Introducing...

Gary Kumfert & Scott Kohn
Tammy Dahlgren, Tom Epperly,
Steve Smith, and Bill Bosl
What is Decaf?

A CCA framework.
100% v-0.5 compliant
Supports “Hello World!” components

Not a competitor with CCAT or CCAFFIENE

▷ Also thought of calling it “mouse”

Implemented using Babel
Proof of Babel’s applicability in CCA
research vehicle for us to grok CCA
distributed as example in 0.6.0 release
Outline

High Level Discussion

cca.sidl & decaf.sidl in detail

Implementing the framework

Implementing “Hello World”

Feedback into the CCA

Highlight Babel’s strengths and weaknesses
High Level Discussion

- cca.sidl from cca-forum.org
- decaf.sidl by hand
- Framework in C++
- “Hello World” component in F77
- Printf component in C
- Drivers in F77, Python & Java

Gary did in 2 days
Scott did in 3 days
## Lines of Code (Decaf only)

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cca.sidl (excl. comments)</td>
<td>28</td>
</tr>
<tr>
<td>decaf.sidl (excl. comments)</td>
<td>12</td>
</tr>
<tr>
<td>Generated C/C++ code (wc –l *)</td>
<td>22,067</td>
</tr>
<tr>
<td>Hand added Implementation</td>
<td>96</td>
</tr>
</tbody>
</table>
Biggest Time Consumers

Some Babel Bugfixes
- new tests added to regression suite
- Babel 0.6.0 or better required

Demystifying CCA Spec
- Lots of missing details
- How to create a component instance?
- How to bind a “uses port” to a “provides port”?
- No main()! Due to GUI heritage?
HelloDriver.java  (1/2)

```java
public class HelloDriver {
    public static void main(String args[]) {

        decaf.Framework decaf =
            new decaf.Framework();

        cca.ComponentID server =
            decaf.createInstance(
                "HelloServer.Component",
                "HelloServerInstance");

        cca.ComponentID client =
            decaf.createInstance(
                "HelloClient.Component",
                "HelloClientInstance");
    }
}
```
HelloDriver.java (2/2)

decaf.connect( client, "HelloServer", server, "HelloServer");

cca.Port port =
    decaf.lookupPort(client, "GoPort");

cca.ports.GoPort go = (cca.ports.GoPort) port._cast("cca.ports.GoPort");
go.go();

decaf.destroyInstance( server );
decaf.destroyInstance( client );
} // end main
} // end class
Outline

High Level Discussion

cca.sidl & decafé.sidl in detail

Implementing the framework

Implementing “Hello World”

Feedback into the CCA spec
Port – Key CCA Abstraction

// cca.sidl
version cca 0.5;
package cca {
    interface Port {
    }

...
PortInfo – Port MetaData

interface PortInfo {
    string getType();
    string getName();
    string getProperty( in string name );
}

class PortInfo implements-all cca.PortInfo {
    void initialize(
        in string name,
        in string type,
        in array<string> properties);
}
Discussion

Is array<string> appropriate for properties?

Can implement richer data structures.
Should Babel have a standard library?

class PortInfo implements all cca.PortInfo {
    void initialize(
        in string name,
        in string type,
        in array<string> properties);
}

decaf.sidl
Component – Base Class for all CCA components

interface Component {
    void setServices(
        in Services svcs
    );
}

CCA sidl

CASC
ComponentID – handle to a component

```java
interface ComponentID {
    string toString();
}

class ComponentID implements-all cca.ComponentID {

    void initialize( in string name );

}
```
Services – component’s view of the world (1/3)

interface Services {
    Port getPort( in string name );
    Port getPortNonblocking( in string name );
    void releasePort( in string name );
    PortInfo createPortInfo(
        in string name,
        in string type,
        in array<string> properties );
    ComponentID getComponentID();
    // ...
}
interface Services {
    // ... continued
    void registerUsesPort( in PortInfo pinfo );
    void unregisterUsesPort( in string name );
    void addProvidesPort( in Port inPort,
                           in PortInfo pinfo );
    void removeProvidesPort( in string name );
}
Services – component’s view of the world (3/3)

class Services implements-all cca.Services {

    void bindPort( in string name,
                    in cca.Port port );

    cca.Port getProvidesPort( in string name );

    void setComponentID(  
            in cca.ComponentID cid );

}
Discussion

Components can only access cca.Services, BuildServices needs decaf.Services

If downcasting from cca.Services to decaf.Services, then the components are framework specific.

Evidence of underspecification?

CCA spec needs to enumerate those special components that are not portable and must be implemented by each framework.
Indispensable Ports (1/2)

```java
package ports {

    interface GoPort extends Port {
        int go();
    }

    interface BuilderService extends Port {
        ComponentID createInstance(
            in string className,
            in string requestedInstanceName);
        void destroyInstance( in ComponentID toDie );
        void connect( in ComponentID user,
                     in string usingPortName,
                     in ComponentID provider,
                     in string providingPortName);
    }
}
```
class Framework implements-all cca.ports.BuilderService {
    cca.Port lookupPort(
        in cca.ComponentID id,
        in string portName );
}
Discussion

BuilderService is still under discussion and not formally adopted into the spec!

How did anyone build anything without create/connect/destroy functionality?
Outline

High Level Discussion

`cca.sidl` & `decafe.sidl` in detail

Implementing the framework

Implementing “Hello World”

Feedback into the CCA spec
Decaf Implementation Details

Used Babel’s C++ Bindings

generated 22K lines of code

Hand edited 8 files

added 96 lines of code by hand

decaf_{ PortInfo, ComponentID, Services } _Impl. _{ hh, cc }
namespace decaf {
    class PortInfo_Impl { 
    private:
        //...
        // DO-NOT-DELETE splicer.begin(decaf.PortInfo._data)
        std::string d_name;
        std::string d_type;
        std::map<std::string, std::string> d_propertyMap;
        // DO-NOT-DELETE splicer.end(decaf.PortInfo._data)
    } 
}

// DO-NOT-DELETE splicer.begin(decaf.PortInfo._includes)
#include <map>
// DO-NOT-DELETE splicer.end(decaf.PortInfo._includes)
string
decaf::PortInfo_impl::getProperty(
    /*in*/ string name )
throw ()
{
    // DO-NOT-DELETE splicer.begin(decaf.PortInfo.getProperty)
    return d_propertyMap[ name ];
    // DO-NOT-DELETE splicer.end(decaf.PortInfo.getProperty)
}
cca::ComponentID
decaf::Framework_impl::createInstance(
  /*in*/ string className,
  /*in*/ string requestedInstanceName )
throw()
{
    // DO-NOT-DELETE splicer.begin(decaf.Framework.createInstance)
    SIDL::BaseClass sidl_class =
        SIDL::Loader::createClass( className );
    cca::Component component = sidl_class;
decaf::ComponentID cid;
    if ( component._not_nil() ) {
        string uniqueName = getUniqueName( requestedInstanceName );
        cid = decaf::ComponentID::_create();
        cid.initialize( uniqueName );
        decaf::Services svc = decaf::Services::_create();
        svc.setComponentID( cid );
        d_instance[ uniqueName ] = std::make_pair( component, svc );
        component.setServices( svc );
    }
    return cid;
    // DO-NOT-DELETE splicer.end(decaf.Framework.createInstance)
}
Outline

High Level Discussion

```
cca.sidl & decaf.sidl in detail
```

Implementing the framework

Implementing “Hello World”

Feedback into the CCA spec
HelloWorld SIDL Files

```idl
version HelloServer 0.5;
package HelloServer {
    interface HelloPort extends cca.Port {
        string sayHello();
    }
    class Component implements-all HelloPort, cca.Component {}
}

version HelloClient 0.5;
package HelloClient {
    class Component implements-all cca.ports.GoPort, cca.Component {}
}
```
subroutine HelloServer_Component_setServices_impl(
    & self, services)
C ...
call SIDL_string__array_create_f(1, 0, 0, properties)
call cca_Services_createPortInfo_f(
    & services,
    & 'HelloServer',
    & 'HelloServer.HelloPort',
    & properties,
    & portinfo)
call HelloServer_Component__cast_f(self, 'cca.Port',
    & port)
call cca_Services_addProvidesPort_f(services, port,
    & portinfo)
call cca_PortInfo_deleteReference_f(portinfo)
call SIDL_string__array_destroy_f(properties)
import decaf.Framework
import cca.ports.GoPort

if __name__ == '__main__':
    decaf = decaf.Framework.Framework()

    server = decaf.createInstance(
        "HelloServer.Component",
        "HelloServerInstance");

    client = decaf.createInstance(
        "HelloClient.Component",
        "HelloClientInstance");
```python
decaf.connect(client,"HelloServer", server,"HelloServer")

port = decaf.lookupPort(client, "GoPort")
go = cca.ports.GoPort.GoPort(port)

g.go()

decaf.destroyInstance(server)
decaf.destroyInstance(client)
```
import decaf.Framework
import cca.ports.GoPort

if __name__ == '__main__':
    decaf = decaf.Framework.Framework()
    server = decaf.createInstance(
        "HelloServer.Component",
        "HelloServerInstance");
    client = decaf.createInstance(
        "HelloClient.Component",
        "HelloClientInstance");

What's non-standard?
HelloDriver.py  (2/2)

What's non-standard?

defaf.connect(client,"HelloServer", server,"HelloServer")

port = decaf.lookupPort(client, "GoPort")
go = cca.ports.GoPort.GoPort(port)

go.go()

defaf.destroyInstance(server)
defaf.destroyInstance(client)
Outline

High Level Discussion

cca.sidl & decaf.sidl in detail

Implementing the framework

Implementing “Hello World”

Feedback into the CCA spec
Feedback about CCA Spec

minimal
to the point of being unusable
cannot implement "Hello World! as is
doesn’t standardize driving from main()
More people probably care about scripted assembly
than GUIs... and it scales better!

no create/bind/go/destroy in core spec
Recommend a cca.Framework interface
Recommend looking at Babel’s SIDL::Loader
Babel is Ready for CCA

- Demonstrated language independent CCA components.
- Not necessarily Bug Free.
- Babel’s firewall between interface and implementation forces better software design

Is CCA ready for Babel?
The End
Work performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48