ROBMOSYS-1FORC

ROQME

DEALING WITH NON-FUNCTIONAL PROPERTIES THROUGH GLOBAL ROBOT QUALITY OF SERVICE METRICS

RoQME Tool-Chain - Installation & User Manual

(v1.0_201904030901)

THIS PROJECT HAS RECEIVED FUNDING FROM THE *EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME* UNDER GRANT AGREEMENT NO. 732410, IN THE FORM OF FINANCIAL SUPPORT TO THIRD PARTIES OF THE ROBMOSYS PROJECT

Content

Pre-Installation (video tutorial available here) 3

Using RoQME within SmartMDSD (video tutorial available here) 8

IMPORTANT: The RoQME Tool-Chain is available in GitHub as a free and open source Eclipse plugin to be installed within the RobMoSys SmartSoft Tool-Chain. It can be downloaded at: robmosys-roqme-itp/downloads/RoQME-Eclipse-Tools-v1.0_201904030901.zip

RoQME Installation Instructions

Pre-Installation (video tutorial available here)

1. Download the SmartSoft virtual machine from:
   https://wiki.servicerobotik-ulm.de/virtual-machine

2. Import and run the virtual machine using, e.g., Virtualbox (https://www.virtualbox.org/).

   IMPORTANT: All the instructions included next must be executed on the SmartMDSD virtual machine

3. Enable "copy & paste" and "drag & drop" between your host OS and the virtual OS:

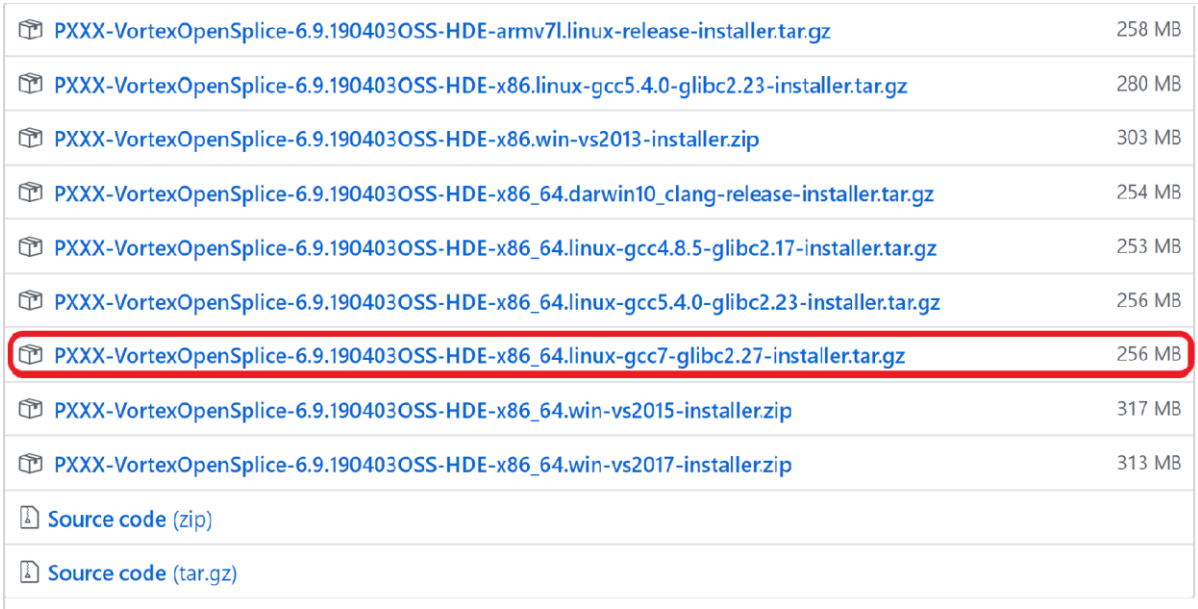
   a) Devices -> Insert Guest Additions CD Image

   b) Devices -> Share Clipboard -> Bidirectional

   c) Devices -> Drag and Drop -> Bidirectional

   d) Restart the virtual machine to get the "Guest additions" running

4. Download OpenSliceDDS Community Edition from:
   https://github.com/ADLINK-IST/opensplice/releases



5. Create a new folder "OpenSpliceDDS" into "/home/smartsoft"

6. Extract the file downloaded in (4) into the folder created in (5)

7. Open the “release.com” file available at:
"/home/smartsoft/OpenSpliceDDS/HDE/x86_64.linux/release.com"

Original file

```
1       if [ "${SPLICE_ORB:=}" = "" ]
2       then
3       SPLICE_ORB=DDS_OpenFusion_1_6_1
4       export SPLICE_ORB
5       fi
6       if [ "${SPLICE_JDK:=}" = "" ]
7       then
8       SPLICE_JDK=jdk
9       export SPLICE_JDK
10      fi
11      if [ -n "${BASH_VERSION}" ]
12      then
13      OSPL_HOME="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
14      else
15      echo "Please manually set OSPL_HOME to the install directory."
16      fi
17      PATH=$OSPL_HOME/bin:$PATH
19      LD_LIBRARY_PATH=$OSPL_HOME/lib${LD_LIBRARY_PATH:+:}$LD_LIBRARY_PATH
20      CPATH=$OSPL_HOME/include:$OSPL_HOME/include/sys:${CPATH:=}
21      if [ "${OSPL_URI:=}" = "" ]
22      then
23      OSPL_URI=file://$OSPL_HOME/etc/config/ospl.xml
24      export OSPL_URI
25      fi
26      OSPL_TMPL_PATH=$OSPL_HOME/etc/idlpp
27      . $OSPL_HOME/etc/java/defs.$SPLICE_JDK
28      export OSPL_HOME PATH LD_LIBRARY_PATH CPATH OSPL_TMPL_PATH VORTEX_DIR
```

8. Add line 17 as follows:

Edited file

```
1         if [ "${SPLICE_ORB:=}" = "" ]
2         then
3         SPLICE_ORB=DDS_OpenFusion_1_6_1
4         export SPLICE_ORB
5         fi
6         if [ "${SPLICE_JDK:=}" = "" ]
7         then
8         SPLICE_JDK=jdk
9         export SPLICE_JDK
10      fi
11      if [ -n "${BASH_VERSION}" ]
12      then
```

```
   13     OSPL_HOME="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
   14     else
   15     echo "Please manually set OSPL_HOME to the install directory."
   16     fi
>> 17     OSPL_HOME="/home/smartsoft/OpenSpliceDDS/HDE/x86_64.linux"
   18     PATH=$OSPL_HOME/bin:$PATH
   19     LD_LIBRARY_PATH=$OSPL_HOME/lib${LD_LIBRARY_PATH:+:}$LD_LIBRARY_PATH
   20     CPATH=$OSPL_HOME/include:$OSPL_HOME/include/sys:${CPATH:=}
   21     if [ "${OSPL_URI:=}" = "" ]
   22     then
   23     OSPL_URI=file://$OSPL_HOME/etc/config/ospl.xml
   24     export OSPL_URI
   25     fi
   26     OSPL_TMPL_PATH=$OSPL_HOME/etc/idlpp
   27     . $OSPL_HOME/etc/java/defs.$SPLICE_JDK
   28     export OSPL_HOME PATH LD_LIBRARY_PATH CPATH OSPL_TMPL_PATH VORTEX_DIR
```

9. Once edited, copy the contents of the previous file at the end of the .bashrc file available at: "/home/smartsoft". Note that this file is hidden by default. To make hidden files visible press Ctrl-H.

10. Open a terminal window and execute the following command to build the DDS library:

```
make -C $OSPL_HOME/custom_lib -f Makefile.Build_DCPS_ISO_Cpp2_Lib
```

11. Open a new terminal, move to the "home/smartsoft" directory, and execute the following command to clone the GitHub repository:

```
git clone https://github.com/DOCGroup/MPC
```

12. Open again the .bashrc file, available at: "/home/smartsoft", and include the following line at the end:

```
export MPC_ROOT="/home/smartsoft/MPC"
```

13. Open a new terminal and execute the following command to create a new environment variable and update the .bashrc file:

```
source .bashrc
```

14. RoQME uses GTK versión 2. In order to move from GTK 3 to GTK 2, open the "eclipse.ini" file available at: "/home/smartsoft/SOFTWARE/SmartMDSD-Toolchain-v3.7/":

Original file

```
    1       -startup
    2       plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
    3       --launcher.library
    4       plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.551.v20171108-1834
    5       -product
    6       org.eclipse.epp.package.modeling.product
    7       -showsplash
    8       org.eclipse.epp.package.common
    9       --launcher.defaultAction
   10       openFile
   11       --launcher.appendVmargs
   12       -vm
   13       /usr/lib/jvm/java-8-openjdk-amd64/bin
   14       -vmargs
   15       -Dosgi.requiredJavaVersion=1.8
   16       -Dosgi.instance.area.default=@user.home/eclipse-workspace
   17       -XX:+UseG1GC
   18       -XX:+UseStringDeduplication
   19       --add-modules=ALL-SYSTEM
   20       -Dosgi.requiredJavaVersion=1.8
   21       -Xms256m
   22       -Xmx1024m
   23       --add-modules=ALL-SYSTEM
   24       -Declipse.p2.max.threads=10
   25       -Doomph.update.url=http://download.eclipse.org/oomph/updates/milestone/latest
   26       -Doomph.redirection.index.redirection=index:/-
>http://git.eclipse.org/c/oomph/org.eclipse.oomph.git/plain/setups/
```

15. Copy lines 11 and 12 and save the file:

Edited file

```
    1       -startup
    2       plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
    3       --launcher.library
    4       plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.551.v20171108-1834
    5       -product
    6       org.eclipse.epp.package.modeling.product
    7       -showsplash
    8       org.eclipse.epp.package.common
    9       --launcher.defaultAction
   10       openFile
>> 11       --launcher.GTK_version
>> 12       2
   13       --launcher.appendVmargs
   14       -vm
   15       /usr/lib/jvm/java-8-openjdk-amd64/bin
   16       -vmargs
   17       -Dosgi.requiredJavaVersion=1.8
   18       -Dosgi.instance.area.default=@user.home/eclipse-workspace
   ...      ...
```

16. Installation of the RoQME Plugin:

    a) Download the RoQME plugin (RoQME-Eclipse-Tools-vXXXX) from: https://github.com/roqme/robmosys-itp/blob/master/downloads/

    b) Create a new "RoQME" folder at "/home/smartsoft" and extract the file previously downloaded in (a)

    c) Launch Eclipse and install the RoQME plugin in Eclipse, by selecting the following menu options:

    Help - > Install New Software... - > Add... - > Local... - > {Repository path}

    d) Restart Eclipse

Using RoQME within SmartMDSD (video tutorial available here)

1. Creating a new RoQME Project:

   a) ***File - > New - > Other -> RoQME Project***

   b) Choose the project name and click the ***Next*** button

   c) Click on ***Load Repository*** and choose the required contexts. For instance, to include the information provided by the bumper as a context, choose the *CommBasicsObjects.BumperEventService* service, available at: "/home/smartsoft/SOFTWARE/smartsoft/repos/DomainModelsRepositories/CommBasicsObjects/model/CommBasicObjects.service"

   d) Browse the services in the tree model until you find the bumper and select: *CommBasicsObjects.BumperEventService: new State[Enum]*

   e) Once selected, press the ***Add context*** button

   f) Double click on the name of the context and change it from *context0* to *bump*

   g) Repeat steps (c) – (f) for each context that needs to be included as an input in the RoQME model

   h) Click the ***Finish*** button

   IMPORTANT: New RoQME Projects include two files: (1) a ***.roqme model***, including the definitions of the contexts relevant for the current project; and (2) a ***.roqmemap*** model that binds the previous contexts with the corresponding Smartsoft services (this mapping is created during the project configuration: steps (c) – (f)).

2. Extending the .roqme model. Once initialized as described in step 1, the RoQME model can be extended to include relevant (non-functional) properties, observations, variables, etc. Next, you can find an example .roqme model including:

   a) Three contexts: *bump*, which is a primitive context (the one directly provided by the corresponding Smartsoft service according to the mapping created in step 1.d); and *bumpEvent* and *bumpCount*, which are two additional contexts derived from *bump*. **IMPORTANT**: adding new primitive contexts requires appropriately updating the corresponding .roqmemap model!!

   b) The *safety* property that takes a default value of 1; and

c) Two observations: *o1* => safety is undermined every time a bumpEvent is detected; and *o2* => safety is (veryhigh) undermined if more than 5 bumpEvents are detected within 1 hour.

```
roqme RoQME_Example

context bump : enum {BUMPER_UNKNOWN, BUMPER_NOT_PRESSED, BUMPER_PRESSED}
context bumpEvent : eventtype := bump::BUMPER_PRESSED
context bumpCount : number := count(bumpEvent,1hour)

property safety reference 1

timer t := 1 hour
observation o1 : bumpEvent undermines safety
observation o2 : t while (bumpCount >5) undermines safety veryhigh
```

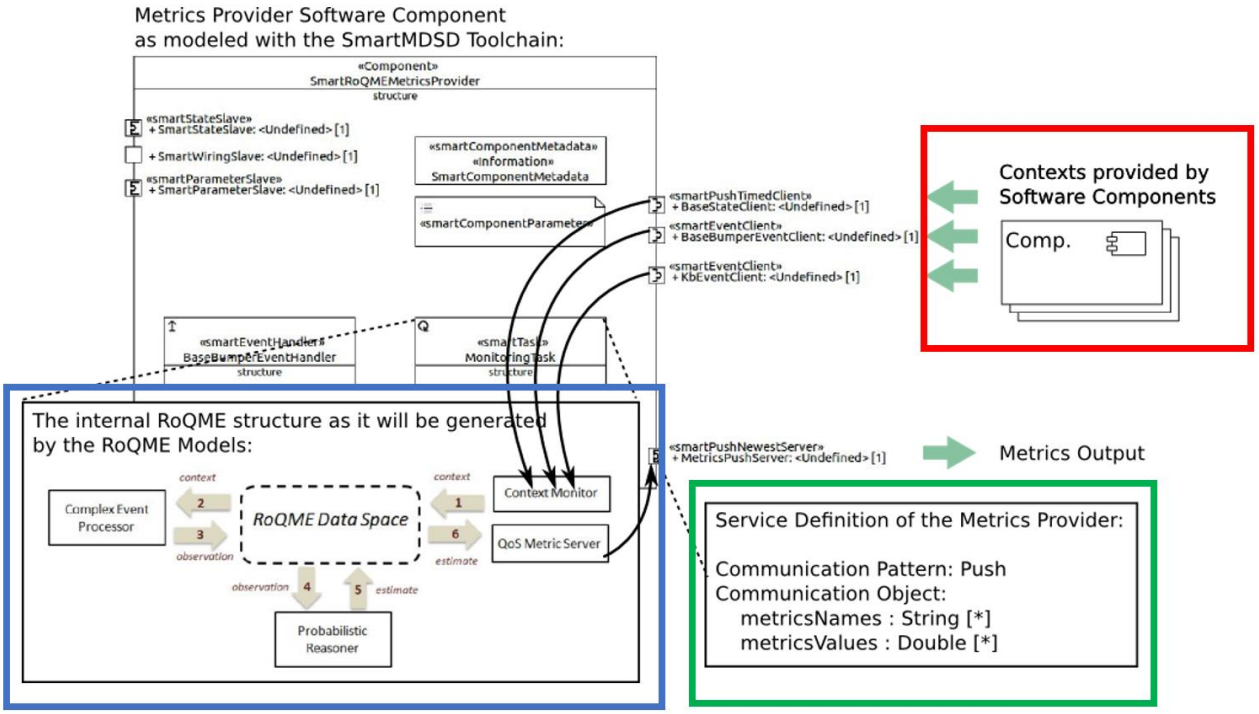
3. Generating the RoQME QoS metric provider component model.

In order to generate the RoQME artifacts from the previous models, right click on the .roqmemap file and select:

RoQME - > Generate Metrics Provider

This generation process will create two projects: (1) a ***QoSMetricProvider project***; and (2) ***CommRoqmeEstimate project***. The former includes the generated Smartsoft component model for the QoS Metric Provider. Smartsoft will use this model as an input to generate the corresponding software. The latter includes the definition of the communication objects used by the QoS metric provider component to communicate the estimated metrics to other components.

The following figure illustrates the structure of the generated Smartsoft component model, including (1) the inputs obtained from the external context providers (red square); (2) the three main processes carried out by the component (i.e., context monitoring, event processing, and probabilistic reasoning) connected through the so called RoQME Data Space (blue square); and (3) the structure of the communication objects used to provide the metrics to other components (green square).



4. Compiling the generated C/C++ code.

   Both the RoQME Data Space (based on DDS) and the probabilistic reasoner are implemented in C++ and, thus, need to be compiled following the steps indicated next:

   - Copy the following code at the end of the .bashrc file, available at: "/home/smartsoft", in order to create the ROQME_ROOT, LD_LIBRARY_PATH and CPATH environment variables:

```
export ROQME_ROOT="/home/smartsoft/workspaces/SmartMDSD-Toolchain/QoSMetricProvider/roqme-dds/cpp"
export LD_LIBRARY_PATH=$ROQME_ROOT/roqmeDDS/lib:$LD_LIBRARY_PATH
export CPATH=$ROQME_ROOT/roqmeDDS/include:$ROQME_ROOT/roqmeDDS/idl:$CPATH
```

   FOR THE DDS Library

   - Move to the generated "roqme-dds" project, available at: "/home/smartsoft/workspaces/SmartMDSD-Toolchain/QoSMetricProvider/roqme-dds/cpp/roqmeDDS", open a new terminal, and execute the following commands:

```
chmod +777 gen_makefile.sh
./gen_makefile.sh
make
```

FOR THE Probabilistic Reasoner

- Move to the generated "reasoner" project, available at: "/home/smartsoft/workspaces/SmartMDSD-Toolchain/QoSMetricProvider/reasoner", open a new terminal, and execute the following commands:

```
chmod +777 gen_makefile.sh
./gen_makefile.sh
make
```

Et voilá! The generated RoQME component can now be used within any Smartsoft architecture as a QoS metrics provider.