CCA
Common Component Architecture

Language Interoperability Using

CCA Forum Tutorial Working Group
http://www.cca-forum.org/tutorials/

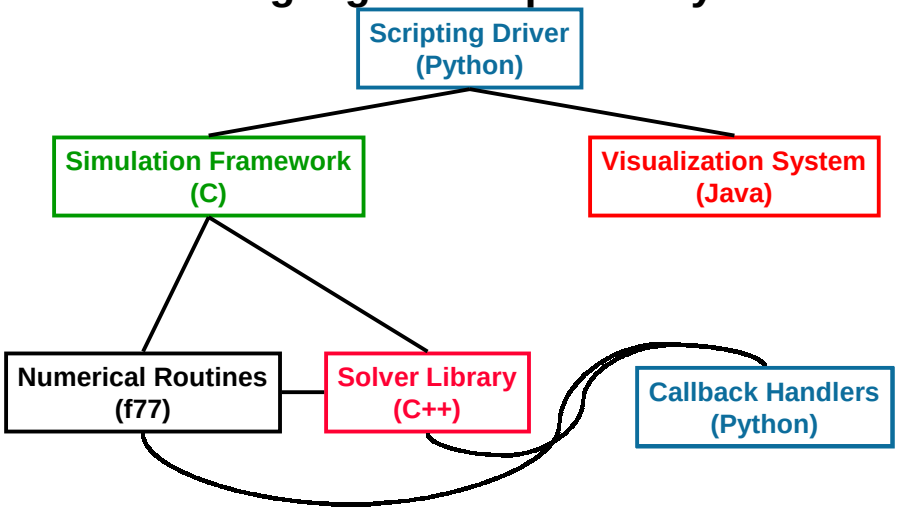
Contributors:

Gary KumfertTammy Dahlgren, Tom Epperly, & Scott Kohn



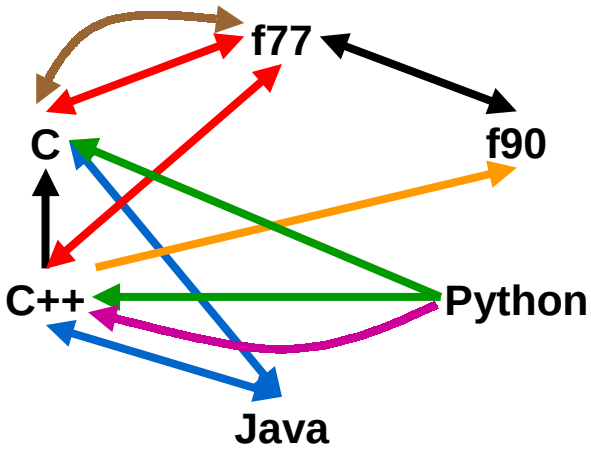
CCA
Common Component Architecture

What I mean by "Language Interoperability"



CCA Common Component Architecture

One reason why mixing languages is hard



Native

cfortran.h

SWIG

JNI

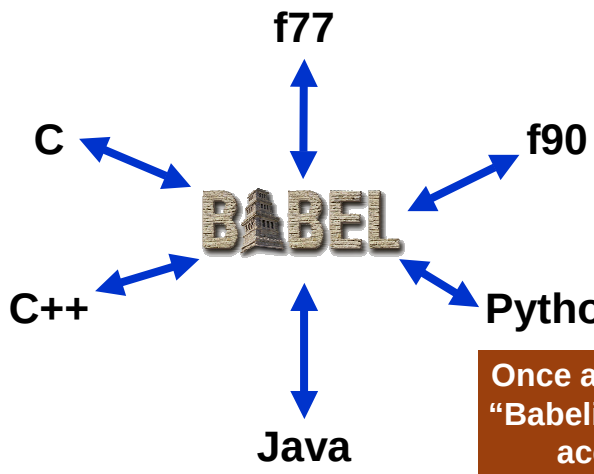
Siloon

Chasm

Platform Dependent

CCA Common Component Architecture

Babel makes all supported languages peers



This is not an LCD Solution!

Once a library has been "Babelized" it is equally accessable from all supported languages

CCA Common Component Architecture

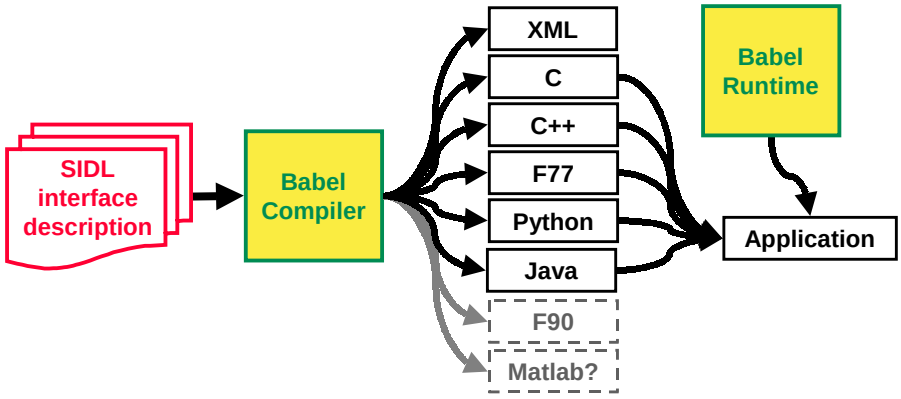
Babel Module's Outline

- Introduction
- Babel Basics
  - What Babel does and how
  - How to use Babel
  - Concepts needed for future modules
- Babel & CCA
  - History & Current directions
  - Decaf Framework
  - Building language independent CCA components
  - Demo

CCA Common Component Architecture

Babel's Mechanism for Mixing Languages

- Code Generator
- Runtime Library



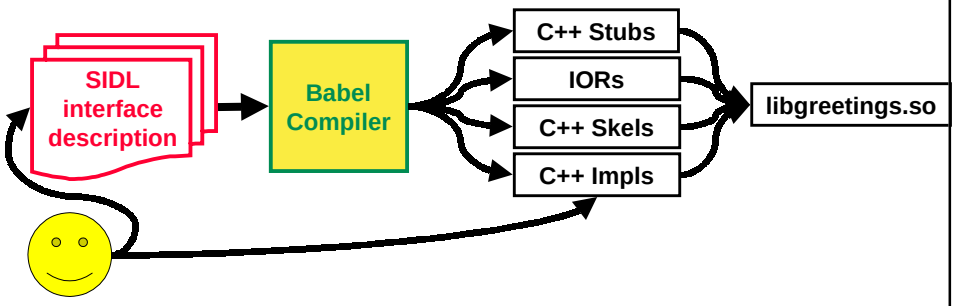
CCA Common Component Architecture

greetings.sidl: A Sample SIDL File

```
version greetings 1.0;
package greetings {
    interface Hello {
        void setName( in string name );
        string sayIt ( );
    }
    class English implements-all Hello {  }
}
```


CCA Common Component Architecture

Library Developer Does This...



- `babel --server=C++ greetings.sidl`
- Add implementation details
- Compile & Link into Library/DLL

CCA Common Component Architecture

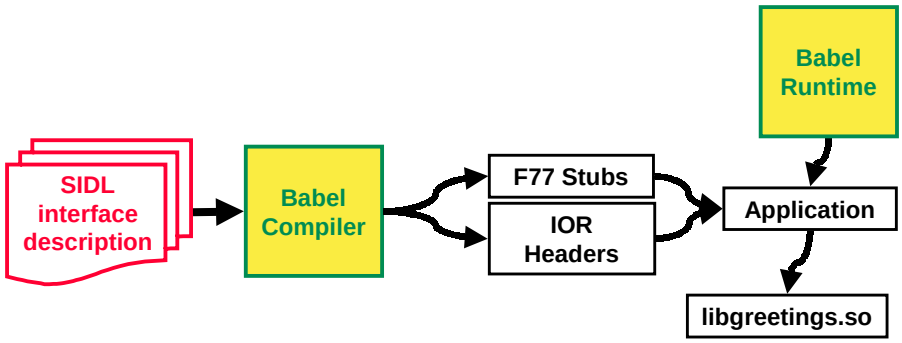
Adding the Implementation

```
namespace greetings {
class English_impl {
  private:
    // DO-NOT-DELETE splicer.begin(greetings.English._impl)
    string d_name;
    // DO-NOT-DELETE splicer.end(greetings.English._impl)
```

```
string
greetings::English_impl::sayIt()
throw ()
{
  // DO-NOT-DELETE splicer.begin(greetings.English.sayIt)
  string msg("Hello ");
  return msg + d_name + "!";
  // DO-NOT-DELETE splicer.end(greetings.English.sayIt)
}
```


CCA Common Component Architecture

Library User Does This...



- `babel --client=F77 greetings.sidl`
- Compile & Link generated Code & Runtime
- Place DLL in suitable location

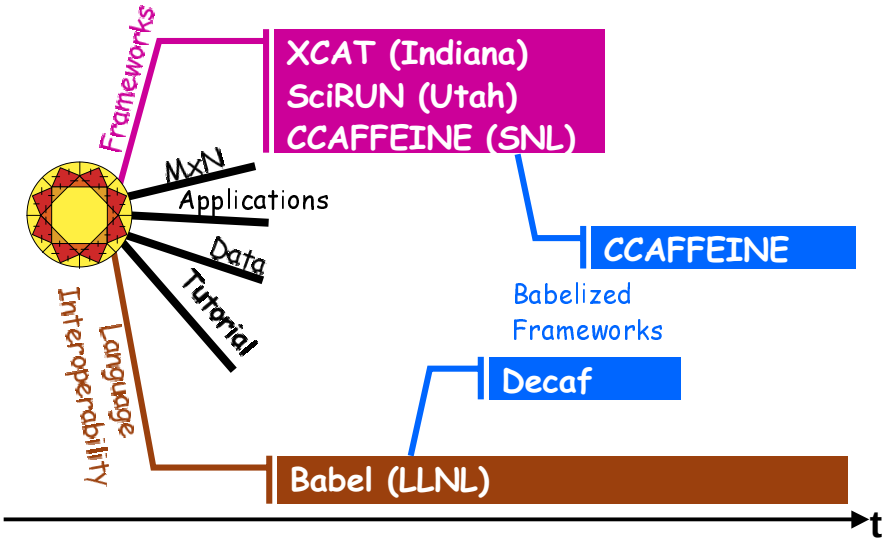
CCA Common Component Architecture

Babel Module's Outline

- Introduction
- Babel Basics
  - What Babel does and how
  - How to use Babel
  - Concepts needed for future modules
- Babel & CCA
  - History & Current directions
  - Decaf Framework
  - Building language independent CCA components
  - Demo

CCA Common Component Architecture

History of Babel & CCA



Decaf Details & Disclaimers

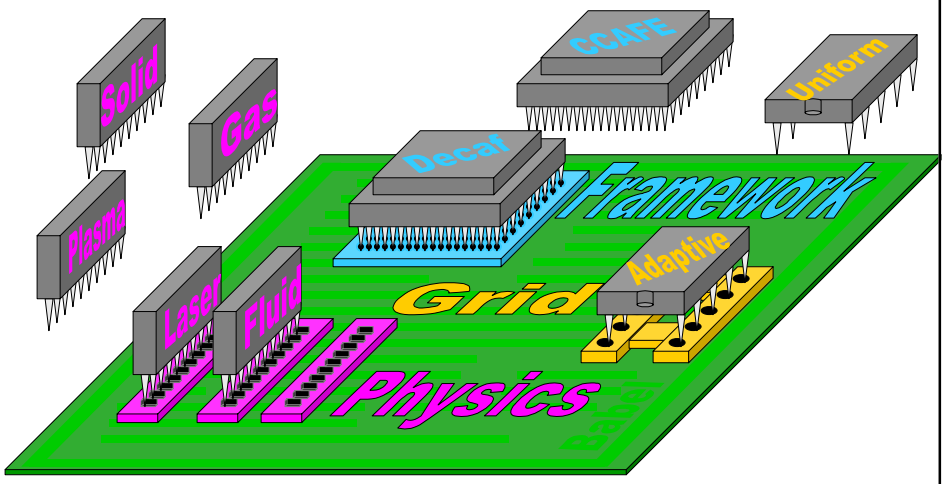
- Babel is a hardened tool
- Decaf is an example, not a product
  - Demonstrate Babel's readiness for "real" CCA frameworks
  - Maintained as a stopgap
  - Distributed in "examples" subdirectory of Babel
- Decaf has no GUI

The CCA Spec is a SIDL File

```
version cca 0.6;
package cca {
    interface Port { }
    interface Component {
        void setServices( in Services svcs );
    }
    interface Services {
        Port getPort( in string portName );
        registerUsesPort( /*etc*/ );
        addProvidesPort( /*etc*/ );
    /*etc*/
```

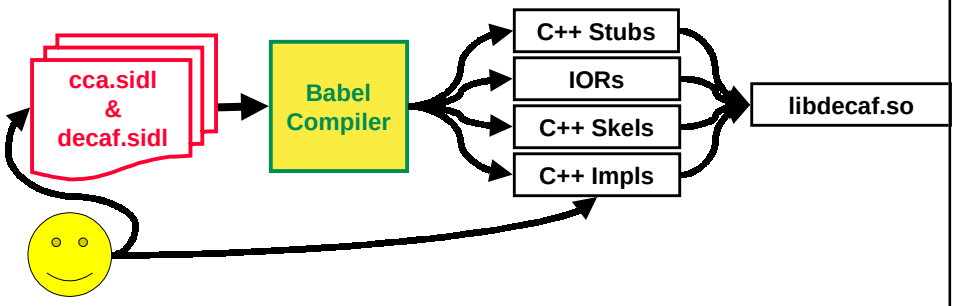
CCA
Common Component Architecture

The CCA from Babel's POV



CCA
Common Component Architecture

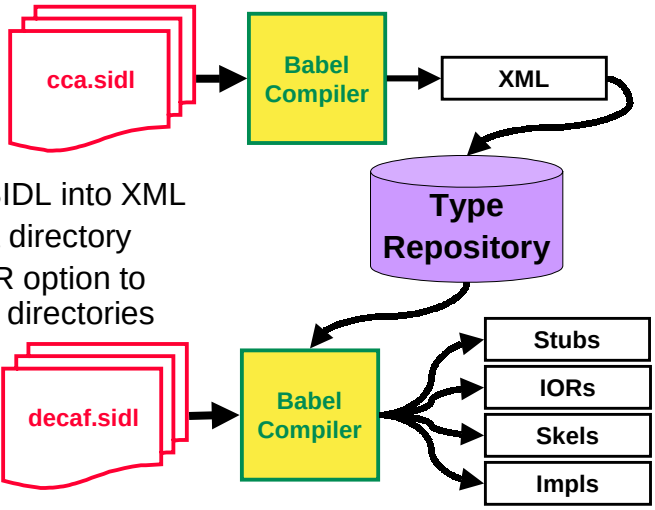
How I Implemented Decaf



- wrote decaf.sidl file
- `babel --server=C++ cca.sidl decaf.sidl`
- Add implementation details
- Compile & Link into Library/DLL

CCA Common Component Architecture

An Extra Babel Tip



- "precompile" SIDL into XML
- store XML in a directory
- Use Babel's –R option to specify search directories

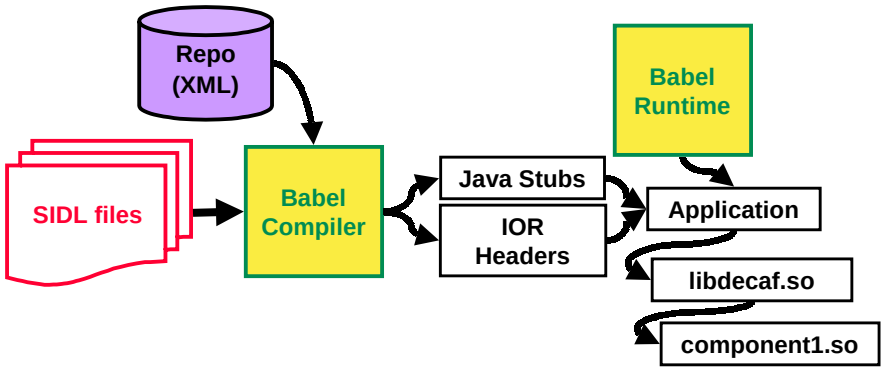
CCA Common Component Architecture

How to Use CCA Components and Decaf

- Decaf doesn't provide a GUI
- Simply program by explicitly
  - creating components
  - connecting points
  - envoking the "goPort"
- Use Babel as needed to generate bindings in your language of choice
- Make sure Babel Runtime can locate DLLs for Decaf and any CCA components.

CCA Common Component Architecture

To Use the Decaf Framework



- `babel --client=Java –Rrepo function.sidl`
- Compile & Link generated Code & Runtime
- Place DLLs in suitable location

CCA Common Component Architecture

Example: A Driver in Python

```
import decaf.Framework
import cca.ports.GoPort
if __name__ == '__main__':
  decaf = decaf.Framework.Framework()

  server = decaf.createInstance( "ServerName",
           "HelloServer.Component", 0 )
  client = decaf.createInstance( "ClientName",
           "HelloClient.Component", 0 )

  decaf.connect(server,"HelloPort",
                client,"HelloPort" )

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

CCA
Common Component Architecture

How to Write and Use Babelized CCA Components

- Define “Ports” in SIDL
- Define “Components” that implement those Ports, again in SIDL
- Use Babel to generate the glue-code
- Write the guts of your component(s)

CCA
Common Component Architecture

SIDL 101: Classes & Interfaces

- SIDL has 3 user-defined objects
  - Interfaces – APIs only, No Implementation
  - Abstract Classes – 1+ methods unimplemented
  - Concrete Classes – All methods are implemented
- Inheritance (like Java/Objective C)
  - Interfaces may extend Interfaces
  - Classes extend no more than one Class
  - Classes can implement multiple Interfaces
- Only Concrete Classes can be Instantiated

CCA Common Component Architecture

SIDL 101: Methods and Arguments

- Methods are public virtual by default
  - static methods are not associated with an object instance
  - final methods can not be overridden
- Arguments have 3 parts
  - Mode: can be in, out, or inout (like CORBA)
  - Type: one of (bool, char, int, long, float, double, fcomplex, dcomplex, array<Type,Dimension>, enum, interface, class)
  - Name:

CCA Common Component Architecture

How to Write A Babelized CCA Component (1/3)

- Define “Ports” in SIDL
  - CCA Port =
    - a SIDL Interface
    - extends cca.Port

```
version tutorial 1.0;

package tutorial {
    interface Function extends cca.Port {
        double evaluate( in double x );
    }
}
```

CCA Common Component Architecture

How to Write A Babelized CCA Component (2/3)

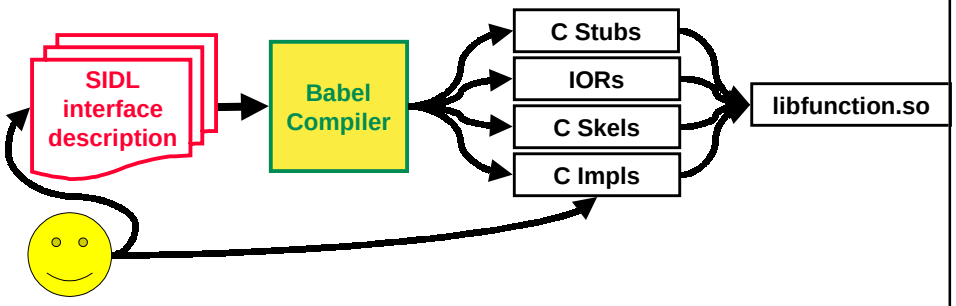
- Define “Components” that implement those Ports
  - CCA Component =
    - SIDL Class
    - implements cca.Component (& any provided ports)

```
class LinearFunction implements tutorial.Function,
                                cca.Component {
      double evaluate( in double x );
      void setServices( in cca.Services svcs );
}
```

```
class LinearFunction implements-all
       tutorial.Function, cca.Component { }
```


CCA Common Component Architecture

How to Write A Babelized CCA Component (3/3)



- Use Babel to generate the glue code
  - `babel --server=C –Rrepo function.sidl`
- Add implementation Details