



A Simple CCA Component Application

CCA Forum Tutorial Working Group

[http://www.cca-forum.org/tutorials/
tutorial-wg@cca-forum.org](http://www.cca-forum.org/tutorials/tutorial-wg@cca-forum.org)



Module Overview

- What the example does: the math.
- From math to components: the architecture.
- The making of components: inheritance and ports.
- Framework-component interactions.
- Putting it all together: the CCafeine ways.
- The application in action.

Goals

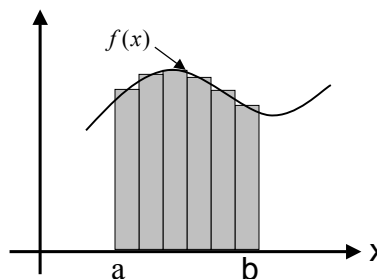
To show how CCA components are used to build an application to numerically integrate a continuous function using two different integration techniques.

3

The Math: Integrator (1)

The midpoint numerical integrator

$$\int_a^b f(x) dx \approx \frac{b-a}{n} \sum_{j=1}^n f\left(\frac{x_{j-1} + x_j}{2}\right)$$



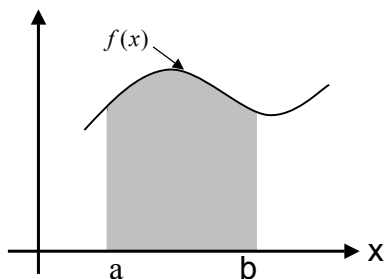
4

The Math: Integrator (2)

The Monte Carlo integrator

$$\int_a^b f(x) dx \approx \frac{1}{b-a} \left(\frac{1}{N} \sum_{i=1}^N f(x_n) \right)$$

x_n Uniformly distributed in $[a, b]$



5

The math: Functions

Linear Function

$$f_1(x) = 2x$$

Nonlinear Function

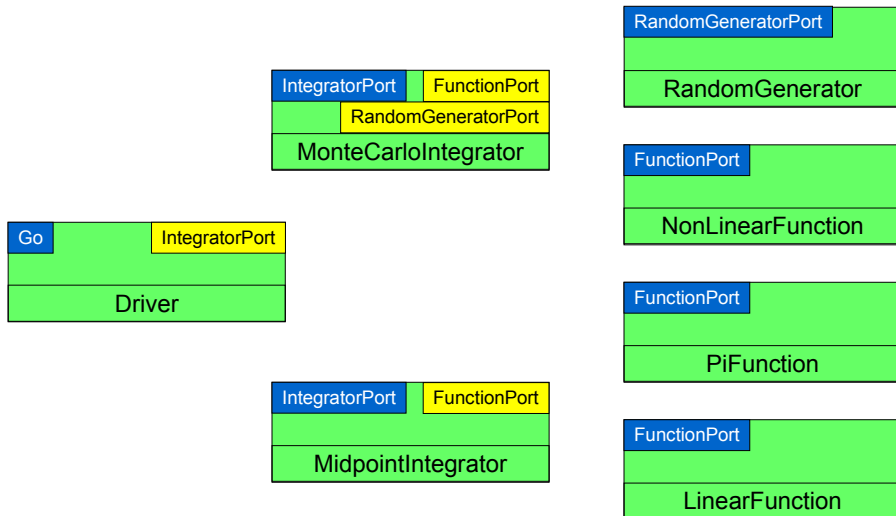
$$f_2(x) = x^2$$

Pi Function

$$f_3(x) = \frac{4}{1+x^2}$$

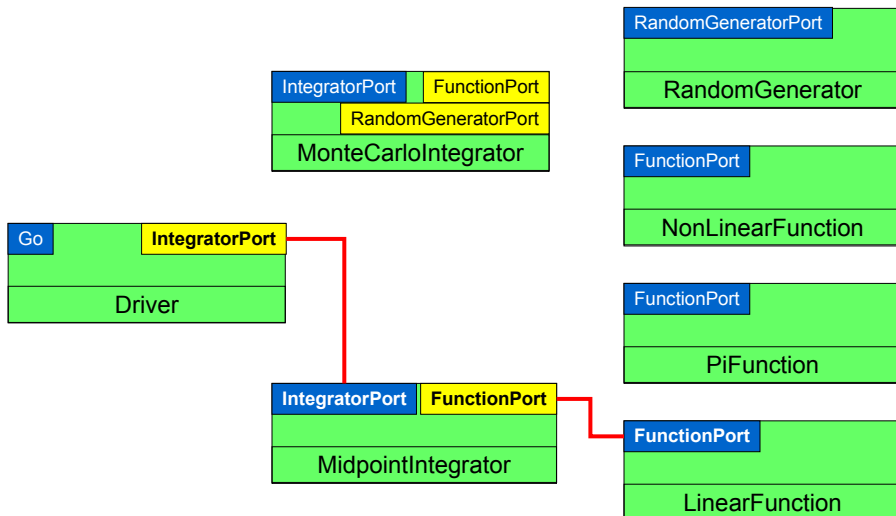
6

Available Components



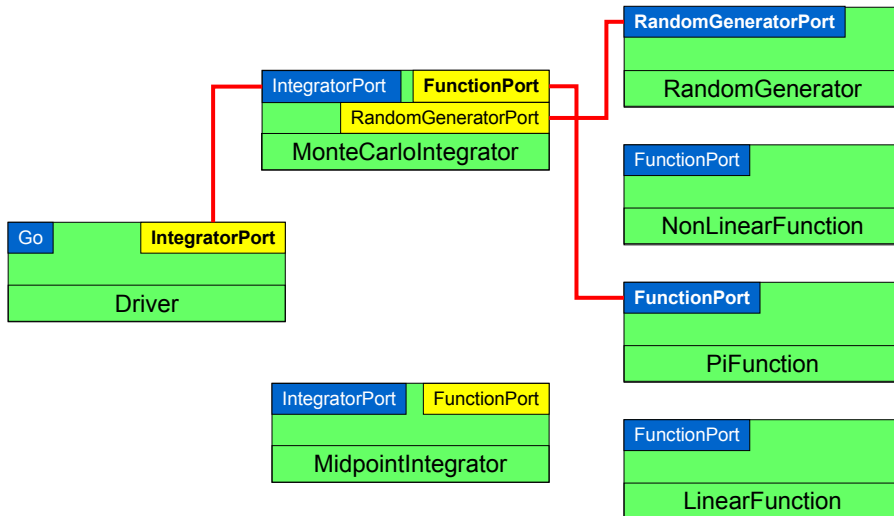
7

Pluggability: Scenario 1



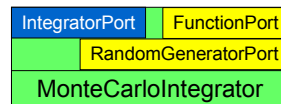
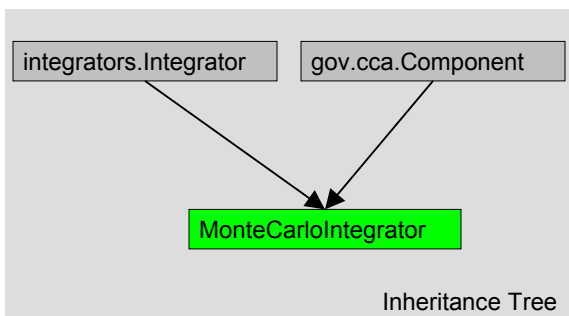
8

Pluggability: Scenario 2



9

MonteCarloIntegrator in Details



Relevant files:
integrator.sidl
function.sidl
random.sidl

What makes it a component?
Inheritance from **gov.cca.Component**

Where does **IntegratorPort** come from?
Inheritance from **integrators.Integrator**

10

Saying it in SIDL

```
version integrators 1.0;

package integrators {

    interface Integrator
        extends gov.cca.Port
    {
        double integrate(in double lowBound,
                        in double upBound, in int count);
    }
    class MonteCarloIntegrator
        implements-all Integrator,
                        gov.cca.Component
    {
        .....
    }
}
```

11

Notes

- Inheritance from ***gov.cca.Component*** furnishes the only method known to the framework: ***setServices()***
- “**Provides**” ports are interfaces that need to inherit from ***gov.cca.Port*** (***Integrator*** in this case)

12

The Framework Role

- Framework-to-Component: **setServices()**
 - Called after the component is constructed.
 - The component's chance to identify:
 - Ports it provides – **addProvidesPort()**
 - Ports it uses – **registerUsesPort()**
 - Component should not acquire the port here – Reason: it may not be there yet !!!!
 - Also used to “shutdown” the component.

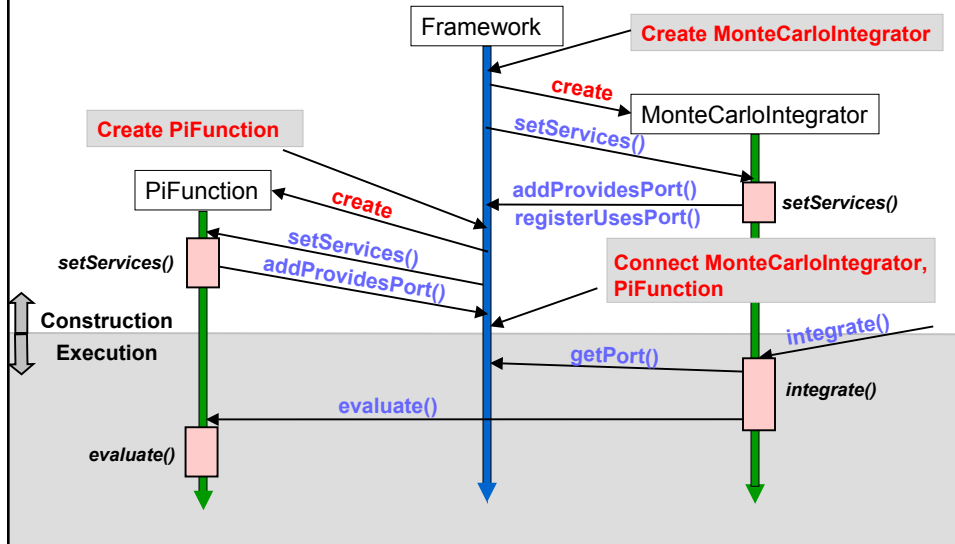
13

Component-to-Framework

- Mainly through **Services** object passed through **setServices()**.
- **addProvidesPort(), registerUsesPort()**:
 - Component “pointer”, PortName, PortType, PortProperties.
- **getPort()**:
 - Called by the using component.
 - Matching using portType (not name).
- **releasePort(), removeProvidesPort()**:
 - When all is done.

14

The Life Cycle Revisited



Example: `setServices()` in MonteCarloIntegrator (C++)

```

.....
frameworkServices = services;
if (frameworkServices._not_nil ()) {
    gov::cca::TypeMap tm = frameworkServices.createTypeMap ();
    gov::cca::Port p = self;
    frameworkServices.addProvidesPort (p,
        portName,
        "IntegratorPort",
        "integrators.Integrator", tm);
    // The Ports I use
    frameworkServices.registerUsesPort (
        portType,
        "FunctionPort",
        "functions.Function", tm);
    frameworkServices.registerUsesPort (
        "RandomGeneratorPort",
        "randomgen.RandomGenerator", tm);
}
.....
    
```

Annotations in the code:

- `portName` points to the string `"IntegratorPort"`.
- `portType` points to the string `"FunctionPort"`.
- `portProperties` points to the string `"randomgen.RandomGenerator"`.

Notes

- **setServices()** mainly used to inform the framework which ports the current component provides and/or uses.
- No actual connections between ports are established in **setServices()**, since the “other” port may not yet exist !!!
- **portName** is unique per component.
- **portType** identifies the “*interface*” that the port implements (used to match user and provider).
- **portProperties** : list of port-specific key-value pairs.

17

Example: *integrate()* in MonteCarloIntegrator (C++)

```
.....  
functions::Function functionPort;  
randomgen::RandomGenerator randomPort;  
double sum = 0.0;  
randomPort = frameworkServices.getPort ("RandomGeneratorPort");  
functionPort = frameworkServices.getPort ("FunctionPort");  
for (int i = 0; i < count; i++){  
    double x = lowBound + (upBound - lowBound) *  
               randomPort.getRandomNumber();  
    sum = sum + functionPort.evaluate(x);  
}  
frameworkServices.releasePort ("FunctionPort");  
frameworkServices.releasePort ("RandomGeneratorPort");  
return (upBound - lowBound) * sum / count;  
.....
```

18

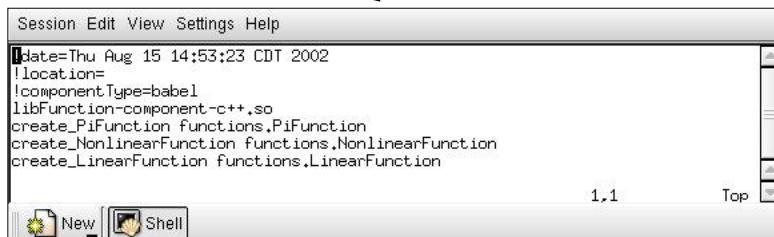
Putting it all together

- Getting the application to do something:
 - Assembling the components into an app.
 - Launching the Application.
- App assembly:
 - Framework need to be told what components to use, and where to find them.
 - Framework need to be told which **uses** port connects to which **provides** port.
- App execution: the **GO** port:
 - Special **provides** port used to launch the application (after connections are established).
 - Has one method, **go()**, that is called by the framework to get the application going.

19

Oh Component , where art thou?.

Which components, and how to create them



The screenshot shows a window titled "Session Edit View Settings Help". The main area contains a log of commands and their outputs:

```

date=Thu Aug 15 14:53:23 CDT 2002
!location=
!componentType=babel
libFunction-component-c++.so
create_PiFunction functions,PiFunction
create_NonLinearFunction functions,NonlinearFunction
create_LinearFunction functions,LinearFunction
  
```

At the bottom, there are buttons for "New" and "Shell". The status bar at the bottom right shows "1,1" and "Top".

More details in the Ccaffeine Module

20

App. Assembly The Ccaffeine way

Session Edit View Settings Help

```

repository get functions.PiFunction
repository get integrators.MonteCarloIntegrator
repository get integrators.MidpointIntegrator
repository get integrators.ParallelIntegrator
repository get tutorial.Driver

# Instantiate and name components that have been loaded
create randomgen.RandRandomGenerator rand
# F(x) = 4.0 / (1 + x^2)
create functions.PiFunction function
create integrators.MonteCarloIntegrator integrator
create tutorial.Driver driver

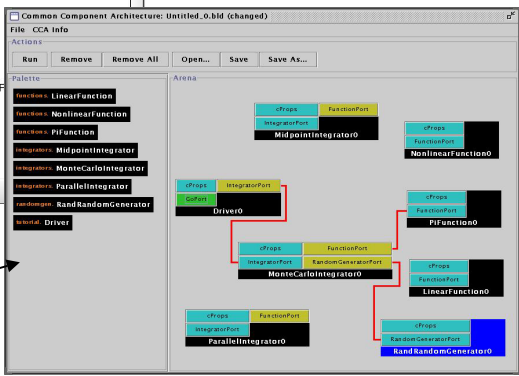
# Connect uses and provides ports
connect integrator FunctionPort function FunctionPort
connect driver RandomGeneratorPort rand RandomGeneratorPort
connect driver IntegratorPort integrator IntegratorPort

# Good to Go!
go driver GoPort
bye

```

New Shell

Command line "script"



GUI Interface