BABEL-ized Libraries Make Large Scale Software Easier to Fabricate, Maintain, and Evolve

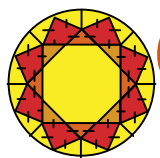
Gary Kumfert,

Tamara Dahlgren, and Thomas Epperly

Lawrence Livermore National Laboratory

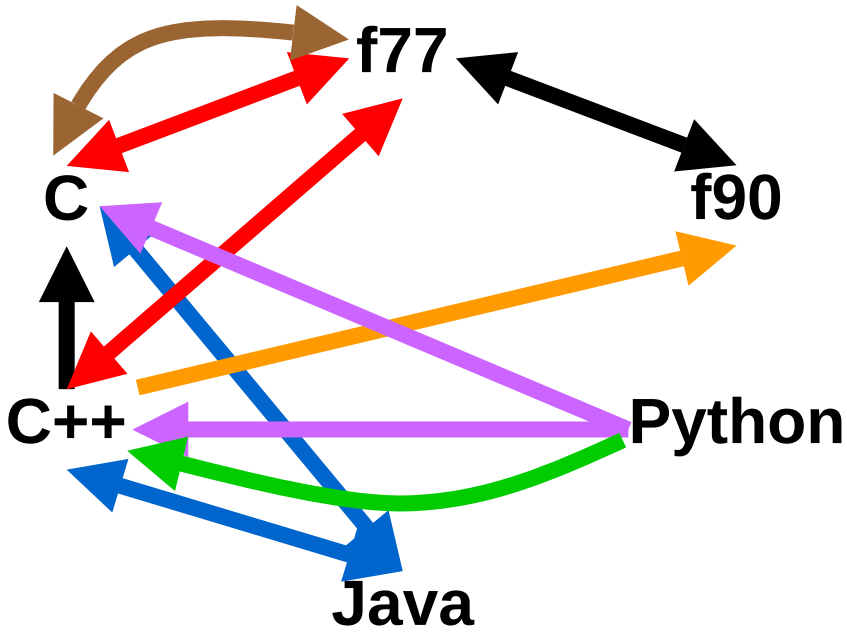
This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

UCRL-PRES-202548

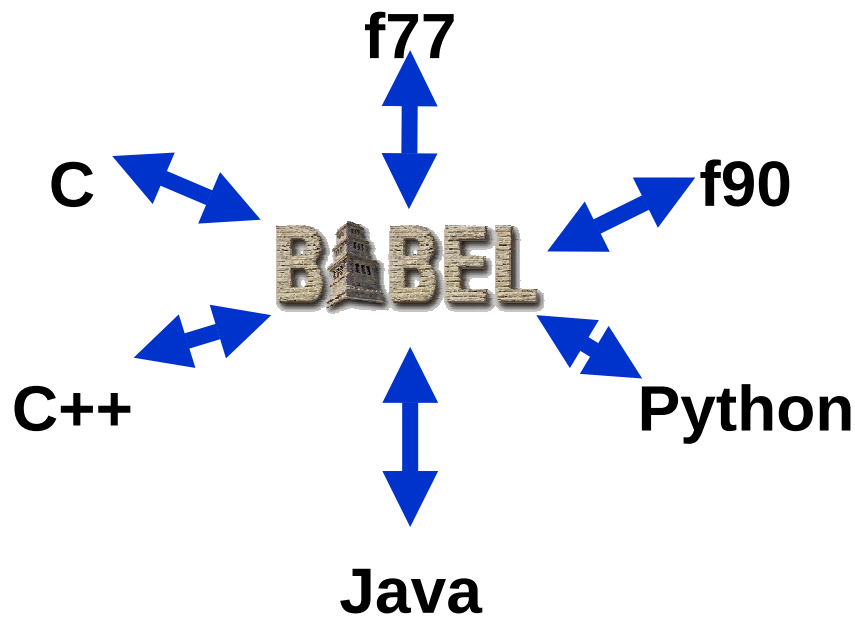


CCA
Common Component Architecture

Babel: Tool of choice for mixing more than 2 languages

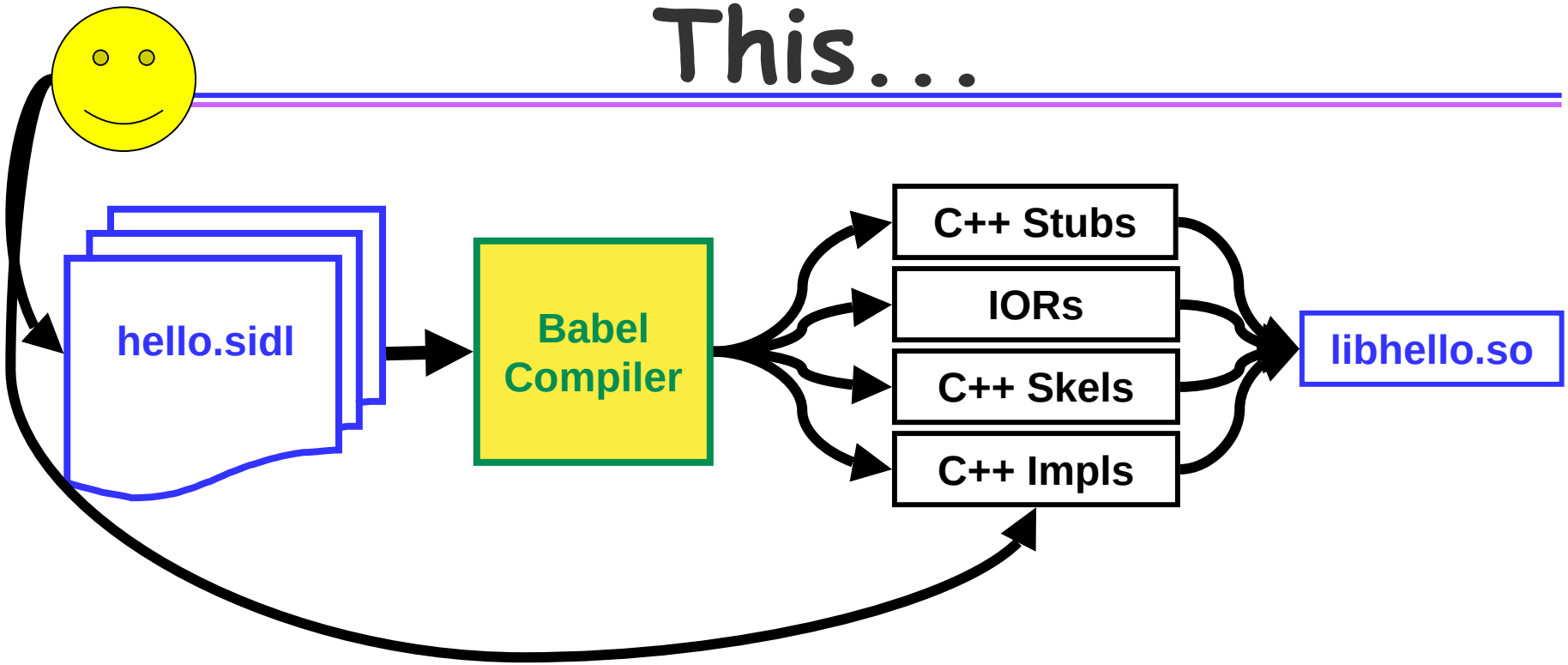


Native	cfortran.h
SWIG	JNI
Siloon	Chasm
Platform	Dependent



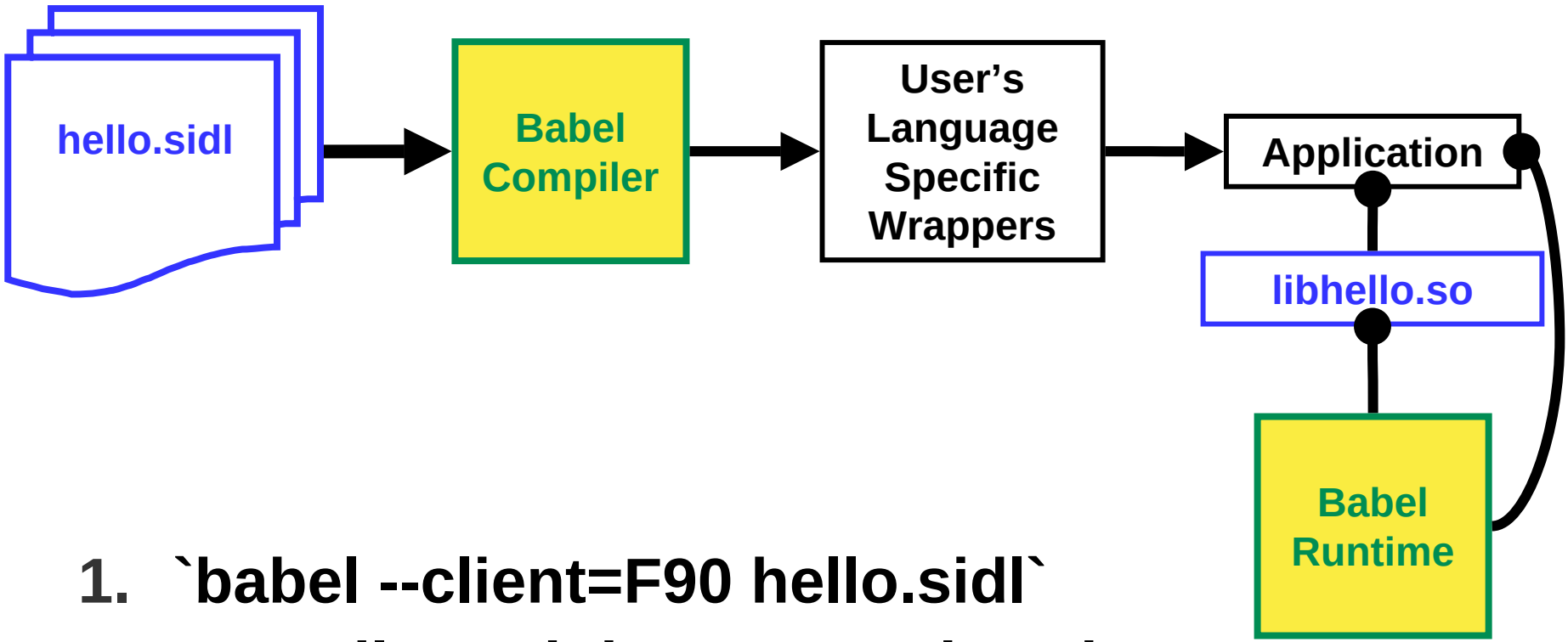
Not a LCD Solution.
Supports polymorphism, reference counting, and exceptions in all languages.

Library Developer Does This...



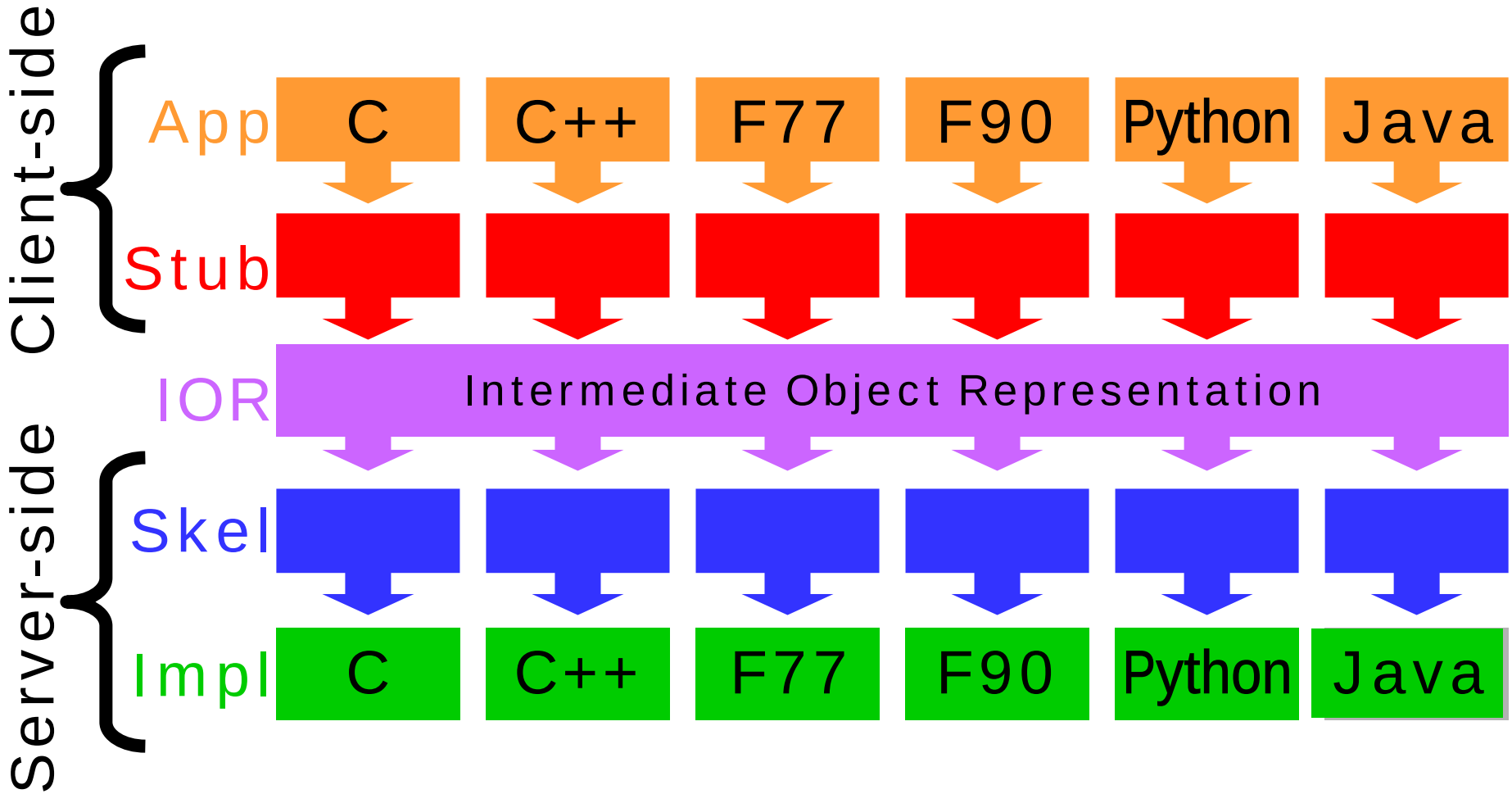
1. Write SIDL File
2. `babel --server=C++ hello.sidl`
3. Add implementation details
4. Compile & Link into Library/DLL

Library User Does This...



1. `babel --client=F90 hello.sidl`
2. Compile & Link generated Code & Runtime
3. Place DLL in suitable location

Another view of Babel's 2-Stage Wrapping Architecture



New Story

Babel is a Language Interoperability Tool, but...

... observed recurring themes (benefits) when Babel was applied in scientific software, not directly related to mixing languages

Babel and Fabrication of Large Scale Scientific SW

- Automates the tedious and mechanical process of connecting languages
- SIDL as a design & negotiation tool
- Reduce dependency entanglement among developers
  - Software expert does SIDL design
  - Everyone else codes whatever they like (in whatever language they like) to add implementation to stubbed out codes
  - Maps well to multidisciplinary teams

Fabrication Examples

- SIDL supports multi-lingual standards
  - CCA, TSTT, TOPS
- People like arguing designs in SIDL
  - Easy to pick up
  - No distracting implementation details
  - Easy to e-mail, no special editors (like UML)
- Hypre is mostly numerical researchers, not OOP software engineers.
  - Lead controls the interfaces
  - Researchers implement the guts

Babel and Maintenance of Large Scale Scientific SW

- Preserve functionality & correctness in the face of external change
- Generated language wrappers are more robust and portable than ad-hoc solutions.
- Babel adds new languages ~ 1/year
- Good software practices encoded in Babel avoids lots of bugs
  - reference counting
  - exception handling
  - semantic checking (in development)

Maintenance Example

- **NWChem uses Babel to connect Fortran 77 to Fortran 77**
  - **2 Fortran 77 codes, some C in each**
  - **Fortran 77 has no binary standard**
    - **some add an underscore to linker symbols**
    - **some don't**
    - **g77 adds two if one already exists, just one otherwise**
  - **The two codes weren't designed for each other, and each made a different assumption.**
  - **They found it easier to wrap both in Babel and let Babel hide the underscore issue inside each library**

Babel and the Evolution of Large-Scale Scientific SW

- Internal change for new capabilities
- Fast Prototyping with Scripting
  - Empower and exploit creativity of users
- High Performance with compiled code
- Polymorphism – create new functionality without
  - Changing existing code
  - Making lots of copies

Evolution Example

- **CCA Forum**
  - **started with 3 “reference frameworks”**
    - **none were interoperable**
  - **Babel later adopted for frameworks to support components multiple languages**
    - **Side effect: Also made components work with frameworks in multiple languages.**
  - **Next generation of CCA frameworks (Ccaffeine, XCAT3 and SCIRun2) will all seamlessly load each other’s Babelized components.**
    - **Will also bridge between older component models and Babel**
    - **Frameworks can also expose themselves as a component to another framework**

Scientific Computing Software is Dominated by Change

- **Changed more often than software of similar size in other fields**
  - **A twenty-year-old LLNL program changed substantively 75 times in one year. It was not a period of major new development or a new machine.** -- Paul F. DuBois, LLNL
- **Both developers and users make changes**
- **Software changes track scientific progress (hard to predict)**
- **Application area may change or expand**
- **Hardware will change, especially high performance, large-scale exotics**

Babelized Libraries appear very "Change Oriented."

- **Project:** http://www.llnl.gov/CASC/components
- **Project Team Email:** components@llnl.gov
- **Mailing Lists:** majordomo@lists.llnl.gov

  subscribe babel-users *[email address]*

  subscribe babel-announce *[email address]*