Revisiting BuilderService & AbstractFramework Interfaces & Component Building Issues

Wael R. Elwasif
Oak Ridge National Laboratory
CCA-Forum Meeting
Eugene, OR, July 2003

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Contents

- The current Interfaces.
- Proposed Additions.
- If we build them they will come, but ***how do we build them***???

BuilderService

- **createInstance**(***instanceName***, ***className***, ***properties***)
- **getComponentIDs**()
- **getComponentProperties**(cca.ComponentID ***cid***)
- **setComponentProperties**(*cid*, cca.TypeMap ***map***)
- cca.ComponentID **getDeserialization**(string **s**)
- **getComponentID**(string ***componentInstanceName***)
- **destroyInstance**(cca.ComponentID ***toDie***, float ***timeout***)
- **getProvidedPortNames**(***cid***)
- **getUsedPortNames**(***cid***)
- **getPortProperties**(***cid***, string ***portName***)
- **setPortProperties**(***cid***, ***portName***, ***map***)
- **connect**(***user***, ***usingPortName***, ***provider***, ***providingPortName***)
- **getConnectionIDs**(array<cca.ComponentID> *componentList*)
- **getConnectionProperties**(cca.ConnectionID ***connID***)
- **setConnectionProperties**(***connID***, cca.TypeMap ***map***)
- **disconnect**(***connID***, ***timeout***)
- **disconnectAll**(***cid1***, ***cid2***, ***timeout***)

BuilderService Breakdown

- **createInstance(*instanceName*, *className*, *properties*)**
- **destroyInstance**(cca.ComponentID ***toDie***, float ***timeout***)

Component Lifetime

- **getComponentProperties**(cca.ComponentID ***cid***)
- **setComponentProperties**(***cid***, cca.TypeMap ***map***)

Component Properties

- cca.ComponentID **getDeserialization**(string **s**)
- **getComponentIDs()**
- **getComponentID**(string ***componentInstanceName***)
- **cca.ComponentID getComponentsByProperty**(in Type ***map***)

Component ID's

BuilderService Breakdown (Cont.)

getProvidedPortNames(***cid***)
getUsedPortNames(***cid***)
getPortProperties(***cid***, string ***portName***)
setPortProperties(***cid***, ***portName***, ***map***)

Ports

connect(***user***, ***usingPortName***, ***provider***, ***providingPortName***)
getConnectionIDs(array<cca.ComponentID> ***componentList***)
getConnectionProperties(cca.ConnectionID ***connID***)
setConnectionProperties(***connID***, cca.TypeMap ***map***)
disconnect(***connID***, ***timeout***)
disconnectAll(***cid1***, ***cid2***, ***timeout***)

Connections

BuilderService Issues

- What are “component properties” good for??
  - Suggest: add
  - • **cca.componentID getComponentsByProperty**(in Type ***map***)
- How does setPortProperties() interact with registerUsesPort() and addProvidesPort()??
- What about getPortByProperty() iin BuilderService??
- What about getConnIDsByProperty()??
- What is the CCA “official” type for the BuilderService port??. Is it “gov.cca.ports.BuilderService“ or “gov.cca.BuilderService“ (a la ccaffeine)??

AbstractFramework

- gov.cca.TypeMap **createTypeMap**()
- getServices(string ***selfInstanceName***, string ***selfClassName***,
  TypeMap ***selfProperties***)
- **releaseServices**(in Services ***svc***)
- void **shutdownFramework**()
- AbstractFramework **createEmptyFramework**()

Issues with AbstractFramework

- What's with the name??
  (gov::cca::AbstractFramework::_create() ??)
- Runtime framework configuration:
  - Components locations. Suggest adding addComponentPath(in string path) (or something similar)
- Should the Framework provide a simple (or maybe not so simple) dictionary/map/... service (maybe similar to PVM's mbox).
- What else???

(Babel) Components Build Issues

- The Issue:
  Component Code X (Impls/Server) in Language A needs to link against client side code (in language A) of all the *ports* that X uses. How to make that process easy(ier)??

- Possible Approaches:
  - Generate the client side ports code along with the server code for X (a la the tutorial)

```
.integrator-component-f90: integrator.sidl cca.sidl
    $(BABEL) --server=f90 --repository-path=repository \
    --output-directory=integrator-f90  --exclude='^gov.*' --exclude='^SIDL.*' \
    --suppress-timestamp integrators randomgen.RandomGenerator \
                                     functions.Function
```

Build Issues (Continued)

- Pros:
  - All in one place. No need to find and link against external libs.
- Cons:
  - Confusing (are we generating server code or client code for the used ports??).
  - Redundancy: same code will be generated for every component that uses those ports (see effect in older version of the tutorial).

More Approaches

- Modify the Babel command line interface (GASP!!!!)

```
.integrator-component-f90: integrator.sidl cca.sidl
    $(BABEL)  --repository-path=repository \
    --output-directory=integrator-f90  --exclude='^gov.*' --exclude='^SIDL.*' \
    --suppress-timestamp –server=f90 integrators \
    -client randomgen.RandomGenerator  functions.Function
```

- Pros:
  - No confusion (this actually ***IS*** important).
- Cons:
  - Still too many replicated files.

Yet more approaches

- One client library ***per port per language.*** (a la the latest tutorial python driver)

```
.integrator-client-python: integrator.sidl cca.sidl
      $(BABEL) --client=python --repository-path=repository \
 --output-directory=integrator-client-python  --exclude='^gov.*' --exclude='^SIDL.*' \
--suppress-timestamp integrators.Integrator
........
.driver-component-python: driver.sidl cca.sidl
      $(BABEL) --server=python --repository-path=repository \
  --output-directory=driver-python  --exclude='^gov.*' --exclude='^SIDL.*' \
      --suppress-timestamp tutorial
```

More on the third approach

- Pros:
  - Minimum or no code replication.
  - “Intuitive” (corresponds closely to **#include**'s)
- Cons:
  - One client library ***per port per language.***
- Bottom line: *Needs well defined structure for multi-language components build directories*.
- **Personal preference: #3,** now about that directory structure !!!!.