CoHLA
Installation Manual

Thomas Nägele
Radboud University
t.nagele@cs.ru.nl

30th July 2019

Contents

1 Introduction

This manual describes the installation steps of the CoHLA project onto your system. The project aims for an easier implementation of co-simulations of systems using the High Level Architecture. Using a DSL, one can easily describe a model-based co-simulation. From this co-simulation specification, code is generated that can be compiled and simulated using OpenRTI.

2 Ubuntu installation

2.1 Requirements

The following requirements should be met before proceeding with this installation manual.

- Ubuntu Desktop 18.04[1]
- A user having root privileges
- Internet connection

2.2 Installation

It might be useful to place all manually downloaded files into one directory to retrieve these files later on or to possibly repeat one or more installation steps. For example, a directory such as `~/Downloads/CoHLA` could be used.

1. Start the installation by ensuring the most recent software is installed by updating and upgrading using APT.

```
sudo apt-get update
sudo apt-get upgrade
```

2. Install the following packages from the Ubuntu software repository:
   - build-essential
   - cmake
   - git
   - libglm-dev
   - libglfw3-dev
   - libboost-system-dev
   - libboost-thread-dev
   - libxerces-c-dev
   - openjdk-8-jdk
   - python3

```
sudo apt-get install build-essential cmake git libglm-dev
    libglfw3-dev libboost-system-dev libboost-thread-dev
    libxerces-c-dev openjdk-8-jdk python3
```

3. Build and install OpenRTI.
   (a) Download and extract OpenRTI[2]

```
wget -O OpenRTI.tar.bz2
    https://sourceforge.net/projects/openrti/files/OpenRTI-0.9.0.tar.bz2
    -Lq
tar xf OpenRTI.tar.bz2
```

[1] https://www.ubuntu.com/desktop/1804
[2] https://sourceforge.net/projects/openrti/

(b) Move into directory that was just created, create a build directory and move into it.

```
cd OpenRTI-0.9.0
mkdir build
cd build
```

(c) Use CMake and Make to build and install OpenRTI.

```
cmake -DOPENRTI_BUILD_SHARED=ON ..
sudo make install
```

4. Build and install FMI Library.

(a) Download and extract FMI Library[3].

```
wget -O "FMI-lib.zip" http://www.jmodelica.org/
    downloads/FMIL/FMILibrary-2.0.3-src.zip
unzip -u FMI-lib.zip
```

(b) Move into the directory that was just created, create a build directory and move into it.

```
cd FMILibrary-2.0.3
mkdir build
cd build
```

(c) Use CMake and Make to build and install FMI Library.

```
cmake -DFMILIB_INSTALL_PREFIX=/usr/local
    -DFMILIB_BUILD_TESTS=OFF
    -DFMILIB_GENERATE_DOXYGEN_DOC=OFF ..
sudo make install
```

The installation directory `/opt/FMI-lib` can be changed to any other desired installation directory.

5. Build and install Bullet Physics Library.

(a) Download and extract Bullet Physics Library[4]. Then create a build directory and move into it.

```
wget -O bullet.tar.gz https://github.com/bulletphysics/
    bullet3/archive/2.86.1.tar.gz
tar xf bullet.tar.gz
cd bullet3-2.86.1
mkdir build
cd build
```

(b) Use CMake and Make to build and install Bullet Physics Library.

```
cmake -DBUILD_SHARED_LIBS=ON -DINSTALL_LIBS=ON
    -DINSTALL_EXTRA_LIBS=ON -DBUILD_OPENGL3_DEMOS=OFF
    -DBUILD_PYBULLET=OFF -DBUILD_CPU_DEMOS=OFF
    -DBUILD_BULLET2_DEMOS=OFF -DBUILD_UNIT_TESTS=OFF ..
sudo make install
```

[3] `http://www.jmodelica.org/FMILibrary`
[4] `http://bulletphysics.org/wordpress/`

6. Download and install CodeSynthesis XSD[5].

```
wget -O cs-xsd.deb
    https://www.codesynthesis.com/download/xsd/4.0/linux-gnu/
    x86_64/xsd_4.0.0-1_amd64.deb
sudo dpkg -i cs-xsd.deb
```

7. Install the OpenRTI Libraries.

   (a) Clone the CoHLA project, create a build directory and move into it.

   ```
   git clone https://github.com/phpnerd/CoHLA.git
   mkdir CoHLA/libs/build
   cd CoHLA/libs/build
   ```

   (b) Build and install the libraries.

   ```
   cmake -DCMAKE_INSTALL_PREFIX=/opt/OpenRTI-libs
       -DBUILD_SHARED_LIBS=ON ..
   sudo make install
   ```

   Instead of `/opt/OpenRTI-libs`, any other desired installation directory may be used.

8. Install the Rotalumis simulator for POOSL models.

   (a) Download the integration executable of Rotalumis[6].

   ```
   wget http://www.es.ele.tue.nl/rotalumis/
       executables/integration/linux/64bit/rotalumis
   ```

   (b) Move the Rotalumis executable to `/usr/bin` and make it runnable.

   ```
   sudo cp rotalumis /usr/bin/rotalumis
   sudo chmod u+x /usr/bin/rotalumis
   ```

[5] `https://www.codesynthesis.com/products/xsd/`
[6] `http://www.es.ele.tue.nl/rotalumis/executables/integration/linux/`

3 Windows installation

3.1 Requirements

The following requirements should be met before proceeding with this installation manual.

- Windows 7 or 10, 64-bit
- An user with administration permissions
- Internet connection
- Having the following software installed
  - Visual Studio 2017 Build Tools: `https://www.visualstudio.com/downloads/#build-tools-for-visual-studio-2017`. The following components are required.
    - Visual C++ Build Tools core features
    - VC++ 2017 v141 toolset (x86,x64)
    - Visual C++ 2017 Redistributable Update
    - Windows 10 SDK for Desktop C++ (x86 and x64)
  - Python 3.6: `https://www.python.org/downloads/`
  - Oracle Java SE Development Kit 8: `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
  - CMake: `https://cmake.org/download/`
  - Git: `https://git-scm.com/download`
  - Archiving tool such as WinRAR[7] or 7-zip[8]
- Executables for Python, Java, CMake and Git should be added to the `PATH` variable.

3.2 Installation

This manual describes installation steps to install all dependencies that are required to work with the HLA-DSL into the directory `C:\opt`. If any other installation location is desired, the provided command should be changed accordingly.

1. Create the directories `C:\opt\bin` and `C:\opt\lib` and add them to the Windows `PATH` variable.
2. Create the directory `C:\opt\downloads`.
3. Download, build and install OpenRTI.
   (a) Download the OpenRTI sources from `https://sourceforge.net/projects/openrti/` to `C:\opt\downloads`.
   (b) Extract the downloaded file into this directory.

[7] `https://rarlab.com/download.htm`
[8] `http://www.7-zip.org/download.html`

(c) Create the directory `OpenRTI-0.9.0\build`.

(d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

```
cd OpenRTI-0.9.0\build
```

(e) Use CMake to prepare, build and install OpenRTI. Warnings may be displayed during the compilation, but no errors should occur.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\opt
    -DOPENRTI_BUILD_SHARED=ON
    -DCMAKE_CXX_FLAGS="-DFD_SETSIZE=2048" ..
cmake --build . --config Release --target install
```

The parameter `FD_SETSIZE` increases the maximum amount of federates that may connect to the RTI. If you plan on building smaller or bigger co-simulations, you may adjust this value accordingly.

4. Download, build and install the FMI Library.

(a) Download the FMI Library 2.0.3 sources (*FMILibrary-2.0.3-src.zip*) from `http://www.jmodelica.org/FMILibrary` to `C:\opt\downloads`.

(b) Extract the downloaded file into this directory.

(c) Create the directory `FMILibrary-2.0.3\build`.

(d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

(e) Use CMake to prepare, build and install FMI Library.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DFMILIB_INSTALL_PREFIX=C:\opt
    -DFMILIB_BUILD_TESTS=OFF
    -DFMILIB_GENERATE_DOXYGEN_DOC=OFF ..
cmake --build . --config Release --target install
```

5. Download, build and install the Bullet Physics Library.

(a) Download the Bullet Physics Library sources from `https://github.com/bulletphysics/bullet3/releases` to `C:\opt\downloads`.

(b) Extract the downloaded file into this directory.

(c) Create the directory `bullet3-2.86.1\build`.

(d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

(e) Use CMake to prepare, build and install Bullet.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\opt -DINSTALL_EXTRA_LIBS=ON
    -DINSTALL_LIBS=ON -DBUILD_OPENGL3_DEMOS=OFF
    -DBUILD_PYBULLET=OFF -DBUILD_CPU_DEMOS=OFF
    -DBUILD_BULLET2_DEMOS=OFF -DBUILD_UNIT_TESTS=OFF
    -DUSE_MSVC_RUNTIME_LIBRARY_DLL=ON ..
cmake --build . --config Release --target install
cmake -DBUILD_SHARED_LIBS=ON ..
cmake --build . --config Release --target install
```

6. Download, build and install GLFW.

   (a) Download the GLFW sources from `http://www.glfw.org/` to `C:\opt\downloads`. This manual uses GLFW version 3.2.1, but more recent versions should also work.

   (b) Extract the downloaded file into this directory.

   (c) Create the directory `glfw-3.2.1\build`. When using a more recent version of GLFW, this directory may be slightly different.

   (d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

   (e) Use CMake to prepare, build and install GLFW.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DBUILD_SHARED_LIBS=ON -DCMAKE_INSTALL_PREFIX=C:\opt
    ..
cmake --build . --config Release --target install
```

7. Download, build and install GLM.

   (a) Download the GLM sources from `https://glm.g-truc.net/` to `C:\opt\downloads`.

   (b) Extract the downloaded file into this directory.

   (c) Create the directory `glm\build`.

   (d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

   (e) Use CMake to prepare, build and install GLM.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\opt ..
cmake --build . --config Release --target install
```

8. Download, build and install the XercesC library.

   (a) Download the XercesC sources from `http://xerces.apache.org/xerces-c/download.cgi` to `C:\opt\downloads`. This manual uses version 3.2.0 of XercesC, but more recent versions should also work.

   (b) Extract the downloaded file into this directory.

   (c) Create the directory `xerces-c-3.2.0\build`. When using a more recent version of XercesC, this directory may be slightly different.

   (d) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

   (e) Use CMake to prepare, build and install XercesC.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\opt -DBUILD_SHARED_LIBS=ON
    ..
cmake --build . --config Release --target install
```

9. Download and set the header files of CodeSynthesis XSD.

(a) Download CodeSynthesis XSD for Windows[9] to `C:\opt\downloads`.

(b) Extract the files in the current directory. Move into the resulting directory.

(c) Copy the folder `libxsd\xsd` to `C:\opt\include`.

(d) You should now have a directory `C:\opt\include\xsd\cxx` containing the header files for CodeSynthesis XSD.

10. Download and install the latest Boost libraries.

(a) Download version 1.65.1 of the Boost binary for Visual Studio from `https://sourceforge.net/projects/boost/files/boost-binaries/1.65.1/`. Make sure you download an installer for "**msvc-14.1-64**". This file should have the name *boost_1_65_1-msvc-14.1-64.exe*.

(b) Run the installer and select the installation folder `C:\opt\boost`.

(c) Add the Boost library directory (e.g. `C:\opt\boost\lib64-msvc-14.1`) to the global `PATH` variable.

11. Fetch, build and install the OpenRTI libraries.

(a) Clone the CoHLA project in the `C:\opt\downloads` directory and move into the directory "CoHLA".

```
git clone https://github.com/phpnerd/CoHLA.git
```

(b) Create a new directory called `build` in the `libs` directory. You should now have a directory `libs\build`.

(c) Open the *Developer Command Prompt for VS 2017* and navigate to the directory that you have just created.

(d) Build and install the libraries using CMake.

```
cmake -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\opt\OpenRTI-libs
    -DBOOST_ROOT=C:\opt\boost -DBUILD_SHARED_LIBS=OFF ..
cmake --build . --config Release
cmake -DBUILD_SHARED_LIBS=ON ..
cmake --build . --config Release --target install
```

12. Download the Rotalumis executable[10] and place it in `C:\opt\bin`.

[9] `https://www.codesynthesis.com/download/xsd/4.0/windows/i686/xsd-4.0.0-i686-windows.zip`

[10] `http://www.es.ele.tue.nl/rotalumis/executables/integration/windows/rotalumis.exe`

4 Workspace installation

This section describes how the development environment for CoHLA can be installed.

1. Download one of the Eclipse[11] packages.
2. Extract the downloaded compressed file to a location where you want Eclipse to be or install Eclipse in a location of your choice.
3. Start Eclipse and select a workspace location for starting your co-simulation projects.
4. Close the welcome screen if it pops up.
5. Open the installation wizard by selecting *Install New Software* from the *Help* menu.
6. Add the plug-in update website to the source locations.
   (a) Press *Add*.
   (b) Give the source a proper name, for example "CoHLA".
   (c) Set the location to `https://files.thomasnagele.nl/cohla/plugin`.
   (d) Select the latest CoHLA feature and press *Next*.
   (e) Proceed to the end and press *Finish*. You may be asked to trust the author of the software.
   (f) Eclipse should eventually be restarted to activate the plug-in.

To start a new CoHLA project to describe a co-simulation, simply create a new project as *Project* (in section *General*). When adding a `.cohla`-file, you will be prompted whether to convert the project to a CoHLA project. Agree if you intended on creating a CoHLA project. Upon building of the HLA project new sources are generated in the `src-gen` folder inside your project. This location can be changed using the Eclipse settings.

[11] `https://www.eclipse.org/downloads/eclipse-packages/`

5 Sample project

The CoHLA project contains a sample project called the RoomThermostat. This project was already downloaded in steps 7 and 11 for Linux and Windows respectively. The RoomThermostat project is a co-simulation of a sample domestic thermostat system for maintaining the temperature in a number of rooms. The sources for the project are located in the *RoomThermostat* directory. The following steps describe how to create a CoHLA project from these sources and how to run a co-simulation.

1. Start your Eclipse instance with the CoHLA plugin installed and select a workspace location for your co-simulation projects, similar to what was described in Section 4.
2. Create a new project. Make sure that it is a General Project as listed in the wizard.
3. Enter "RoomThermostat" as project name and press *Finish*.
4. Copy the RoomThermostat sources from the CoHLA project into the project using a file manager. The project should now contain a directory called "models" and two files having the `.cohla` file extension.
5. Refresh the Package Explorer in Eclipse by pressing `F5`.
6. Open one of the two source files, either `House.cohla` or `orti.cohla`.
7. Eclipse should now ask whether to convert the project to an Xtext project or not. Press *Yes* to convert the project.
8. Eclipse automatically generates the source files for the co-simulation upon saving the main file, which is `House.cohla` for this project. The generated files are located in the `src-gen` directory in the project.
9. Open a terminal (Linux) or *Developer Command Prompt for VS 2017* (Windows) and move into the directory containing the generated sources: `RoomThermostat/src-gen/House`.
10. Build the sources using the run script. Warnings may be displayed, but the script should return without an error.

```
./run.py -bv
```

11. Use the run script to run the co-simulation with the *ComfyBase* situation applied to the simulations.

```
./run.py -t conf/house.topo -s conf/House/ComfyBase.situation
    -e
```

12. Depending on the speed of your system, the simulation may run for a while. When the co-simulation finished, the logs – including a CSV file – can be found in the directory `RoomThermostat/src-gen/House/logs/<timestamp>`.