to properly simulate eye or hand movements and different visual fields (foveal and peripheral).

The data exchange with the environment is organized in two layers (see bottom of Figure 3). The information itself is transported on the Data Layer, while the modality is defined in the Topology Layer. In the setting described in this paper the technology used for implementing the Data Layer is HLA.

On the Topology Layer, the modality is defined in a separate topology file. Based on the modality further properties can be defined in the topology to specify in more detail, how the information is presented to the VUM, like size and location of an information object. It is differentiated between static and dynamic properties. If the property does not change over time it is statically declared and defined in the topology. If it changes over time, it is declared in the topology, but the actual value of the property is dynamically received via the Data Layer.

This is illustrated on the case study scenario. Consider the current heading of a ship as an information object `heading`. It can be visualized on a display in a text field at a fixed position. In that case the modality properties do not change over time. Alternatively, the heading information can also be visualized on an electronic sea chart as a ship icon that is oriented in the direction of the current heading. The location of the icon moves with the movements of the ship. In that case the location and orientation properties have to be received dynamically via the Data Layer.

Besides the declaration in the topology file all information communicated via the Data Layer needs to be specified in the SOM files to inform the HLA software about the exchange (see top of Figure 2).

B. Virtual User Interface

A VUM does not interact directly with the physical interface. That is to say, it does not move levers or type texts. These interactions are simulated on a virtual user interface (VUI) and forwarded to the underlying control logic of the technical system or to a software mock-up.

A virtual user interface should be tailored to the cognitive architecture that is used for the evaluation of the user interface. Different cognitive architectures are able to interact with different kinds of information. For example, some architectures focus on the simulation of reaction to salient information like a flashing light in its peripheral view [10]. Others are able to use properties of visual objects like colour, orientation or shape to guide visual attention in search tasks [11]. Thus, a VUI only needs to consider those aspects that the cognitive architecture can process in a meaningful way. It mainly defines the level of abstraction of the VUI. Keeping the description language of a VUI simple makes it easier to change a UI prototype during the design phase. This again facilitates rapid evaluation of different design concepts.

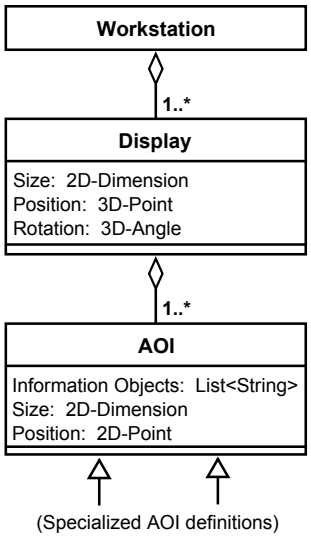

Figure 4. UML model of a workstation. It consists of a set of displays and AOIs located on it. An AOI specifies the information objects it presents.

For the simulation of the virtual user interfaces a framework presented by Sobiech et al. [4] is used. It defines UI models in an XML-based format that only considers the properties size, location and orientation for visually communicated information. Each VUI describes the layout of an entire workstation that may contain multiple displays. For each display multiple areas of interest (AOIs) are defined, which again contain multiple information objects (see Figure 4).

Each information object in an AOI is specified by its name. Within the VUI framework this name is linked to an information source. This source can be a simulated sensor, e.g., from a high-fidelity vehicle simulation or environment simulation, or it can be a real-world data record form a database (see Figure 2). This representation is often sufficient for a large part of the workstation. However, it might be beneficial to represent some AOIs in more detail - especially complex and dynamic AOIs like radar screens or sea charts. In such a case a specialized AOI class needs to be derived from the AOI class (Figure 4), that also defines the simulation of the dynamic properties.

III. Creation of Data and Topology Layer Definitions

Using the HLA software developed by Puch et al. [7], CASCaS has been integrated in simulations in the automotive [12] and maritime [4] domain. The VUI framework presented by Sobiech et al. [4] was used to prototype interfaces on ship bridges. Both tools facilitate setting up a working simulation environment. However, there is still a task that has been shown to be prone to errors and needs to be done repetitively in rapid prototyping scenarios. If a UI prototype is changed and re-evaluated, the information objects exchanged between VUI and

 ISBN: 978-1-61208-382-7

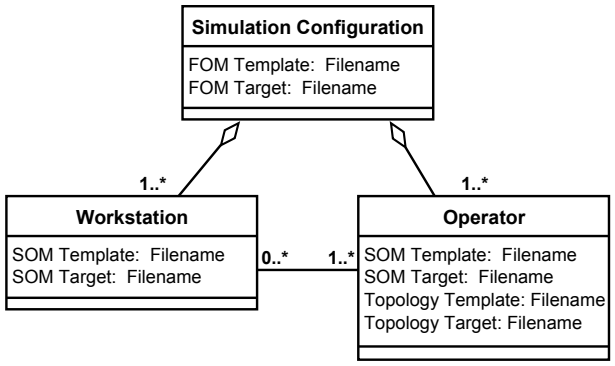

Figure 5. UML model of the simulation configuration.

VUM typically also change. This requires changes in several documents (see Figure 2):

- The SOM definitions of VUI and VUM federates
- The FOM definition of the federation in order to reflect the changes in the SOM definitions
- The topology definitions of the VUMs

Changes in the VUI design typically also lead to changes in the procedure of the VUM, because users interact in different ways with different UI elements. The question, how a change in a VUI design can efficiently be reflected in the VUM, is focus of ongoing research and is not considered in this paper.

Updating the above listed documents is a simple but error prone task. It involves manual edits in several files. Our goal is to support this task and ease the rapid evaluation of different UI designs. The approach taken here is to define all aspects of the entire simulation configuration that are relevant for the communication between VUIs and VUMs in a simple way in a single repository.

The model of the simulation configuration can be seen in Figure 5. All aspects of the communication between VUIs and VUMs in the object model files and topology files can be created automatically from the model of the simulation configuration. Any changes in the configuration lead to an automatic modification of all involved simulation documents.

A configuration consists of one or more operators and workstations. An operator can be assigned to multiple workstations. A workstation can be operated by one or more operators. These relations are defined in the simulation configuration model. The data displayed on the workstation and the commands available on the workstation define, what is exchanged between VUI and VUM. This kind of information is specified for each workstation in its workstation model (Figure 4).

The data exchange between the simulation components is not limited to exchange between VUMs and VUIs. VUIs and VUMs also interact with the simulation of the environment and the technical system controlled by the VUM via the VUI. The object models of this kind of data exchange must also be included in the SOM, FOM and topology definitions. The data exchange with the environment typically does not change when the UI design is changed. Therefore, in rapid prototyping scenarios we do not expect many changes for these object models, if any. We assume that the data exchange between environment and VUI/VUM is already specified in the

$OP \leftarrow$ All operators in the simulation configuration
$WS \leftarrow$ All workstations in the simulation configuration
$ws.aois \leftarrow$ All AOIs in the workstation definition ws.

```
1:  function CreateObjectModelDefinitions(s)
2:      for all ws ∈ WS do
3:          for all aoi ∈ ws.aois do
4:              Insert aoi definition in SOM target of ws;
5:              Insert aoi definition in FOM target;
6:          end for

7:          OP_ws ← List of operators, that have access to ws;

8:          for all op ∈ OP_ws do
9:              for all aoi ∈ ws.aois do
10:                 Insert aoi definition in SOM target of op;
11:                 Insert aoi definition in topology target of op;
12:             end for
13:         end for
14:     end for

15:     for all op_1 ∈ OP do
16:         for all op_2 ∈ OP do
17:             if op_1 ≠ op_2 then
18:                 Insert object models for voice
19:                 communication from op_1 to op_2 in
20:                     1) topology target of op_1
21:                     2) topology target of op_2
22:                     3) SOM target of op_1
23:                     4) SOM target of op_2
24:                     5) FOM target
25:             end if
26:         end for
27:     end for
28: end function
```

Figure 6. Creation of all SOM/FOM and topology files.

topology and SOM/FOM files. Dibbern et al. [13] presented a method that enables the automatic generation of SOM and FOM specifications related to the data exchange to the environment simulation.

The VUI/VUM related object models need to be integrated into the existing topology and SOM/FOM files. The simulation configuration model therefore references two types of files for topologies and SOM/FOM definitions: Templates and Targets (see Figure 5). The template files contain the environment related object models. The target files are created automatically by inserting the VUI/VUM related object models. How this is done is described as pseudo-code in Figure 6.

The algorithm iterates over all workstations defined in the simulation configuration. For all aois defined on any display of a workstation, the information objects communicated via the AOI are defined in the SOM target file of the workstation. The same is done in the target file for the federation object model (FOM). Furthermore, the algorithm iterates over all operators, that have access to the workstation and inserts the definitions for the communicated information objects into the SOM files of the respective operator. It also generates the layout and modality definitions in the topology file of the operator.

 ISBN: 978-1-61208-382-7

```
<Configuration>
  <!-- Workstation definitions -->
  <Workstation name="Bridge">
    <Display filename="conning.xml"/>
    <Display filename="ecdis.xml"/>
    <Display filename="radar.xml"/>
    <Operator name="Master"/>
    <Operator name="Pilot"/>
  </Workstation>
  <Workstation name="VTS">
    <Display filename="vts.xml"/>
    <Operator name="VTS"/>
  </Workstation>
  <!-- Global fom file-->
  <FOM template="TmpFom.xml" target="GenFom.xml"/>
  <!-- Som File for VUI-->
  <SOM template="TmpSomVUI.xml"target="GenSomVUI.xml"/>
  <!-- Operator definitions. -->
  <Operator name="Master">
    <SOM template="TmpSomBridge.xml" target="GenSomM...
    <Topology template="TmpTopBridge.xtop" target="G...
  </Operator>
  <Operator name="Pilot">
    <SOM template="TmpSomBridge.xml" target="GenSomP...
    <Topology template="TmpTopBridge.xtop" target="G...
  </Operator>
  <Operator name="VTS">
    <SOM template="TmpSomVTS.xml" target="GenSomVTS.x...
    <Topology template="TmpTopVTS.xtop" target="GenTo...
  </Operator>
</Configuration>
```

Figure 7. Configuration file with three operators and two workstations.

Finally the communication between each VUM is defined. Users do not only interact with the technical system, but also with other users. Therefore, information objects are defined in the SOM targets of each operator. These information objects describe the information exchange via voice communication between each other operator. In the topology target these information objects are defined as voice actions (speaker) or as acoustic voice information (listener).

IV. Case Study: Maritime Systems

The described approach is tested in the context of eMIR (eMaritime Integrated Reference Platform) [14]. eMIR is a reference platform that provides a means for testing and demonstrating new maritime safety systems or eNavigation systems. eMIR combines a physical test bed in the south east German Bight with a simulation-based software test bed.

For the case study a pure simulation-based setting is used, in which the environment, traffic, the ship dynamics, user interfaces and all operators are simulated. Simulation components for the environment, maritime traffic and the ship dynamics are available in the eMIR simulation test bed. In this section only the VUIs and VUMs are described. Two scenarios are planned in which new user interface concepts will be evaluated.

1. Scenario: **Route exchange.** This scenario deals with a new system for exchanging routes between ships and vessel traffic services (VTS) [15]. It considers the ship master and the VTS operator as human operators, who interact with two different workstations (the bridge system and the VTS system).

2. Scenario: **Collision avoidance.** It deals with cooperative tasks in a collision avoidance scenarios. For this scenario two operators (pilot and ship master) are considered, who interact with one workstation (the bridge system).

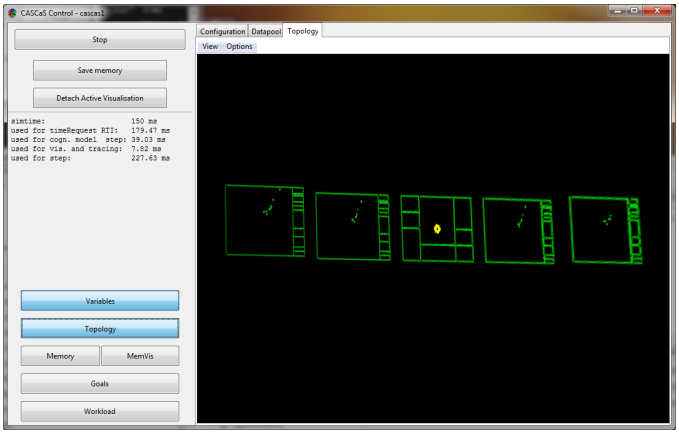

Figure 8. Representation of dynamic topology in CASCaS.

Both scenarios share common components (ship master and bridge system). The automatic generation of the simulation configuration eases the reuse of these components. Even a scenario involving all operators and UIs can be defined. The corresponding configuration file is shown in Figure 7.

All three operators are modelled with the cognitive architecture CASCaS. A formal description of the operators' task procedures is used as input for CASCaS to simulate the operators' behaviour. We aim at building up a knowledge base of these procedures. The current focus is on the procedures of the ship master. The procedures are obtained from different sources. The first step was a literature review. The high level structure of the seafarer's tasks is based on the Operator Function Model (OFM) of navigation tasks defined by Sanquist et al. [16]. The OFM is a framework for representing how a human operator might decompose control functions to meet system objectives and ensure system safety [17]. The OFM is not on a level of detail that allows dynamic simulation. Field observations on ship bridges and VTS control rooms, as well as interviews with the operators, are conducted to gain knowledge about details of certain task procedures. This is work-in-progress.

For the simulation of the UIs of the VTS control room and the ship bridge the VUI framework presented in [4] is used. Our case study scenarios only focus on UI concepts for the support of navigation tasks. Thus, the VUIs do not model the entire bridge system, but only those parts relevant for navigation. This is mainly the Electronic Chart Display and Information System (ECIDS) and the Radar system. The same applies for the operator procedures. These also focus on the relevant navigation tasks.

For the bridge system, Figure 8 shows how CASCaS internally represents the topology layer. What can be seen are the information of the Topology Layer in its current state for all visual information objects of the bridge system. The wireframes show the outlines of all AOIs from all displays defined in the topology of the virtual ship master model in CASCaS. Represented are two ECIDS displays, two Radar displays and one conning display. This topology has been created by the algorithm in Figure 6 based on its VUI model and is used for the virtual model of the pilot and the ship master. The VUI only describes those aspects of the UI, that can be interpreted by CASCaS and are relevant for the investigated tasks.

 ISBN: 978-1-61208-382-7

After all VUMs have been sufficiently defined, the next step will be to evaluate how well the simulation-based approach is able to support rapid prototyping in such complex and safety-critical domains.

V. Conclusion and Future Work

In this paper, a simple algorithm was presented that creates the configuration files for the Data Layer and the Topology Layer required for co-simulations of virtual user models and virtual user interfaces. HLA is used as simulation infrastructure. CASCaS is used as architecture for the virtual user models. The VUI framework presented by Sobiech et al. [4] is used for the virtual user interfaces.

Even though, there is no standard for connecting cognitive models with environment or user interface simulations, the information that needs to be interchanged is essentially the same, regardless of the specific cognitive architecture or user interface description language that is being used. The main difference when using other cognitive architectures like ACT–R [18], or other user interfaces description languages like UsiXML [19] is the representation of this information. Thus we believe that the general approach presented in this paper can also be used in similar settings. However, there definitely are differences between different cognitive architectures and different user interface description languages. Therefore using a different one, would require a re-implementation of the generation algorithm shown in Figure 6.

The automatic generation of the files for the Data and Topology Layer lessens the effort for changing UI concepts in a simulation with virtual user models. This is a prerequisite for efficiently doing rapid UI prototyping. The next steps in our research will be to use this algorithm in actual UI prototyping processes for the described maritime scenarios. We will explore whether it is sufficient for fast and efficient simulation-based evaluation of UIs in rapid prototyping processes.

Furthermore, we believe that our approach also supports the development of normative task procedures in the same way that it supports the development of UIs. New procedure concepts can be evaluated by using the VUMs to simulate the procedures and analysing the simulated behaviour. This is planned to be focus of a subsequent research step.

Acknowledgment

This research has been performed with support from the Cosinus project (http://www.emaritime.de/projects/cosinus/), funded by the *Bundesministerium für Wirtschaft und Technologie (BMWI)* in the framework programme *Next-Generation Maritime Technologies (2011-2015)*, and with support from the EU FP7 project CASCADe, GA N.: 314352. Any contents herein are from the authors and do not necessarily reflect the views of the European Commission.

References

[1] D. D. Salvucci, "Rapid prototyping and evaluation of in-vehicle interfaces," ACM Transactions on Computer-Human Interaction, vol. 16, no. 2, Juni 2009, pp. 9:1–9:33.

[2] B. Wortelen, M. Baumann, and A. Lüdtke, "Dynamic simulation and prediction of drivers' attention distribution," Transportation Research Part F: Traffic Psychology and Behaviour, vol. 21, 2013, pp. 278–294.

[3] B. E. John, "Using predictive human performance models to inspire and support ui design recommendations," in Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 983–986.

[4] C. Sobiech, M. Eilers, C. Denker, A. Lüdtke, P. Allen, G. Randall, and D. Javaux, "Simulation of socio-technical systems for human-centred ship bridge design," in Proceedings of International Conference on Human Factors in Ship Design & Operation 2014. 8-9 Northumberland Street, London: The Royal Institution of Naval Architects, February 2014, pp. 115–122.

[5] A. Lüdtke, J.-P. Osterloh, T. Mioch, F. Rister, and R. Looije, "Cognitive modelling of pilot errors and error recovery in flight management tasks," in Proceedings of the 7th Working Conference on Human Error, Safety and Systems Development Systems Development (HESSD), ser. LNCS 5962, M. W. Jean Vanderdonckt, Philippe Palanque, Ed., vol. 1. Springer, 2009, pp. 54–67.

[6] M. Halbrügge, "Bridging the gap between cognitive models and the outer world," in Grundlagen und Anwendungen der Mensch-Technik-Interaktion. 10. Berliner Werkstatt Mensch-Maschine-Systeme, E. Brandenburg, L. Doria, A. Gross, T. Günzler, and H. Smieszek, Eds. Berlin: Universittsverlag der TU Berlin, October 2013, pp. 205–210.

[7] S. Puch, M. Fränzle, J.-P. Osterloh, and C. Lässche, "Rapid virtual-human-in-the-loop simulation with the high level architecture," in Proceedings of Summer Computer Simulation Conference (SCSC) 2012. Curran Associates, Inc., July 2012, pp. 44–50.

[8] 1516-2010 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), IEEE Computer Society Std., August 2010.

[9] A. Lüdtke, J.-P. Osterloh, L. Weber, and B. Wortelen, "Modeling pilot and driver behavior for human error simulation," in Digital Human Modeling, ser. Lecture Notes in Computer Science, V. G. Duffy, Ed. Springer, Berlin, 2009, vol. 5620/2009, pp. 403–412.

[10] R. Rukšėnas, J. Back, P. Curzon, and A. Blandford, "Verification-guided modelling of salience and cognitive load," Formal Aspects of Computing, vol. 21, no. 6, 2009, pp. 541–569.

[11] J. M. Wolfe, "Guided search 4.0: Current progress with a model of visual search," in Integrated Models of Cognitive Systems, 1st ed., W. D. Gray, Ed. New York: Oxford: Oxford University Press, April 2007, ch. 8, pp. 99–119.

[12] S. Puch, B. Wortelen, M. Fränzle, and T. Peikenkamp, "Evaluation of drivers interaction with assistant systems using criticality driven guided simulation," in Digital Human Modeling and applications in Health, Safety, Ergonomics and Risk Management, ser. LNCS, V. G. Duffy, Ed., vol. 1, no. 8025. Springer, 2013, pp. 108–117.

[13] C. Dibbern, A. Hahn, and S. Schweigert, "Interoperability in co-simulations of maritime systems," in Proceedings of 28th European Conference on Modelling and Simulation, Brescia, Italy, May 2014.

[14] eMIR, "eMaritime Integrated Reference Platform," retrieved: 2015.01.08. [Online]. Available: http://www.emaritime.de

[15] A. Bolles, A. Hahn, M. Braun, S. Rohde, M. Gluch, and K. Benedict, "Cosinus – cooperative vessel guidance for nautical safety," in Proceedings of the International Symposium Information on Ships, ISIS 2014, Hamburg, Germany, 04.-05. September 2014.

[16] T. F. Sanquist, J. D. Lee, and A. M. Rothblum, "Cognitive analysis of navigation tasks: A tool for training assessment and equipment design," U.S. Department of Transportation – United States Coast Guard, Interim Report, April 1994, retrieved: 2015.01.08. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a284392.pdf

[17] E. Palmer and C. M. Mitchell, "Operator function modeling: Cognitive task analysis, modeling and inteligent aiding in supervisory control systems," Center for Human-Machine Systems Research, School of Industrial & Systems Engineering, Georgia Institute of Technology, Tech. Rep., September 1990, NASA Ames Grant NAG 2-413. Retrieved: 2015.01.08. [Online]. Available: https://archive.org/details/nasa_techdoc_19910002067

[18] J. R. Anderson, How can the Human Mind Occur in the Physical Universe, ser. Oxford Series on Cognitive Models and Architectures. Oxford University Press, August 2009.

[19] UsiXML 2010, Proceedings of the 1st International Workshop on USer Interface eXtensible Markup Language, 2010, Retrieved: 2015.01.08. [Online]. Available: http://www.usixml.eu/sites/default/files/FaureVanderdonckt-UsiXML2010.pdf