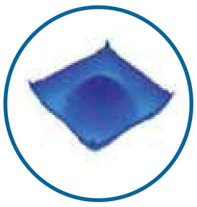
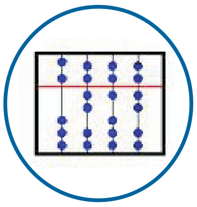
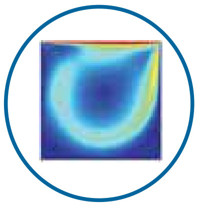
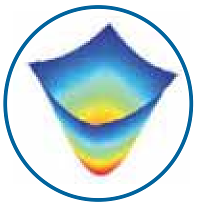
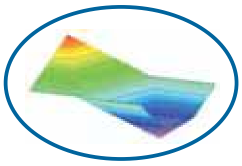
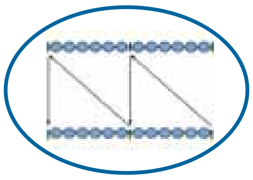
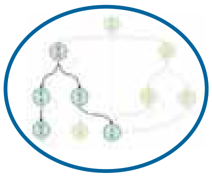
SNE SIMULATION NEWS EUROPE





Special Issue

Parallel and Distributed Simulation Methods and Environments

Volume 16 Number 2

September 2006, ISSN 0929-2268

Journal on Developments and Trends in Modelling and Simulation

Special Issue

ARGESIM

Dear readers,

We are glad to present the first SNE Special Issue - a Special Issue on 'Parallel and Distributed Simulation Methods and Environments'. The idea for special issues was born in ASIM, the German Simulation Society. As there was and as there still is a need for state-of-the-art publications in topics of modelling and simulation, ASIM first tried to publish monographs on this subject. But publication of such books showed disadvantages: too slow production time, too high costs, and lack of publication issues. ASIM, seeking for alternatives, contacted ARGESIM with the idea of SNE Special Issues - while ARGESIM itself thought on Special Issues, because of lack in publication space in the regular SNE issues. Now, one year after the first contact, we can present the first Special Issue, edited by Thorsten & Sven Pawletta from University Wismar, Germany.

The editorial policy of SNE Special Issues is to publish high quality scientific and technical papers concentrating on state-of-the-art and state-of-research in specific modeling and simulation oriented topics in Europe, and interesting papers from the world wide modeling and simulation community. This Special Issue 'Parallel and Distributed Simulation Methods and Environments' (SNE 16/2), will be sent to all ASIM members - together with the regular SNE 16/1 (SNE 46), and sample copies will be sent to other European Simulation Societies; furthermore, it is available on basis of an individual subscription of SNE - SNE Special Issues are open for everybody, for publication and subscription (not only for ASIM). We think also on Special Issues publishing selected papers from EUROSIM conferences.

We hope, you enjoy this Special Issue, which presents state-of-the-art in parallel and distributed simulation, from theory with lookahead formulas via implementation with HLA and other systems to applications in ship desgin and blood flow simulation.

It is planned to publish a SNE Special Issue each year, for 2007 a Special Issue on 'Verification and Validation' (Guest Editor Sigrid Wenzel, University Kassel) is scheduled (SNE 17/2). I would like to thank all people who helped in managing this first Special Issue, especially the Guest Editors, Thorsten and Sven Pawletta from Wismar University.

Felix Breitenecker, Editor-in-Chief SNE; *Felix.Breitenecker@tuwien.ac.*at

Content

SNE Editorial Board

Felix Breitenecker (Editor-in-Chief), Vienna Univ. of Technology, *Felix.Breitenecker@tuwien.ac.at*

Peter Breedveld, University of Twenty, Div. Control Engineering, *P.C.Breedveld@el.utwente.nl*

Francois Cellier, ETH Zurich, Inst. f. Computational Science / University of Arizona, *fcellier@inf.ethz.ch*,

Russell Cheng, Fac. of Mathematics / OR Group, Univ. of Southampton, *rchc@maths.soton.ac.uk*

Rihard Karba, University of Ljubljana, Fac. Electrical Engineering, *rihard.karba@fe.uni-lj.si*

David Murray-Smith, University of Glasgow, Fac. Electrical & Electronical Engineering; *d.murray-smith@elec.gla.ac.uk*

Horst Ecker, Vienna Univ. of Technology. Inst. f. Mechanics, *Horst.Ecker@tuwien.ac.at*

Thomas Schriber, University of Michigan, Business School *schriber@umich.edu*

Sigrid Wenzel, University of Kassel, Inst. f. Production Technique and Logistics, *S.Wenzel@uni-kassel.de*

Guest Editors Special Issue *Parallel and Distributed Simulation Methods and Environments*

Thorsten Pawletta, *pawel@mb.hs-wismar.de*

Sven Pawletta, *s.pawletta@et.hs-wismar.de*

Res. Group Computational Engineering and Automation, Wismar University, 23952 Wismar, Germany

WWW.MB.HS-WISMAR.DE/cea

SNE Contact

SNE-Editors / ARGESIM
c/o Inst. f. Analysis and Scientific Computation
Vienna University of Technology
Wiedner Hauptstrasse 8-10, 1040 Vienna, AUSTRIA
Tel + 43 - 1- 58801-10115 or 11455, Fax - 42098
sne@argesim.org; WWW.ARGESIM.ORG

Editorial Info - Impressum

SNE Simulation News Europe ISSN 1015-8685 (0929-2268).
Scope: Development in modelling and simulation, benchmarks on modelling and simulation, membership info for EUROSIM and Simulation Societies.
Editor-in-Chief: Felix Breitenecker, Inst. f. Analysis and Scientific Computing, Mathematical Modelling and Simulation, , Vienna University of Technology, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; *Felix.Breitenecker@tuwien.ac.at*
Layout: A. Breitenecker, ARGESIM TU Vienna / Linz; Anna.Breitenecker@liwest.at
Printed by: Grafisches Zentrum TU, Wiedner Hauptstr. 8-10, A-1040, Wien
Publisher: ARGESIM and ASIM
ARGESIM, c/o Inst. for Analysis and Scientific Computation, TU Vienna, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria, and
ASIM (German Simulation Society), c/o Wohlfartstr. 21b, 80939München
© ARGESIM / ASIM 2006

HLA Applied to Military Ship Design Process

Christian Stenzel, Sven Pawletta, Wismar University, Germany
christianstenzel@gmx.net, WWW.MB.HS-WISMAR.DE/cea

Richard Ems, Petra Bünning, MTG Marinetechnik GmbH, Hamburg, Germany
{Richard.Ems; Petra.Buenning}@mtg-marinetechnik.de

This article reports on an ongoing project in the field of naval architecture where existing implementations for simulating surface vessels in seaway have to be integrated into an HLA compliant distributed simulation and visualization federation. Like in other engineering fields, existing Fortran codes for extensive numerical problems play an important role in the ship design process. Unfortunately, relevant RTI implementations provide language bindings only for C++ and Java. Therefore, it is currently not straightforward to build HLA federates using existing Fortran codes. The article points out possible coupling variants between an RTI and Fortran code and discusses pros and cons. It is also shown how this research is influenced by experiences from related efforts to provide Matlab/HLA connectivity. Subsequently, the current implementation state of the simulation federation is presented. Finally, the overall development effort is evaluated and possible ways towards complexity reduction are pointed out.

Introduction

In 2002, after successful completion of the *Simulation Based Design and Virtual Prototyping* (SBDVP) program, the *NATO Naval Armaments Group* NG6 has charged the formal sub-group SG61 with establishing standards for modeling and simulation in naval ship acquisition. One of the objectives of this subgroup is the development of the *Virtual Ships* (VS) STANAG ([2], [5]). The VS STANAG is currently in the final draft state and in preparation for ratification. What is technically most important, is that the VS STANAG defines a simulation architecture for virtual ships based on the HLA standard. The standardization activities of the NATO indicate that also in Europe the HLA technology becomes important as strategic market factor for simulation and military suppliers in future.

The MTG Marinetechnik GmbH is an independent center of excellence for planning and designing surface warships and operates already since 1966 primarily for the German Navy. In cooperation with the *Federal Office of Defense Technology and Procurement* (BWB) MTG already deals with the topic Simulation Based Design and Virtual Prototyping for a while. Thus, the computer-aided model VORGES was already conceived for the development and evaluation of ship designs.
The main objective of VORGES is to point out possible realization variants for the future ship design. Furthermore, VORGES should support the selection of a ready to build solution as well as the process of construction, proving, and operation control ([9]). During this ship design process the simulation tools and models utilized have to interoperate.

Thereby, it has to be distinguished between non-runtime and runtime interoperability. *Non-runtime interoperability* can be achieved through use of shared databases, common access to *Product Data Management* (PDM) systems, and data exchange standards ([2]). *Runtime interoperability* has to be achieved via HLA technology because that is mandatory to become compliant with the upcoming VS STANAG.

Therefore, MTG started a project in cooperation with Wismar University in 2005 to explore feasibility and effort of HLA connectivity between existing simulation software. MTG decided to start with a medium scale problem - called *SIMBELFederation* - of connecting two existing Fortran codes for simulating ship motion in seaway and computing seaway heights in a spatially bounded region with a third visualization component. At first view it seems that only the technical problem of linking Fortran code with an RTI implementation has to be overcome to solve the overall problem. After more general examination one realizes that the problem belongs to an HLA application domain with specific characteristics.

In [8] three dissimilar approaches to build HLA based federates are distinguished, which fulfill the specific needs of different application domains:

1.) *Implementation (programming) of federates using common object-oriented programming languages*. That is the approach for which HLA RTIs are originally designed for. Consequently, all relevant RTI implementations provide C++ and Java language bindings.
This approach seems to meet the needs of a wide range of defense simulation applications because that is the domain for which HLA and RTI development was initiated by the U.S. Department of Defense.

2.) *Implementation (modeling) of federates using simulation tools*. This approach is still more a need than reality from the perspective of possible industrial application domains, where it is common to utilize simulation tools. Although extensive research on interfacing HLA RTIs from so-called COTS (*commercial off-the-shelf*) simulation packages has produced solutions in principle ([10]), HLA support is not widely provided by today's COTS.
3.) Beside the above domains, there exists a community in various fields of engineering where neither C++ and Java programming nor modeling with COTS is the preferred approach. Characteristic problems in this domain are extensive numerical simulations and other computations, which are traditionally coded in Fortran. In some areas (e.g. control engineering) Fortran coding has been replaced by computing environments like MATLAB.

The *SIMBELFederation* problem obviously belongs to the last domain. Unfortunately, also in this domain, HLA connectivity is not a matter of course yet. But at least for the MATLAB computing environment there exist experiences from earlier research and also some commercial solutions are available in the meanwhile. Some essential aspects of MATLAB/HLA integration and the current state of the art in this branch are summarized in Section 1. Basic coupling variants between Fortran and HLA RTIs are discussed in Section 2. Section 3 outlines the *SIMBELFederation* problem and its current implementation state. In the Conclusions Section, the overall development effort is discussed and possible ways towards complexity reduction are pointed out.

1 MATLAB/HLA Connectivity

In 1998 the development of an *HLA toolbox* for MATLAB was started at the University of Rostock. Essential design challenges and solutions were published in [7], [8]. A generalization to the entire class of SCEs (*Scientific and Technical Computing Environments*) can be found in [6]. Results of this project which are also meaningful for the problem of Fortran/HLA connectivity are summarized in the following subsections.
In 2005, the first two commercial solutions for MATLAB/HLA connectivity have been released. They are viewed in the last subsection briefly.

1.1 RTI Linkage

The concrete structure of an RTI implementation is vendor dependent. However, the connection between an application and a certain RTI is always realized in a uniform way. For C++ coded applications, RTI implementations provide interface libraries, which have to be linked. This technique can also be used to build a connection between MATLAB and an RTI.
For extension purposes, MATLAB provides a number of external interfaces. One of them is the so-called MEX-interface, which allows dynamic linkage of C code. By this way also C++ libraries can be linked if C style name mangling is enforced (extern ‘C’).

1.2 Procedural HLA Interface

The HLA interface is specified in an object-oriented manner and consists of two fundamental classes which define the *RTI Ambassador* and the *Federate Ambassador*. But as a rule, an application is not simultaneously federate in more than one federation. Hence, only single RTI and Federate Ambassador instances are needed per application.
Consequently, in a MATLAB/HLA integration it is possible to instantiate the two ambassadors below the MEX-interface on the C/C++ layer. Then, on the MATLAB layer only a procedural interface is required to access RTI services and to provide federate services.
For simple handling of an HLA interface in MATLAB, the original very long designations of the RTI and federate services should be replaced by abbreviations. This is in particular necessary for the interactive way of working in MATLAB. Since from the huge number of federate services typically only a few are needed in an application, a MATLAB/HLA-interface should provide predefined federate services.

1.3 Vectorization

RTI and federate services perform scalar operations with elementary data objects as parameter as it is usual in conventional programming. However, efficient MATLAB programming is based on vectorization, whereby code complexity is reduced considerably. Therefore, the routines of a MATLAB/HLA-interface should be vectorized as much as possible.
Due to vectorization not only application code is simplified but also the complexity of a MATLAB/HLA-interface is reduced. For example, in the HLA Toolbox presented in [8] which is based on a DMSO RTI more than 80 auxiliary routines of the primary C++ interface became unnecessary in the MATLAB/HLA interface.

1.4 Commercial Tools

The first commercial product for MATLAB/HLA connectivity was the HLA Toolbox from ForwardSim, Inc. released in May 2005 ([1]). It provides a procedural MATLAB/HLA interface with some features stated in Subsection 1.2. By now, the toolbox supports only HLA 1.3 compliant RTI implementations.

Since July 2005, MÄK Technologies, Inc. offers the product The MÄK HLA/DIS Toolbox for MATLAB and Simulink ([4]). But in the above sense this product is not a direct MATLAB/HLA-interface. Rather the toolbox provides a MATLAB interface to VR-Link. VR-Link is also a product of MÄK Technologies which provides a higher middleware-independent interface. VR-Link can run on top of DIS, HLA 1.3, and IEEE 1516 compliant middleware.

2 Fortran/HLA Connectivity

In various fields of engineering, especially where complex numerical problems have to be solved, Fortran is still the dominating programming language. This fact will probably not change in future because there are a large number of well-tested Fortran codes; the language is continuously refined and standardized. Furthermore, Fortran is the most common language in High Performance Computing (HPC).

In the original HLA main application areas, which are *Distributed Virtual Training Environments* (DTVE) and wargaming, Fortran has no importance. Therefore, established RTI implementations offer programming interfaces for C++ and Java, but not for Fortran.

Currently, development of HLA federates based on Fortran or by using existing Fortran codes is only possilbe by applying one of two indirect methods: The first method requires an appropriate modularization of the Fortran code to allow subsequent integration into an HLA-capable language environment. Then, all HLA specific parts have to be implemented in this language environment. The second method is to employ an HLA interface within Fortran that permits access to an external RTI implementation. Problem specific as well as HLA specific parts are coded in Fortran.

2.1 Integration of Fortran Routines into HLA-Capable Language Environments

Appropriate language environments for this approach are C++, Java, and Matlab. In the following, basic solutions for these three language environments are discussed.

On the object code level, usual Fortran and C++ compilers have very different naming conventions for subroutines and methods, respectively. Therefore, linkage of Fortran and C++ object code is impossible without taking special precautions. But most C++ compilers can provide C-style naming on demand. C and Fortran naming are not identical but the differences are small and can be handled on source code level.

The following code fragments illustrate, how Fortran routines can be integrated into a C++ program:

```
C  Computing routine implemented in Fortran
   SUBROUTINE COMPUTE(result)
     DOUBLE PRECISION result
     ...
     END
/* External declaration of the Fortran computing
   routine as C++ function with C-style naming */
extern "C"{
     extern void compute_(double *result);
}
/* Usage of the Fortran computing routine
     in a C++ program                           */
int main () {
     ...
     compute_(&result);
     rtiamb.updateAttributeValues(...);
     ...
}
```

The code fragments are based on name conventions of the GNU compiler suite and are therefore not generally valid. Hence, this approach is compiler dependent.

Furthermore, differences between representations of data objects in memory in Fortran and C++ have to be taken into consideration. Especially, if arrays are passed as parameters, conversion from the column-oriented representation in Fortran to the row-oriented representation in C++ is required. Such conversions are potentially error-prone.

Java offers integration of extern software via the *Java Native Interface* (JNI). However, only C/C++ is directly supported by JNI. Due to that fact, integration of Fortran routines is only possible by using additional C/C++ wrappers. Hence, for this approach all problems already discussed for the C++ integration have to be solved. In addition, some more platform specific conversions for correct parameter passing between C/C++ and Java need to be accomplished. Consequently, error-proneness increases and runtime performance decreases.

In contrast to C++ and Java, MATLAB with its MEX-Interface supports the integration of Fortran routines very well. Parameter passing between Fortran and MATLAB is also well-supported by the MEX-Interface. No extensive conversions are necessary because MATLAB uses the same representation of array data types like Fortran.

Main advantage of the approach to integrate Fortran routines into HLA-capable language environments is that it is not necessary to implement an HLA interface within Fortran. Thus, realization of HLA federates on basis of existing small and medium Fortran codes is possible without greater effort. The approach is not suitable if HLA federates have to be build using large and complex Fortran codes.

In such cases, it can be difficult to modularize the codes into subroutines in such a way that all HLA parts can be done on top of these routines in the integrating language environment. From a structural and technical point of view, implementations based on the integration approach have to be characterized as ad hoc solutions. The approach enforces the separation of problem specific from HLA specific implementation, which can lead to insufficient structures. The Fortran integration technique into C++ and particular into Java is very wasteful and error-prone.

2.2 Fortran Integrated HLA Access

The integration approach discussed in the previous subsection does not provide direct HLA access within Fortran. But actual Fortran integrated HLA access can be realized in a similar way, as presented for MATLAB in Section 1. Therefore, in analogy, the necessary software layer, which realizes the Fortran/HLA connectivity, is called *HLA toolbox for Fortran*. Prototypes of such an HLA toolbox are currently developed at Wismar University. Essential design issues are discussed subsequently.

A Fortran/HLA toolbox has to work on top of common RTI implementations, which provide C++ and Java interfaces. The C++ interfaces are clearly preferred to build up a Fortran/HLA toolbox. If Java interfaces were used, additional mapping and conversion problems would occur like in the integration approach discussed in the previous subsection.
A Fortran/HLA toolbox which is internally based on a C++ interface has to link an RTI class library. That can be done by mapping the relevant C++ class methods to ordinary functions for which C style naming is enforced.
Like in the MATLAB/HLA toolbox, the *RTI Ambassador* as well as the *Federate Ambassador* can be internally instantiated in the C++ layer. Thus, it becomes possible to build up a purely procedural Fortran interface consisting of subroutines to access RTI services and to provide federate services. As in MATLAB, the very long designations of the RTI and the federate services should be replaced by abbreviations, because it is common Fortran-style to use short designators. Likewise, a Fortran/HLA toolbox should provide predefined federate services. Further design issues depend on the concrete Fortran version. Relevant versions are FORTRAN 77 and Fortran 90/95. FORTRAN 77 does not support vectorization and optional subroutine parameters. Therefore, the interface of an HLA toolbox for FORTRAN 77 cannot be simplified so much as it is possible for Fortran 90/95 and MATLAB.

3 The SIMBELFederation

Figure 1 shows a simplified architecture of the so-called *SIMBELFederation*. It is a medium scale problem consisting of real-world software components used in the ship design process. It serves as test problem for a stepwise examination of suitable solutions for integrating existing software with the HLA technology, especially Fortran codes of various scale. As result of this ongoing research, it should be possible for future HLA projects to form a realistic estimate of the overall development effort and reachable runtime performance in advance.

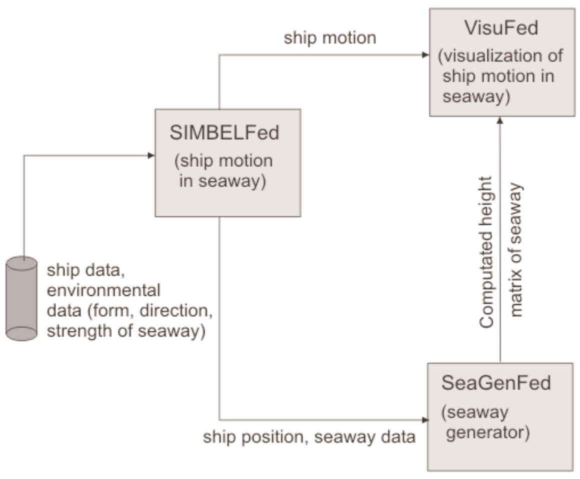

ship motion
VisuFed
(visualization of ship motion in seaway)
SIMBELFed
(ship motion in seaway)
ship data, environmental data (form, direction, strength of seaway)
Computated height matrix of seaway
SeaGenFed
(seaway generator)
ship position, seaway data

Figure 1: Simulation compound SIMBELFederation.

The problem examined consists of three components, which should work together in an HLA federation. *SIMBELFed* is a simulator federate which computes the motion and position of a ship in seaway. The computing federate *SeaGenFed* generates a spatial representation of the seaway around the ship position. The visualization of the ship motion in seaway takes place in the federate *VisuFed*.
The implementation state subsequently discussed is based on the DMSO RTI-1.3NG. Further investigations will also include commercial RTI implementations.

3.1 SIMBELFed

The federate *SIMBELFed* has to integrate the simulation package *SIMBEL* which is used to compute hydrodynamic forces and moments of monohull and multihull ships in different seaways. This permits conclusions about the suitability of ship designs in the operating range supposed. Already since the end of the 1980s, the simulation package *SIMBEL* is developed in cooperation with TU Hamburg-Harburg and the MTG. It is coded in FORTRAN 77 completely and comprises currently approx. 50.000 lines of code.

Because of this high complexity, a first step was to realize an offline-coupling of the *SIMBEL* simulator and the federation through files (Figure 2). In this case, the *SIMBELFed* federate has to contain only simple Fortran routines for reading out simulation data. Consequently, the appropriate integration approach in terms of Section 2 is to insert the Fortran routines into a C++ program, where necessary HLA parts have to be implemented. But this approach is not suitable for the intended online integration of the *SIMBEL* simulator. In that case, a fully operational Fortran/HLA-interface will be necessary.

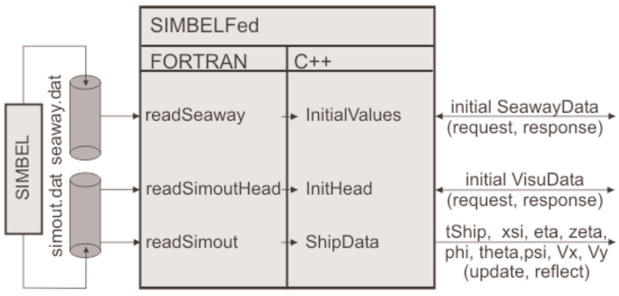

SIMBELFed
FORTRAN
C++
SIMBEL
simout.dat
seaway.dat
readSeaway
InitialValues
readSimoutHead
InitHead
readSimout
ShipData
initial SeawayData (request, response)
initial VisuData (request, response)
tShip, xsi, eta, zeta, phi, theta,psi, Vx, Vy (update, reflect)

Figure 2: Current state of federate *SIMBELFed*.

During initialization *SIMBELFed* reads the file seaway.dat using the Fortran routine readSeaway. The content of this file describes initial environmental data like form, direction, and strength of seaway. The data are hold in the C++ class InitialValues. The file simout.dat contains among others header information about the simulated ship type, which is also read during initialization. All initial data are made available to the federation by the HLA request/response mechanism.

After initialization, the simulated ship motion data are read time stepwise from the file simout.dat. It contains motion data of six degrees of freedom, where *xsi*, *eta*, and *zeta* represent the translations and *phi*, *theta*, and *psi* the rotations in the directions *x*, *y*, and *z*. Furthermore, the ship velocity *Vx* in *x* and *Vy* in *y* direction as well as the timestamp *tShip* are included. After reading, the data are made available to the federation by the HLA update/reflect mechanism.

3.2 SeaGenFed

For the generation of a seaway height matrix also existing Fortran code should be used. It permits the computation of complex natural seaways on basis of a JONSWAP spectrum. The amplitudes of the individual wave components are additively overlaid for each time step and point of the seaway height matrix. The computation effort rises quadratically with the grid size. Currently, grid sizes up to 64x64 points can be computed in real time on a standard PC. The demand for computability of the seaway height matrix in real time arises from the presence of the visualization component *VisuFed* in the federation. If it is necessary to generate seaways of higher accuracy in the future, parallel processing has to be employed. With this in mind, the placement of the seaway generation into a separate federate is very meaningful.

Currently, the federate *SeaGenFed* contains only sequential FORTRAN 77 code, which could be easily integrated into a C++ program. The necessary HLA parts are implemented on the C++ level (see Figure 3). If in future the federate is to employ parallel processing, at least parts of the exiting code have to be ported to Fortran 90. Then it is more efficient to implement also HLA parts in Fortran following the approach described in Section 2.2.

All input data of the federate *SeaGenFed* are obtained from the federate *SIMBELFed*. The initial environmental data *SeawayData* are received during initialization. The current central ship position presented by *xsi* and *eta* is transferred time stepwise. On that basis, *SeaGenFed* computes a seaway height matrix *zmatrix*, which is spatially centered around the central ship position. Subsequently, *zmatrix* is communicated together with its position in the *xy*-plane and with a timestamp *tSeaway* using the HLA update/reflect mechanism.

3.3 VisuFed

The federate *VisuFed* is the primary interface of the federation to the user. It visualizes the moving ship in seaway as a photorealistic three-dimensional representation. Additionally, two-dimensional representations of the ship motion can be displayed.

Recent visualization software is mostly written in C++ or Java. Therefore, there usually exists no HLA connectivity problem. That applies also to the federate *VisuFed* presented in Figure 4, which is currently based on visualization software written in Java.

4 Conclusions

Due to the establishment of new standards the HLA technology will play an important role also in the military ship design process.

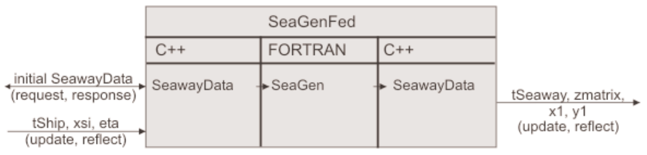

SeaGenFed
C++
FORTRAN
C++
initial SeawayData (request, response)
tShip, xsi, eta (update, reflect)
SeawayData
SeaGen
SeawayData
tSeaway, zmatrix, x1, y1 (update, reflect)

Figure 3: Current state of federate *SeaGenFed.*

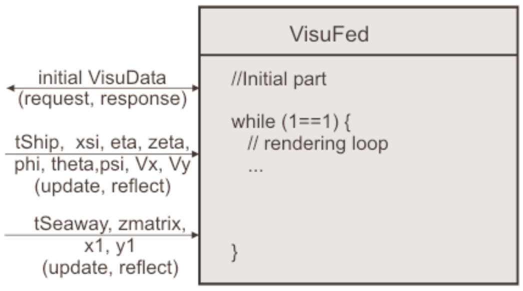


Figure 4: Current state of federate *VisuFed.*

On the other side, simulation- and other engineering software have been developed with large effort over many years and have to be reused as much as possible in future. Therefore, it is necessary to investigate how HLA connectivity can be reached for existing software, and which development effort has to be spent for that.

The research presented in this article is based on earlier works in the field of Matlab/HLA connectivity, which have provided important patterns for the development of Fortran/HLA connectivity. Both, MATLAB and Fortran are essential language environments in many engineering fields. For Fortran two basic coupling approaches were examined:

- Fortran integration in HLA-capable language environments like C++, Java, and MATLAB,
- Fortran integrated HLA access according to the pattern of an HLA toolbox.

The approach exposed first can only be viewed as ad hoc solution for small and medium scale problems. For C++ and Java as integrating language environments it is potentially error-prone and difficult to handle for application developers, because special knowledge in different language environments is required.
The second approach is well suited for problems of arbitrary scale. Application developers do not have to use different programming languages, what is profitable regarding error-proneness and implementation effort. Essential design issues of an HLA toolbox for Fortran were discussed.
Further on, the application problem SIMBELFederation was introduced. The presented current implementation state is based on the approach of integrating Fortran routines into C++ as HLA-capable language environment. With it the proof of concept has been provided, it is possible to connect existing real-world Fortran codes with the HLA technology and the concrete realtime requirements of the application are not violated by the distributed processing approach. As a matter of fact, the correct and effective usage of the HLA concepts depends on deep knowledge in the field of distributed simulation. No interface technology can eliminate this prerequisite.

The discussed Fortran/HLA connectivity approaches are based directly on the HLA interface specification. More than 130 HLA services require a considerable amount of initial training. Therefore, a number of software packages exist defining an interface layer above the HLA service set for complexity reduction. Typical representatives are VR-Link from MÄK technologies (see 2.4, and pSISA [3]), developed by the German Armed Forces (WTD 81). Similar to RTI implementations, C++ language bindings are provided. Therefore, the integration techniques examined in this article can also be applied to realize Fortran connectivity to these packages.

References

[1] *HLA Toolbox - The MATLAB interface to HLA*. Brochure, ForwardSim, Inc., Sainte-Foy, QC, 2005.
[2] K. J. de Kraker, J. Duncan, E.-W. Budde, R. Reading, R.: *NATO Standards for Virtual Ships*. Procs. Fall 2005 Simulation Interoperability Workshop, Paper 05F-SIW-020, Orlando, FL, 2005.
[3] U. Krosta, H.-P. Menzler, K. Pixius: *Implementierung von HLA-Schnittstellen mittels pSISA*. HLA Forum, Magdeburg, 2001.
[4] *MÄK HLA/DIS Toolbox for MATLAB and Simulink*. Broch., MÄK Technologies, Cambridge, MA, 2005.
[5] *NATO STANAG for Virtual Ships*, Study Draft v0.10, Military Agency for Standardisation.
[6] S. Pawletta, W. Drewelow, T. Pawletta: *On the Integration of HLA into SCEs*. TRANSACTIONS of SCS, Vol. 18, No. 2 (2001), pp. 92-97.
[7] S. Pawletta, B. Lampe, T. Pawletta, W. Drewelow: *Eine HLA-Toolbox für Matlab*. In Proc. Simulation und Visualisierung, Magdeburg 2000, SCS Int., pp. 31-44.
[8] S. Pawletta, T. Pawletta, W. Drewelow: *HLA-based Simulation within an Interactive Engineering Environment*. In Proc. 4th IEEE Workshop on Distributed Simulation and Real-Time Applications, San Francisco 2000, pp. 97-102.
[9] H. Schütz, H.: *Wohin steuert die maritime Rüstung in Deutschland?* In Marineforum 04/2003.
[10] S. Straßburger: *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains*. Advances in Simulation, SCS-Europe BVBA Ghent, Belgium, 2001.

Corresponding author: Christian Stenzel
Christian Stenzel, Sven Pawletta, Group Computational Engineering and Automation, Wismar University, PF 1210, 23952 Wismar, Germany; WWW.MB.HS-WISMAR.DE/cea; *christianstenzel@gmx.net*
Richard Ems, Petra Bünning, MTG Marinetechnik GmbH Wandsbeker Königsstraße 62, 22041 Hamburg, Germany
{Richard.Ems; Petra.Buenning}@mtg-marinetechnik.de

Received: June 17, 2006
Revised: July 7, 2006
Accepted: July 20, 2006