Orocos data logging and visualization

Best Practices in integrating complex robotic systems

Gianni Borghesan
KU Leuven
Febraury 2020

Reporters I

Role of the components

- Save (or send) data
- From real time parts, without hampering them.

Reporters II

Common interface

- `reportComponent(string const& Component) :bool`
- `reportData(string const& Component, string const& Data) :bool`
- `reportPort(string cons& Component, string const& Port) :bool`
- `snapshot( ) :void`

Reporters properties

```
bool WriteHeader    = true
       (Set to true to start each report with a header.)
bool Decompose      = true
       (Set to false in order to not decompose the port data. The marshaller must be able to handle
       this itself for this to work.)
bool Snapshot       = false
       (Set to true to enable snapshot mode. This will cause a non-periodic reporter to only report
       data upon the snapshot() operation.)
bool Synchronize    = false
                (Set to true if the timestamp should be synchronized with the logging)
PropertyBag ReportData     = 1 Properties
        (A PropertyBag which defines which ports or components to report.)
ConnPolicy ReportPolicy   = {type = 0, size = 0, lock_policy = 2, init = true, pull = false,
       buffer_policy = 1, mandatory = true, transport = 0, data_size = 0, name_id =  }
       (The ConnPolicy for the reporter's port connections.)
bool ReportOnlyNewData = false
  (Turn on in order to only write out NewData on ports and omit unchanged ports. Turn off
  in order to sample and write out all ports (even old data).)
```

Reporters types I

File reporter

- Save data to a file
- ascii, tab separated values

Netcdf Reporter

- Save data to a file
- binary, use NetCDF (Network Common Data Form)

Reporters types II

Tcp Reporter

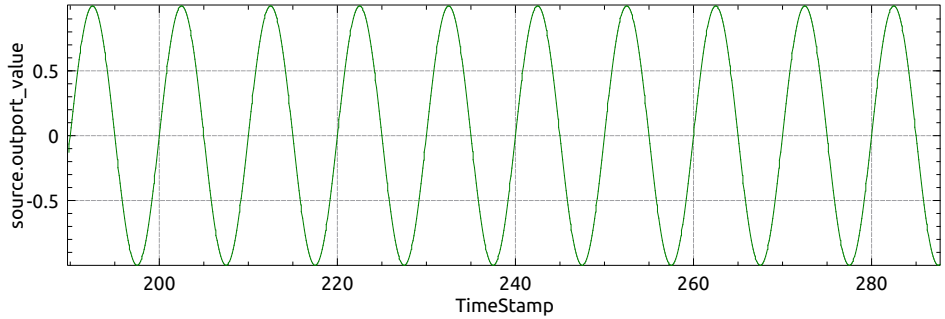
- Create a server that sends the data encoded in strings over TCP/IP
- Needs some commands from the other side.

File and NetCDF reporters

- They also have a property to set the file name, `ReportFile`
- the data is time stamped
- we can visualise the data with `kst2`

kst I

Is a program that allows for data visualization.
https://kst-plot.kde.org/



kst II

- reads boths formats but works better with NetCDF
- really a lot of options, but mainly you can plot a variable against time or against other variables,
- can handle several plots in a single window
- you can save the configuration in a `.kst` file
- allows also some annotations for generating a figure

TCP Reporter I

- It opens a TCP server, needs some specific command to be sent to start working
  - `VERSION 1.0`
  - `HEADERS`
  - `SUBSCRIBE 'dataname'`
- https://orocos.github.io/ocl/toolchain-2.9/api/html/classOCL_1_1TcpReporting.html

TCP Reporter II

To test: use the netcat command: e.g. `nc localhost 3145`

```
>  nc localhost 3142
100 Orocos 1.0 TcpReporting Server 1.0
VERSION 1.0
101 OK
HEADERS
305 TimeStamp
305 source.outport_value
305 TimeStamp
305 source.outport_value
306 End of list
SUBSCRIBE source.outport_value
302 source.outport_value
201 0 -- begin of frame
202 source.outport_value
205 0.992362
203 0 -- end of frame
201 1 -- begin of frame
202 source.outport_value
```

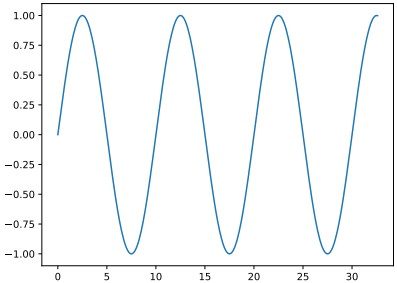
Example of deployment

```
import ("exe1")
loadComponent("source","source2")
source.setPeriod(0.1)
source.configure
source.amplitude=1
source.frequency=0.1

loadComponent("reporterN","OCL::NetcdfReporting")
loadComponent("reporterF","OCL::FileReporting")
loadComponent("reporterT","OCL::TcpReporting")
connectPeers ("source","reporterN")
connectPeers ("source","reporterF")
connectPeers ("source","reporterT")
reporterN.reportPort("source","outport_value")
reporterF.reportPort("source","outport_value")
reporterT.reportPort("source","outport_value")
reporterN.setPeriod(0.1)
reporterF.setPeriod(0.1)
reporterT.setPeriod(0.1)

reporterN.start()
reporterF.start()
reporterT.configure()
reporterT.start()
source.start()
```

Reading NetCDF files from python I

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#sudo apt-get install python-netcdf4 (or python3)

from netCDF4 import Dataset
import matplotlib.pyplot as plt

fh = Dataset('reports.nc', mode='r')
t=fh['TimeStamp'][:]
x=fh['source.outport_value'][:]

fh.close()
plt.plot(t,x)
plt.show()
```

Reading NetCDF files from python II



code so far.. https://github.com/gborghesan/exe1.git