Combat

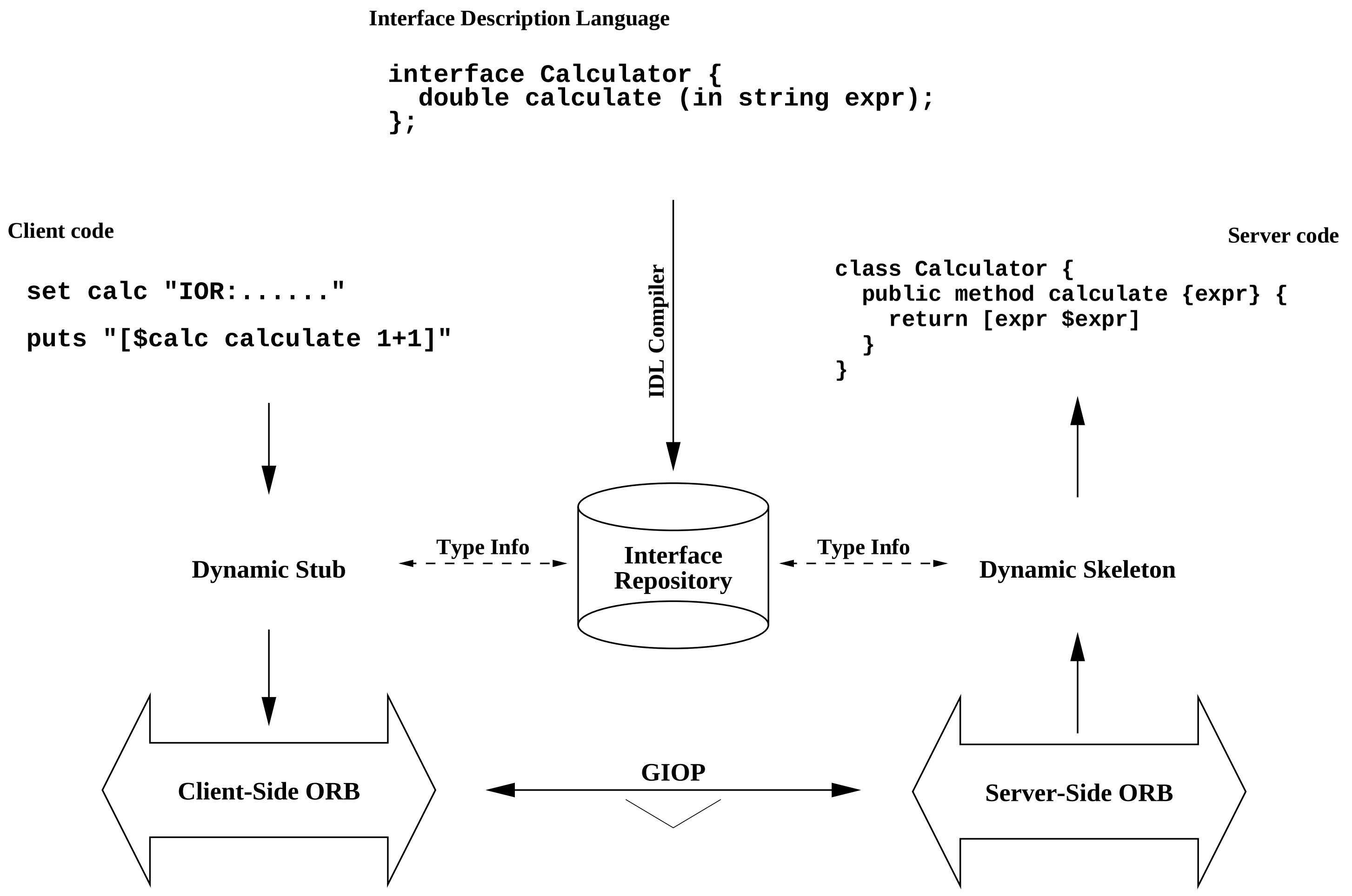
A CORBA Language Mapping for Tcl

Frank Pilhofer
`fp@fpx.de`

CORBA

- De-facto standard for distributed systems
- Ideal for heterogeneous systems, independent of Hardware, Operating System, Programming Language and Vendor
- Open standard, documents freely available
- Based on Client/Server model
- Object-oriented

Processing a CORBA Request



```
interface Calculator {
  double calculate (in string expr);
};
```

```
set calc "IOR:......"

puts "[$calc calculate 1+1]"
```

```
class Calculator {
  public method calculate {expr} {
    return [expr $expr]
  }
}
```

CORBA Language Mapping

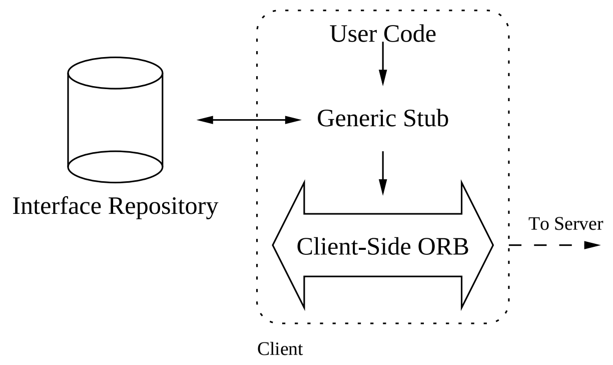
- Maps IDL data types to target language types
  - Simple data types (`boolean`, `short`, `long`, `double`, `string`, `...`)
  - Complex data types (`struct`, `sequence`, `any`, `...`)
- Defines representation of client-side stubs
- Representation of server-side skeletons
- Official language mappings exist for C, C++, Java, Smalltalk, Ada, COBOL, Python and IDLscript

Tcl Language Mapping - IDL Types

- Most simple IDL types can be represented with native Tcl types
- Other types (`unsigned long` (32 bit), `long long` (64 bit), `fixed` (fixed-point decimal)) are mapped to Tcl strings
- Complex types are mapped to Tcl lists, e.g. `{foo bar}` matches `sequence<string>`
- Thanks to Tcl's lax type system, IDL data types blend in naturally and intuitively

Tcl Language Mapping - Client-Side Stubs

- Stubs are mapped to Tcl procedures, “handles”, that encapsulate object address
- First argument is interpreted as operation name, remaining aguments as parameters



- Type information is retrieved from the Interface Repository at runtime $\Rightarrow$ client side is fully dynamic

- Open question: garbage collection of handles? (Core patch!)

Tcl Language Mapping - Server-Side Skeletons

- *Servants* (instances of a Skeleton) have state, code and identity $\Rightarrow$ servants are objects!
- Consequently, skeletons are mapped to [INCR TCL] classes
- The user extends the generic skeleton base class and implements an interface’s methods
- Type information is retrieved from the Interface Repository at runtime $\Rightarrow$ server side is fully dynamic
- Open questions: multiple inheritance, garbage collection?

Language Mapping Summary

- IDL types must be mapped to native Tcl types and have an intuitive string representation
- Stubs and skeletons should be fully dynamic, without requiring compile-time information
- Client side needs asynchrony features to keep the event loop running (especially for seamless Tk integration)
- Is garbage collection of handles/objects compulsory, or should the mapping only rely on current Tcl/[incr Tcl] features?

Current State

- Combat/C++
  - Tcl extension that hooks up to any CORBA 2.3 ORB
  - Full Client- and Server-side solution
- Combat/Tcl
  - Proof-of-concept version in pure Tcl
  - Full Client-side implementation
  - Ideal for heterogeneous environments, embedded devices or sandboxes