

"Port Monitor": progress & open questions

Torsten Wilde and James Kohl
Oak Ridge National Laboratory

CCA Forum Quarterly Meeting
Santa Fe, NM ~ October 2003



4 Or MY FIRST REAL CCA/CCAFAE EXPOSURE

- 4 Create a simple component application with PortMonitor from scratch
 - 4 Using a very simple getSum ($A+B=C$) component
 - 4 Built a PortMonitor for the getSum port and plug-in-pray
 - 4 Starting with an object oriented application
 - 4 Converting the object design step by step into a component design to get the feel of what is needed

- 4 No problem installing CCA stuff on Mandrake (after fixing configure errors by hand in Babel & Ccafe)
 - 4 using sources (configure with prefix's like rpm installations)
 - 4 and 'make install'

4 Difficult to start a CCA application project from scratch

- 4 Necessary information is distributed over a lot of documentation and guides
 - 4 A short compilation of steps and operations is needed
- 4 Hard to choose the appropriate source tree & install structure for the project
 - 4 Maybe a standard is needed?
- 4 A lot of support structure has to be built from scratch
 - 4 Scripts for building code from sidl files in appropriate locations
 - 4 Macros for finding other stuff (depending on the chosen file dependencies and install structure)
 - 4 Build structure, including configure- & Make-files

- 4 Using Automake, Autoconf, Libtools & Autoheader makes building & maintaining really easy (for a new component/application)
 - 4 That's why a well defined build and install structure is needed
- 4 Shared library dependencies are hell ☺
 - 4 Trade-off shared library efficiency (resource usage) for software build complexity
 - 4 Using too many makes it hard to keep track of and hard to make work correctly
 - 4 Define what is installed from whom in a standard distribution
 - 4 Eg. Babel XML repository, ccafe XML repository
 - 4 Client side sidl file and/or libraries
 - 4 Tutorial examples are currently restructured by Wael & Boyana
- 4 Having an experienced component guy available helps a lot as well (thank you Wael ☺)



Potential Install Tree Structure 1

4 CCA/

4 Ports repository

4 Components repository

4 Port Monitor repository

4 Application repository

4 Libs

4 Cca-ports

4 Port client libs

4 Includes

4 Cca-ports

4 Port header files

- dependency structure

- ports

- Components need port definition

- Port monitors need port definition

- Applications use components & portmonitors

Potential Install Tree Structure 2

4 Ports repository

- 4 Xml port repository
- 4 Port 1 dir
- 4 ...
- 4 Port N dir
 - 4 Port sidl file

4 Components

- 4 Component 1 dir
- 4 ...
- 4 Component N dir
 - 4 Component library
 - 4 Component cca file



Potential Install Tree Structure 3

4 Port Monitor repository

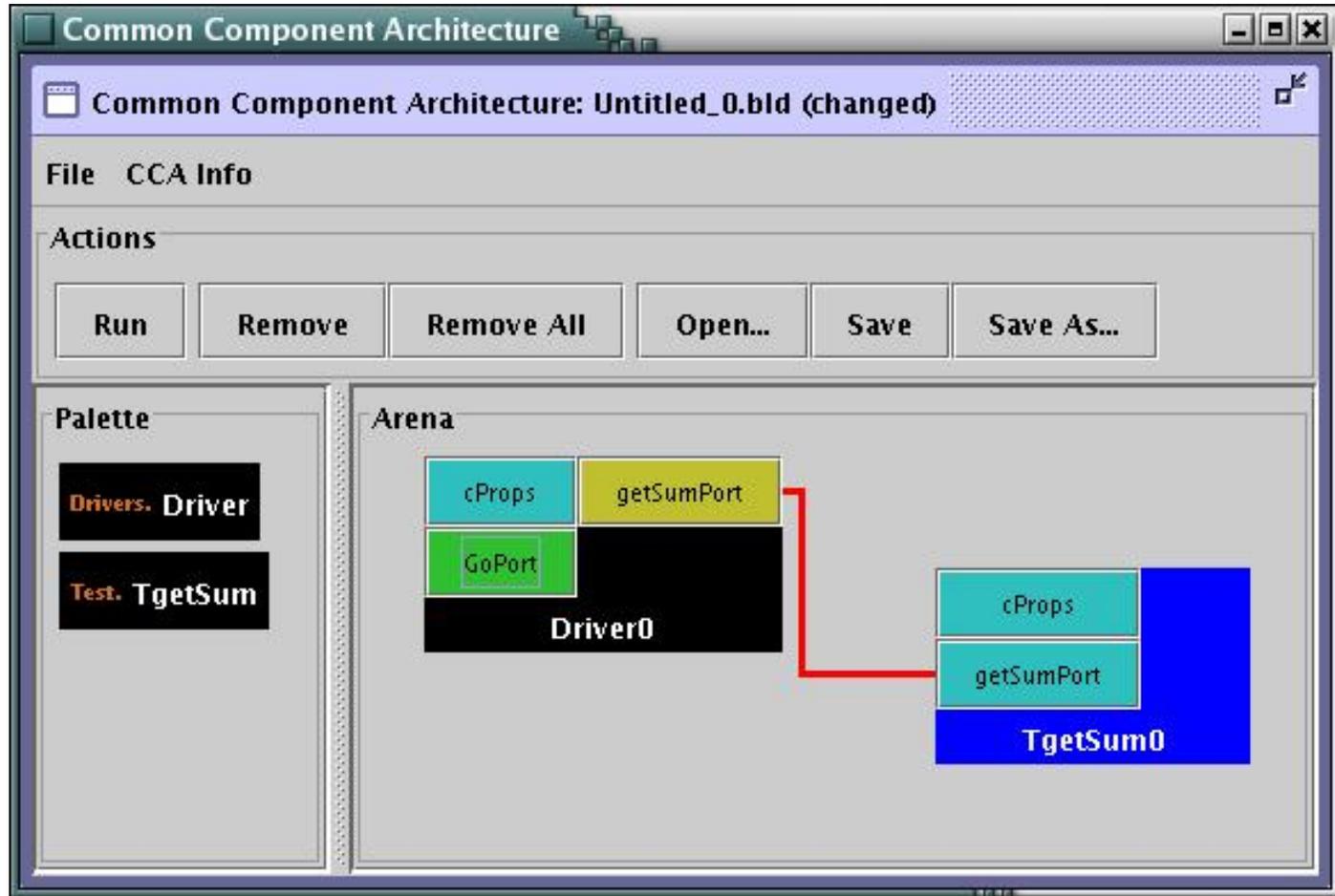
- 4 Port monitor 1 dir
- 4 ...
- 4 Port monitor N dir
 - 4 Port monitor library
 - 4 Port monitor cca file

4 Application repository

- 4 Application 1 dir
- 4 ...
- 4 Application N dir
 - 4 Ccafe scripts
 - 4 Application driver
 - 4 Driver cca file
 - 4 Driver libraries



Trivial Component Application



4 Simple calculation of the sum of two integer inputs



```
wilde@dumpster:~/ssh
File Edit View Terminal Go Help
Loading /home/wilde/tmp//cca/apps-repository/app-Test-ge
Loading /usr/local/cca/components/Test_getSum/libTest_ge
pulldown Test.TgetSum TgetSum0
gui:"display component TgetSum0

gui>addProvidesPorts TgetSum0 cProps ::classic::gov::cca

gui>addUsesPorts TgetSum0
rows = 2

gui>gui:"connect Driver0 getSumPort TgetSum0 getSumPort
connect Driver0 getSumPort TgetSum0 getSumPort

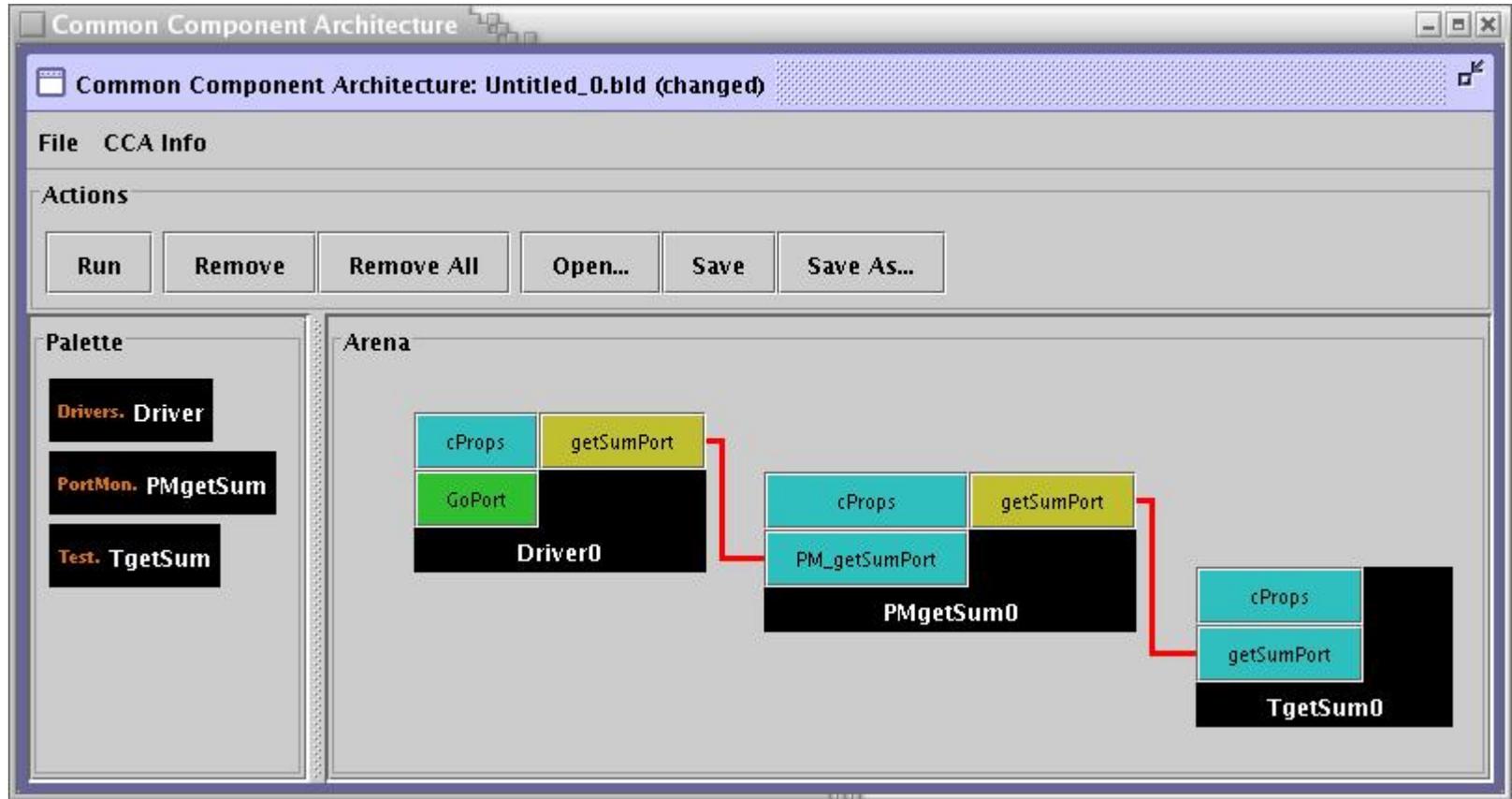
gui>gui:"go Driver0 GoPort
result :0+0 = 0
##specific go command successful

gui>
```

4 Hey, $0+0=0$ what a blast 😊



Component Application With PortMonitor



4 With added PortMonitor component (custom-designed for port)



```
wilde@dumpster:~/ssh
File Edit View Terminal Go Help
connect PMgetSum0 getSumPort TgetSum0 getSumPort

gui>

gui:"go Driver0 GoPort

Method 'getSum' called!
To proceed press a key!
Value 1 is: 0
Value 2 is: 0!
Returned value is: 0
result :0+0 = 0
##specific go command successful

gui>
```

4 Cool, $0+0$ is still 0 (PortMonitor didn't break anything 😊)

Next Functionality Priority List - Part1

4 Needed functionality (priority can be discussed):

4 Writing port invocation trace files

- 4 Error analysis, port invocation replay & replay with different input values

- 4 Creating standard input format for black-box testing

4 Interface with the port monitor component for interactive functionality

- 4 Use GUI component inside or outside the component framework

- 4 But what to do in batch mode?

- 4 Communicate with user GUI via communication port, outside the scope of the component framework



Next Functionality Priority List - Part2

- 4 Providing user input for:
 - 4 Toggling trace file generation (even selecting what to save in file)
 - 4 Selecting what to monitor (specific function calls on a port, which ports ...)
 - 4 Toggling lock-step functionality
 - 4 Changing input parameter before resuming execution

- 4 How to support debugging better?
 - 4 Start extern debugger when entering suspicious component
 - 4 How to automatically pause code for debugger attachment?



Action list - Feedback appreciated

4 Selecting trace file storage format

- 4 Good to select an established format that allows use of existing tools for analysis
- 4 Storage requirements (compression) depend on application size, frequency of invocation and amount of data transferred via port calls
 - 4 Dynamically choose to store big data packed (increased complexity)
- 4 Readable & editable for parameter changes (Ascii vs. binary)
- 4 Trace similar to performance trace
 - 4 Maybe TAU provides a good starting point
 - 4 Might switch to Vampire format (Epilog)



Action list - Feedback appreciated

4 Serializing ports

- 4 Support passing of objects via port method invocation
- 4 Needs support from port or object itself
 - 4 Get_serialized_me method
 - 4 Set_serialized_me method