CERTI - Bindings to Matlab and Fortran

Christian Stenzel, Sven Pawletta

RG Computational Engineering and Automation
University of Wismar, Germany

10. Magdeburger HLA-Forum
27.02.2008
Outline

1. Introduction

2. Non-Commercial RTIs

3. CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4. Summary and Outlook
Outline

1 Introduction

2 Non-Commercial RTIs

3 CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4 Summary and Outlook
RG Computational Engineering and Automation

- research group at the University of Wismar
- primary research fields:
  - fundamentals of modeling and simulation
  - modeling, simulation and control applications
  - robotics, engine control
  - distributed and parallel scientific computing in the engineering domain
- highly interested in advanced simulation technologies like HLA
- main focus on the engineering domain
research group at the University of Wismar

primary research fields:

- fundamentals of modeling and simulation
- modeling, simulation and control applications
- robotics, engine control
- distributed and parallel scientific computing in the engineering domain

highly interested in advanced simulation technologies like HLA

main focus on the engineering domain
research group at the University of Wismar

primary research fields:

- fundamentals of modeling and simulation
- modeling, simulation and control applications
- robotics, engine control
- distributed and parallel scientific computing in the engineering domain

highly interested in advanced simulation technologies like HLA

main focus on the engineering domain
Motivation

- today’s RTIs provide C++ and/or Java APIs
  - but:
    - simulation model design and execution in the engineering domain today characterized by the usage of Scientific and technical Computation Environments (SCEs) → Matlab
    - existing Fortran codes are daily used, Fortran primary programming language in HPC community

- aims:
  - provide engineers HLA access within their usual working environment
  - provide a way to easily extend existing code to HLA federates
  - increase acceptance of HLA in the engineering domain
Motivation

today’s RTIs provide C++ and/or Java APIs

but:

- simulation model design and execution in the engineering domain today characterized by the usage of Scientific and technical Computation Environments (SCEs) → Matlab
- existing Fortran codes are daily used, Fortran primary programming language in HPC community

aims:

- provide engineers HLA access within their usual working environment
- provide a way to easily extend existing code to HLA federates
- increase acceptance of HLA in the engineering domain
Motivation

- today’s RTIs provide C++ and/or Java APIs
  - but:
    - simulation model design and execution in the engineering domain today characterized by the usage of Scientific and technical Computation Environments (SCEs) → Matlab
    - existing Fortran codes are daily used, Fortran primary programming language in HPC community

- aims:
  - provide engineers HLA access within their usual working environment
  - provide a way to easily extend existing code to HLA federates
  - increase acceptance of HLA in the engineering domain
Motivation

- today’s RTIs provide C++ and/or Java APIs
  - but:
    - simulation model design and execution in the engineering domain today characterized by the usage of Scientific and technical Computation Environments (SCEs) → Matlab
    - existing Fortran codes are daily used, Fortran primary programming language in HPC community

- aims:
  - provide engineers HLA access within their usual working environment
  - provide a way to easily extend existing code to HLA federates
  - increase acceptance of HLA in the engineering domain
Outline

1 Introduction

2 Non-Commercial RTIs

3 CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4 Summary and Outlook
Non-Commercial RTIs

preferred RTI

- free, open source → possibility to study RTI implementation and to participate in development
- C++ bindings → simplifies Fortran binding
- cross platform RTI (including Windows)
- advanced implementation status of HLA services
- active developer community

- a number of non-commercial RTIs exist
- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

- preferred RTI
  - free, open source → possibility to study RTI implementation and to participate in development
    - C++ bindings → simplifies Fortran binding
    - cross platform RTI (including Windows)
    - advanced implementation status of HLA services
    - active developer community

- a number of non-commercial RTIs exist
- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

- preferred RTI
  - free, open source → possibility to study RTI implementation and to participate in development
  - C++ bindings → simplifies Fortran binding
  - cross platform RTI (including Windows)
  - advanced implementation status of HLA services
  - active developer community

- a number of non-commercial RTIs exist

- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

- preferred RTI
  - free, open source → possibility to study RTI implementation and to participate in development
  - C++ bindings → simplifies Fortran binding
  - cross platform RTI (including Windows)
    - advanced implementation status of HLA services
    - active developer community

- a number of non-commercial RTIs exist
- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

- preferred RTI
  - free, open source → possibility to study RTI implementation and to participate in development
  - C++ bindings → simplifies Fortran binding
  - cross platform RTI (including Windows)
  - advanced implementation status of HLA services
  - active developer community

- a number of non-commercial RTIs exist
- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

preferred RTI

- free, open source → possibility to study RTI implementation and to participate in development
- C++ bindings → simplifies Fortran binding
- cross platform RTI (including Windows)
- advanced implementation status of HLA services
- active developer community

a number of non-commercial RTIs exist

overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

- preferred RTI
  - free, open source → possibility to study RTI implementation and to participate in development
  - C++ bindings → simplifies Fortran binding
  - cross platform RTI (including Windows)
  - advanced implementation status of HLA services
  - active developer community

- a number of non-commercial RTIs exist
- overview about existing non-commercial RTI implementations required
Non-Commercial RTIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor</th>
<th>Standard</th>
<th>Bindings</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BH-RTI</td>
<td>Beijing University</td>
<td>1.3, IEEE 1516</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CERTI</td>
<td>ONERA</td>
<td>1.3 partial, IEEE 1516</td>
<td>C++, Java planned</td>
<td>GPL, LGPL</td>
</tr>
<tr>
<td>EODiSP HLA</td>
<td>P&amp;P Software</td>
<td>IEEE 1516 partial</td>
<td>Java</td>
<td>GPL</td>
</tr>
<tr>
<td>GERTICO</td>
<td>Fraunhofer IITB</td>
<td>1.3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Open HLA</td>
<td></td>
<td>1.3 partial, IEEE 1516</td>
<td>Java</td>
<td>Apache License</td>
</tr>
<tr>
<td>RTI-S</td>
<td>US JFCOM J9 Directorate</td>
<td>1.3 partial</td>
<td>C++, Java</td>
<td>US Government</td>
</tr>
<tr>
<td>poRTIco</td>
<td>littlebluefrog labs</td>
<td>1.3 partial, IEEE 1516</td>
<td>Java, C++</td>
<td>CDDL</td>
</tr>
<tr>
<td>Rendezvous RTI</td>
<td>NUST, Pakistan</td>
<td>1.3</td>
<td>C++, Java</td>
<td>NUST</td>
</tr>
</tbody>
</table>

Table: www.wikipedia.org, Feb. 2008
## Non-Commercial RTIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor</th>
<th>Standard</th>
<th>Bindings</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BH-RTI</td>
<td>Beijing University</td>
<td>1.3, IEEE 1516</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CERTI</td>
<td>ONERA</td>
<td>1.3 partial, IEEE</td>
<td>C++, Java planned</td>
<td>GPL, LGPL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1516 planned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EODiSP HLA</td>
<td>P&amp;P Software</td>
<td>IEEE 1516 partial</td>
<td>Java</td>
<td>GPL</td>
</tr>
<tr>
<td>GERTICO</td>
<td>Fraunhofer IITB</td>
<td>1.3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Open HLA</td>
<td></td>
<td>1.3 partial, IEEE</td>
<td>Java</td>
<td>Apache License</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1516 partial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTI-S</td>
<td>US JFCOM J9 Directorate</td>
<td>1.3 partial</td>
<td>C++, Java</td>
<td>US Government</td>
</tr>
<tr>
<td>poRTico</td>
<td>littlebluefrog labs</td>
<td>1.3 partial, IEEE</td>
<td>Java, C++</td>
<td>CDDL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1516 partial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rendezvous RTI</td>
<td>NUST, Pakistan</td>
<td>1.3</td>
<td>C++, Java</td>
<td>NUST</td>
</tr>
</tbody>
</table>

*Table: www.wikipedia.org, Feb. 2008*

*not downloadable or no activities since 12 months; only binaries available*
## Non-Commercial RTIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor</th>
<th>Standard</th>
<th>Bindings</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BH-RTI</td>
<td>Beijing University</td>
<td>1.3, IEEE 1516</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CERTI</td>
<td>ONERA</td>
<td>1.3 partial, IEEE 1516 planned</td>
<td>C++, Java planned</td>
<td>GPL, LGPL</td>
</tr>
<tr>
<td>EODiSP HLA</td>
<td>P&amp;P Software</td>
<td>IEEE 1516 partial</td>
<td>Java</td>
<td>GPL</td>
</tr>
<tr>
<td>GERTICO</td>
<td>Fraunhofer IITB</td>
<td>1.3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Open HLA</td>
<td></td>
<td>1.3 partial, IEEE 1516 partial</td>
<td>Java</td>
<td>Apache License</td>
</tr>
<tr>
<td>RTI-S</td>
<td>US JFCOM J9 Directorate</td>
<td>1.3 partial</td>
<td>C++, Java</td>
<td>US Government</td>
</tr>
<tr>
<td>poRTIco</td>
<td>littlebluefrog labs</td>
<td>1.3 partial, IEEE 1516 partial</td>
<td>Java, C++</td>
<td>CDDL</td>
</tr>
<tr>
<td>Rendezvous RTI</td>
<td>NUST, Pakistan</td>
<td>1.3</td>
<td>C++, Java</td>
<td>NUST</td>
</tr>
</tbody>
</table>

**Table:** www.wikipedia.org, Feb. 2008

not downloadable or no activities since 12 months; only binaries available
## Non-Commercial RTIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Vendor</th>
<th>Standard</th>
<th>Bindings</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BH-RTI</td>
<td>Beijing University</td>
<td>1.3, IEEE 1516</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CERTI</td>
<td>ONERA</td>
<td>1.3 partial, IEEE 1516</td>
<td>C++, Java planned</td>
<td>GPL, LGPL</td>
</tr>
<tr>
<td>EODiSP HLA</td>
<td>P&amp;P Software</td>
<td>IEEE 1516 partial</td>
<td>Java</td>
<td>GPL</td>
</tr>
<tr>
<td>GERTICO</td>
<td>Fraunhofer IITB</td>
<td>1.3</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Open HLA</td>
<td></td>
<td>1.3 partial, IEEE 1516</td>
<td>Java</td>
<td>Apache License</td>
</tr>
<tr>
<td>RTI-S</td>
<td>US JFCOM J9 Directorate</td>
<td>1.3 partial</td>
<td>C++, Java</td>
<td>US Government</td>
</tr>
<tr>
<td>poRTIco</td>
<td>littlebluefrog labs</td>
<td>1.3 partial, IEEE 1516</td>
<td>Java, C++</td>
<td>CDDL</td>
</tr>
<tr>
<td>Rendezvous RTI</td>
<td>NUST, Pakistan</td>
<td>1.3</td>
<td>C++, Java</td>
<td>NUST</td>
</tr>
</tbody>
</table>

**Table:** [www.wikipedia.org](http://www.wikipedia.org), Feb. 2008

*not downloadable or no activities since 12 months; only binaries available*
The poRTIco project

General Information

- Former known as jaRTI, developed since 2005, first public release June 2006
- Since May 2007 poRTIco
- License: COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL, more "commercial friendly" than GNU LGPL)
- Homepage and development site: http://porticoproject.org

Pros

- Open source, based on Java
- C++ and Java bindings
- HLA1.3 partial, IEEE1516 partial

Cons

- Three main developers, all PhD students → What follows after their thesis?
- v0.8rc1, Feb. 2008; version < 1.0
the poRTIco project

• general information
  ▶ former known as jaRTI, developed since 2005, first public release June 2006
  ▶ since May 2007 poRTIco
  ▶ license: COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL, more "commercial friendly" than GNU LGPL)
  ▶ homepage and development site: http://porticoproject.org

• pros
  ▶ open source, based on Java
  ▶ C++ and Java bindings
  ▶ HLA1.3 partial, IEEE1516 partial

• cons
  ▶ three main developers, all phd students → What follows after their thesis?
  ▶ v0.8rc1, Feb. 2008; version < 1.0
general information

- former known as jaRTI, developed since 2005, first public release June 2006
- since May 2007 poRTIco
- license: COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL, more "commercial friendly" than GNU LGPL)
- homepage and development site: http://porticoproject.org

pros

- open source, based on Java
- C++ and Java bindings
- HLA1.3 partial, IEEE1516 partial

cons

- three main developers, all phd students → What follows after their thesis?
- v0.8rc1, Feb. 2008; version < 1.0
general information

- developed at CERT (ONERA Research Centre, ONERA - French aerospace lab)
- license: GNU GPL, libraries under GNU LGPL
- homepage: http://www.cert.fr/CERTI
- devel@Savannah: http://savannah.nongnu.org/projects/certi

pros

- active developer community (11 members)
- current release 3.2.5, well tested RTI
- HLA1.3 nearly complete, HLA1516 planned
- cmake build system, different tool chains possible (gcc, msc, icc)
- HP CERTI, optimized for SMP multiprocessor machines

cons

- at present only C++ bindings; Java bindings planned
general information

- developed at CERT (ONERA Research Centre, ONERA - French aerospace lab)
- license: GNU GPL, libraries under GNU LGPL
- homepage: http://www.cert.fr/CERTI
- devel@Savannah: http://savannah.nongnu.org/projects/certi

pros

- active developer community (11 members)
- current release 3.2.5, well tested RTI
- HLA1.3 nearly complete, HLA1516 planned
- cmake build system, different tool chains possible (gcc, msc, icc)
- HP CERTI, optimized for SMP multiprocessor machines

cons

- at present only C++ bindings; Java bindings planned
general information
- developed at CERT (ONERA Research Centre, ONERA - French aerospace lab)
- license: GNU GPL, libraries under GNU LGPL
- homepage: http://www.cert.fr/CERTI
- devel@Savannah: http://savannah.nongnu.org/projects/certi

pros
- active developer community (11 members)
- current release 3.2.5, well tested RTI
- HLA1.3 nearly complete, HLA1516 planned
- cmake build system, different tool chains possible (gcc, msc, icc)
- HP CERTI, optimized for SMP multiprocessor machines

cons
- at present only C++ bindings; Java bindings planned
bullet general information
  ▶ developed at CERT (ONERA Research Centre, ONERA - French aerospace lab)
  ▶ license: GNU GPL, libraries under GNU LGPL
  ▶ homepage: http://www.cert.fr/CERTI
  ▶ devel@Savannah: http://savannah.nongnu.org/projects/certi
bullet pros
  ▶ active developer community (11 members)
  ▶ current release 3.2.5, well tested RTI
  ▶ HLA1.3 nearly complete, HLA1516 planned
  ▶ cmake build system, different tool chains possible (gcc, msc, icc)
  ▶ HP CERTI, optimized for SMP multiprocessor machines
bullet cons
  ▶ at present only C++ bindings; Java bindings planned

CERTI fulfills all requirements
Outline

1 Introduction

2 Non-Commercial RTIs

3 CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4 Summary and Outlook
CERTI Architecture

- federates interfacing libRTI through HLA interface specification
- RTIA process interacts with federate process through UNIX-domain sockets (Windows TCP sockets)
- RTIG interacts with RTIA through TCP/UDP sockets
- HP CERTI replaces socket communication through shared memory communication

CERTI Architecture

- federates interfacing libRTI through HLA interface specification
- RTIA process interacts with federate process through UNIX-domain sockets (Windows TCP sockets)
- RTIG interacts with RTIA through TCP/UDP sockets
- HP CERTI replaces socket communication through shared memory communication

CERTI Architecture

- federates interfacing libRTI through HLA interface specification
- RTIA process interacts with federate process through UNIX-domain sockets (Windows TCP sockets)
- RTIG interacts with RTIA through TCP/UDP sockets
- HP CERTI replaces socket communication through shared memory communication
CERTI Architecture

- federates interfacing libRTI through HLA interface specification
- RTIA process interacts with federate process through UNIX-domain sockets (Windows TCP sockets)
- RTIG interacts with RTIA through TCP/UDP sockets
- HP CERTI replaces socket communication through shared memory communication

Bindings to Procedural Language Environments

- object oriented HLA interface $\iff$ procedural Matlab, Fortran
- mapping not straightforward, general problems:
  - bidirectional communication
  - object instantiation
  - function overloading
  - exception handling
- data type conversion
- linkage against libRTI
Bindings to Procedural Language Environments

- object oriented HLA interface \(\leftrightarrow\) procedural Matlab, Fortran

  - mapping not straightforward, general problems:
    - bidirectional communication
    - object instantiation
    - function overloading
    - exception handling

- data type conversion
- linkage against libRTI
Bindings to Procedural Language Environments

- object oriented HLA interface $\iff$ procedural Matlab, Fortran
- mapping not straightforward, general problems:
  - bidirectional communication
  - object instantiation
  - function overloading
  - exception handling
- data type conversion
- linkage against libRTI
Outline

1. Introduction
2. Non-Commercial RTIs
3. CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran
4. Summary and Outlook
MatlabHLA-Toolbox

- general information:
  - project developed by RG CEA at the University of Wismar
  - started in 1998
  - homepage: http://www.mb.hs-wismar.de/cea/sw_projects.html
  - development: http://savannah.nongnu.org/projects/certi

- features:
  - abbreviated RTI service designators
  - selectable exception handling
  - default interactive federate services
  - use of vectorization, implicit data types → simpler RTI interface

- implementation status:
  - standard: HLA 1.3
  - Federation Management, Declaration Management, Object Management, Time Management complete
  - Data Distribution Management, Ownership Management not yet implemented
MatlabHLA-Toolbox

general information:
- project developed by RG CEA at the University of Wismar
- started in 1998
- homepage: http://www.mb.hs-wismar.de/cea/sw_projects.html
- development: http://savannah.nongnu.org/projects/certi

features:
- abbreviated RTI service designators
- selectable exception handling
- default interactive federate services
- use of vectorization, implicit data types → simpler RTI interface

implementation status:
- standard: HLA 1.3
- Federation Management, Declaration Management, Object Management, Time Management complete
- Data Distribution Management, Ownership Management not yet implemented
MatlabHLA-Toolbox

- **general information:**
  - project developed by RG CEA at the University of Wismar
  - started in 1998
  - homepage: [http://www.mb.hs-wismar.de/cea/sw_projects.html](http://www.mb.hs-wismar.de/cea/sw_projects.html)

- **features:**
  - abbreviated RTI service designators
  - selectable exception handling
  - default interactive federate services
  - use of vectorization, implicit data types → simpler RTI interface

- **implementation status:**
  - standard: HLA 1.3
  - Federation Management, Declaration Management, Object Management, Time Management complete
  - Data Distribution Management, Ownership Management not yet implemented
MatlabHLA-Toolbox
Bidirectional Communication

- Matlab External (MEX) interface allows access to external libraries

- RTI Services
  - Matlab federate calls RTI service m-function
  - m-function directly calls function within C++-wrapper (m2RtiSrv)
  - type conversion (m2c)
  - m2RtiSrv calls appropriate RTIamb method

- Federate Services
  - libRTI calls implemented federate services in C++-wrapper
  - type conversion (c2m)
  - fedService m-file invoked by mexCallMatlab()
MatlabHLA-Toolbox
Bidirectional Communication

- Matlab External (MEX) interface allows access to external libraries
- RTI Services
  - Matlab federate calls RTI service m-function
  - m-function directly calls function within C++-wrapper (m2RtiSrv)
  - type conversion (m2c)
  - m2RtiSrv calls appropriate RTIamb method
- Federate Services
  - libRTI calls implemented federate services in C++-wrapper
  - type conversion (c2m)
  - fedService m-file invoked by mexCallMatlab()
MatlabHLA-Toolbox

Bidirectional Communication

- Matlab External (MEX) interface allows access to external libraries
- RTI Services
  - Matlab federate calls RTI service m-function
  - m-function directly calls function within C++-wrapper (m2RtiSrv)
  - type conversion (m2c)
  - m2RtiSrv calls appropriate RTIamb method
- Federate Services
  - libRTI calls implemented federate services in C++-wrapper
  - type conversion (c2m)
  - fedService m-file invoked by mexCallMatlab()
MatlabHLA-Toolbox

Object Instantiation

- objects (RTIamb, fedAmb) statically instantiated once in C++-wrapper (one module)
- plain procedural interface towards libRTI
- accidental/automatic module unloading has to be avoided → module locking
MatlabHLA-Toolbox
Object Instantiation

- objects (RTIamb, fedAmb) statically instantiated once in C++-wrapper (one module)
- plain procedural interface towards libRTI
- accidental/automatic module unloading has to be avoided → module locking
MatlabHLA-Toolbox

Object Instantiation

- objects (RTIamb, fedAmb) statically instantiated once in C++-wrapper (one module)
- plain procedural interface towards libRTI
- accidental/automatic module unloading has to be avoided → module locking
MatlabHLA-Toolbox

Function Overloading

- RTI-interface makes extensive use of overloaded methods
- Matlab does not support overloading natively
- MATLAB/MEX allows analysis of function signatures (number, types)

Matlab Example

```matlab
function fedService(in1, in2, in3, in4)
    if nargin==4
        ...
    else
        ...
    end
```
MatlabHLA-Toolbox

Function Overloading

- RTI-interface makes extensive use of overloaded methods
- Matlab does not support overloading natively
- MATLAB/MEX allows analysis of function signatures (number, types)

Matlab Example

```matlab
function fedService(in1, in2, in3, in4)
    if nargin==4
        ...
    else
        ...
    end
```
MatlabHLA-Toolbox

Function Overloading

- RTI-interface makes extensive use of overloaded methods
- Matlab does not support overloading natively
- MATLAB/MEX allows analysis of function signatures (number, types)

Matlab Example

```matlab
function fedService(in1, in2, in3, in4)
    if nargin==4
        ...
    else
        ...
    end
```
MatlabHLA-Toolbox

Function Overloading

- RTI-interface makes extensive use of overloaded methods
- Matlab does not support overloading natively
- MATLAB/MEX allows analysis of function signatures (number, types)

Matlab Example

```matlab
function fedService(in1, in2, in3, in4)
    if nargin==4
        ...
    else
        ...
    end
```
MatlabHLA-Toolbox

Exception Handling

- possible exception caught (C++-Wrapper) and returned to Matlab
- MatlabHLA m-files provide optional error return value
- complex error handling in Matlab federate possible

Matlab Example

```matlab
... 
rtiSrv(in1, in2)
... 
err = rtiSrv(in1, in2) 
switch err
  case 'RTIinternalError'
  ...
```
MatlabHLA-Toolbox

Exception Handling

- possible exception caught (C++-Wrapper) and returned to Matlab
- MatlabHLA m-files provide optional error return value
- complex error handling in Matlab federate possible

Matlab Example

```matlab
... rtiSrv(in1, in2) ...
err = rtiSrv(in1, in2)
switch err
  case 'RTIinternalError'
    ...
```
MatlabHLA-Toolbox

Exception Handling

- possible exception caught (C++-Wrapper) and returned to Matlab
- MatlabHLA m-files provide optional error return value
- complex error handling in Matlab federate possible

Matlab Example

```matlab
...  
rtiSrv(in1, in2)  
...  
err = rtiSrv(in1, in2)  
switch err  
case 'RTIinternalError'  
...  
```
MatlabHLA-Toolbox
Exception Handling

- possible exception caught (C++-Wrapper) and returned to Matlab
- MatlabHLA m-files provide optional error return value
- complex error handling in Matlab federate possible

Matlab Example

```matlab
... rtiSrv(in1, in2) ...
err = rtiSrv(in1, in2)
switch err
  case 'RTIinternalError'
  ...
...```
Outline

1. Introduction

2. Non-Commercial RTIs

3. CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4. Summary and Outlook
libF90HLA

- general information:
  - project developed by RG CEA at the University of Wismar
  - library for use with Fortran90, FORTRAN77 subset of Fortran90
  - MatlabHLA used as design pattern
  - homepage: [http://www.mb.hs-wismar.de/cea/sw_projects.html](http://www.mb.hs-wismar.de/cea/sw_projects.html)

- features:
  - abbreviated RTI service designators
  - selectable exception handling
  - default federate services
  - support of gcc 3.x, gcc 4.x, icc 9.x, icc 10.x

- implementation status:
  - standard: HLA 1.3
  - Federation Management complete
  - other management areas partial
libF90HLA

general information:
▶ project developed by RG CEA at the University of Wismar
▶ library for use with Fortran90, FORTRAN77 subset of Fortran90
▶ MatlabHLA used as design pattern
▶ homepage: http://www.mb.hs-wismar.de/cea/sw_projects.html
▶ development: http://savannah.nongnu.org/projects/certi

features:
▶ abbreviated RTI service designators
▶ selectable exception handling
▶ default federate services
▶ support of gcc 3.x, gcc 4.x, icc 9.x, icc 10.x

implementation status:
▶ standard: HLA 1.3
▶ Federation Management complete
▶ other management areas partial
libF90HLA

- general information:
  - project developed by RG CEA at the University of Wismar
  - library for use with Fortran90, FORTRAN77 subset of Fortran90
  - MatlabHLA used as design pattern
  - homepage: [http://www.mb.hs-wismar.de/cea/sw_projects.html](http://www.mb.hs-wismar.de/cea/sw_projects.html)

- features:
  - abbreviated RTI service designators
  - selectable exception handling
  - default federate services
  - support of gcc 3.x, gcc 4.x, icc 9.x, icc 10.x

- implementation status:
  - standard: HLA 1.3
  - Federation Management complete
  - other management areas partial
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

realizable by

- C++ compiler: C style naming (extern "C") → linkage Fortran C
- preprocessor constants for compiler dependencies
- functions for type conversion

```fortran
PROGRAM CALLC
   /* rtisrv as c++ function with C-style naming */
   extern "C" {
      void rtisrv_(int *a);
   }
   INTEGER :: a
   CALL rtisrv(a) ----> void rtisrv_(int *a) {
      RTItype a_rti = fortranInt2RTItype(*a);
      rtiAmb.rtiSrv(a_rti);
   }
END PROGRAM CALLC
```
**libF90HLA**

**Linkage against libRTI**

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```fortran
PROGRAM CALLC /* rtisrv as c++ function with C-style naming */
!
IMPLICIT NONE
void rtisrv_(int *a);
!
INTEGER :: a
!
CALL rtisrv(a) ---> void rtisrv_(int *a) {
!
RTItype a_rti = fortranInt2RTItype(*a);
!
rtiAmb.rtiSrv(a_rti);
!
}
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```fortran
PROGRAM CALLC
  /* rtisrv as c++ function with C-style naming */
  extern "C" {
    void rtisrv_(int *a);
  }
  INTEGER :: a
  CALL rtisrv(a) ----> void rtisrv_(int *a) {
    RTItype a_rti = fortranInt2RTItype(*a);
    rtiAmb.rtiSrv(a_rti);
  }
END PROGRAM CALLC
```
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```fortran
PROGRAM CALLC
!
IMPLICIT NONE
!
INTEGER :: a
!
CALL rtisrv(a) ---> void rtisrv_(int *a) {
    RTItype a_rti = fortranInt2RTItype(*a);
    rtiAmb.rtiSrv(a_rti);
}
!
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```
PROGRAM CALLC /* rtisrv as c++ function with C-style naming */
!
IMPLICIT NONE
!
INTEGER :: a
!
CALL rtisrv(a)  ---> void rtisrv_(int *a) {
  !
  void rtisrv_(int *a) {
    RTItype a_rti = fortranInt2RTItype(*a);
    rtiAmb.rtiSrv(a_rti);
    
  RTItype a_rti = fortranInt2RTItype(*a);
  rtiAmb.rtiSrv(a_rti);
}
!
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```fortran
PROGRAM CALLC
    /* rtisrv as c++ function with C-style naming */
    extern "C" {
        void rtisrv_(int *a);
    }
    void rtisrv_(int *a) {
        RTItype a_rti = fortranInt2RTItype(*a);
        rtiAmb.rtiSrv(a_rti);
    }
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```fortran
PROGRAM CALLC
    /* rtisrv as c++ function with C-style naming */
    !
    IMPLICIT NONE
    !
    INTEGER :: a
    !
    CALL rtisrv(a)    --> void rtisrv_(int *a) {
        RTItype a_rti = fortranInt2RTItype(*a);
        rtiAmb.rtiSrv(a_rti);
    }
END PROGRAM CALLC
```
libF90HLA

Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: `modulename_mp_fcnname_`, gfortran: `__modulename__fcnname`)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```cpp
PROGRAM CALLC /* rtisrv as c++ function with C-style naming */
!
IMPLICIT NONE
!
INTEGER :: a
!
CALL rtisrv(a) ---> void rtisrv_(int *a) {
    RTItype a_rti = fortranInt2RTItype(*a);
    rtiAmb.rtiSrv(a_rti);
}
!
END PROGRAM CALLC
```
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```c
PROGRAM CALLC /* rtisrv as c++ function with C-style naming */
!
IMPLICIT NONE !
!
INTEGER :: a !
!
CALL rtisrv(a) ---> void rtisrv_(int *a) {
!
RTItype a_rti = fortranInt2RTItype(*a);
rtiAmb.rtiSrv(a_rti);
!
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference

- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```plaintext
PROGRAM CALLC
    /* rtisrv as c++ function with C-style naming */
    extern "C" {
        void rtisrv_(int *a);
    }
    INTEGER :: a
    CALL rtisrv(a)  --->  void rtisrv_(int *a) {
        RTItype a_rti = fortranInt2RTItype(*a);
        rtiAmb.rtiSrv(a_rti);
    }
END PROGRAM CALLC
```
libF90HLA
Linkage against libRTI

- linkage Fortran C++ not straightforward
  - C++/Fortran compiler use non-standardized name mangling (ifort: modulename_mp_fcnname_, gfortran: __modulename__fcnname)
  - data types, e.g. C row-major order, Fortran column major-order
  - parameters only passed by reference
- realizable by
  - C++ compiler: C style naming (extern "C") → linkage Fortran C
  - preprocessor constants for compiler dependencies
  - functions for type conversion

```
PROGRAM CALLC
    /* rtisrv as c++ function with C-style naming */
    extern "C" {
        void rtisrv_(int *a);
    }
    INTEGER :: a
    CALL rtisrv(a)  --> void rtisrv_(int *a) {
        RTItype a_rti = fortranInt2RTItype(*a);
        rtiAmb.rtiSrv(a_rti);
    }
END PROGRAM CALLC
```
libF90HLA
Bidirectional Communication

- RTI Services
  - F90 federate invokes RTI services by calling the rtiModSrv fcn
  - signature analysis and C fcn call
  - C++ type conversion, call rtiAmb method

- Federate Services
  - libRTI calls implemented federate services
  - type conversion from C/C++ types into Fortran types
  - appropriate Fortran90 federate service implementation is called
libF90HLA
Bidirectional Communication

- **RTI Services**
  - F90 federate invokes RTI services by calling the rtiModSrv fcn
  - signature analysis and C fcn call
  - C++ type conversion, call rtiAmb method

- **Federate Services**
  - libRTI calls implemented federate services
  - type conversion from C/C++ types into Fortran types
  - appropriate Fortran90 federate service implementation is called
libF90HLA
Object Instantiation

- similar to MatlabHLA-Toolbox
- RTIamb, fedAmb statically instantiated in libF90HLA (rtiOn)
- plain procedural interface towards libRTI
**libF90HLA**

**Function Overloading**

- F90 allows optional function parameters
- intrinsic function *present* can test existence of optional parameters
- tests performed in F90 module

```fortran
subroutine rtiSrv(in1, in2)  
! implicit none  
! integer, intent(in) :: in1  
integer, intent(in), optional :: in2  
...  
if (present(in2)) then  
...  
end if  
...  
end subroutine rtiSrv
```
F90 allows optional function parameters

intrinsic function *present* can test existence of optional parameters

tests performed in F90 module

```fortran
subroutine rtiSrv(in1, in2)
  ! implicit none
  ! integer, intent(in) :: in1
  integer, intent(in), optional :: in2
  ...
  if (present(in2)) then
    ...
  end if
  ...
end subroutine rtiSrv
```
libF90HLA

Exception Handling

- similar method to realize selectable exception handling
- default: program termination
- optional error parameter allows exception handling in F90 fed

```
rtiSrvMod.f90

subroutine rtiSrv(err)
  ...
  integer, intent(out), optional :: err
  integer :: tmpErr = 0
  ...
  call rtisrvwrap(tmpErr)
  if (present(err)) then
    err = tmpErr
  else
    if (tmpErr.lt.0) then
      write(*,*) "Error = ", tmpErr
      stop "Terminating"
    end if
  end if
  ...
```

F90HLAWrap.cpp

```cpp
rtsrv_ {
  For2RTItypes();
  rtiAmb.rtiSrv();
}
```
libF90HLA

Exception Handling

- similar method to realize selectable exception handling
- default: program termination
- optional error parameter allows exception handling in F90 fed

**rtiSrvMod.f90**

```fortran
subroutine rtiSrv(err)
  ...
  integer, intent(out), optional :: err
  integer :: tmpErr = 0
  ...
  call rtisrvwrap(tmpErr)
  if (present(err)) then
    err = tmpErr
  else
    if (tmpErr.lt.0) then
      write(*,*) "Error = ", tmpErr
      stop "Terminating"
    end if
  end if
  ...
```

F90HLAWrap.cpp

```cpp
rtisrv_ {
  For2RTItypes();
  rtiAmb.rtiSrv();
}
```
Outline

1. Introduction

2. Non-Commercial RTIs

3. CERTI Bindings
   - Basic Aspects
   - Bindings to Matlab
   - Bindings to Fortran

4. Summary and Outlook
Summary and Outlook

Summary

» at present two noteworthy open source RTIs: poRTIco, CERTI
  → CERTI
» CERTI well tested RTI, remarkable development site at Savannah
» introduction of two new open source projects: MatlabHLA, libF90HLA
» CERTI first RTI with native bindings to Matlab and Fortran

Outlook

» completing work at libF90HLA
» Simulink-Toolbox on basis of HLA-Toolbox, HLA integration into other free SCEs (e.g. Octave)
» finding bugs, applying patches and looking for help ...