

Customizing IDL Mappings and ORB Protocols

Girish Welling & Maximilian Ott

C&C Research Laboratories, NEC USA, Inc.
www.ccrl.nj.nec.com



Where are we coming from?

- Distributed Multimedia Toolkit (started 1992)
- Organically grown
 - from OO C with own scripting language to C++/Java
 - string based RPC (debug with telnet)
 - NO threads! (First we didn't have any, now we don't want any)
 - SunOS/X, Solaris/XIL, Windows/DirectX
 - Ethernet, ATM (with own driver extensions), Wireless ATM (with own hardware)



Heidi - Collaborative Multimedia



NEC

Other projects

- Embedded systems
 - pSOS on i960
 - ⇒ footprint!
- Mobile computing
 - adapting to changes in resource availability
 - intermittent connectivity
 - ⇒ policy versus functionality!

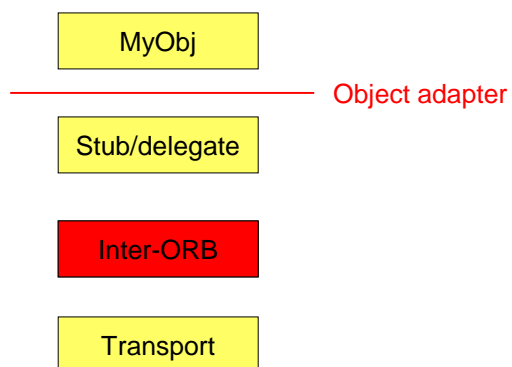
NEC

Standards are fine, but ...

- Legacy code
 - lots of “dead adapter” code required to tie to a specific framework (if you are lucky)
- Customizing the ORB
 - code size is an issue for embedded systems
- Customizing the protocol
 - string based protocols are great for debugging
- Custom optimizations
 - dispatching through string comparison ⇒ slow

NEC

What bothered us?



NEC

Heidi: What we had

- Large C++ application
 - “real” interfaces
- Inter-process object communicator
- Hand-crafted stubs & skeletons

NEC

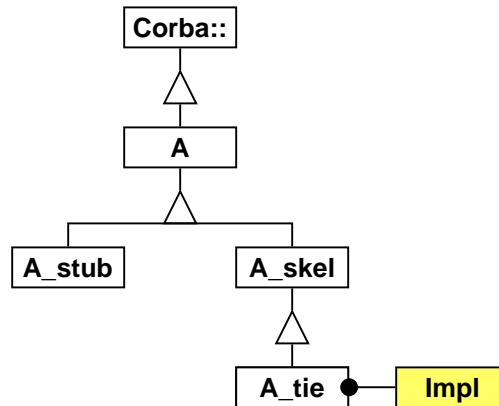
Heidi: What we wanted

- Automate as much as possible
 - stubs? skeletons?
- Reuse what we already had
 - C++ interface definitions
 - C++ object implementations

Would CORBA help?

NEC

Corba: Stubs & Skeletons



NEC

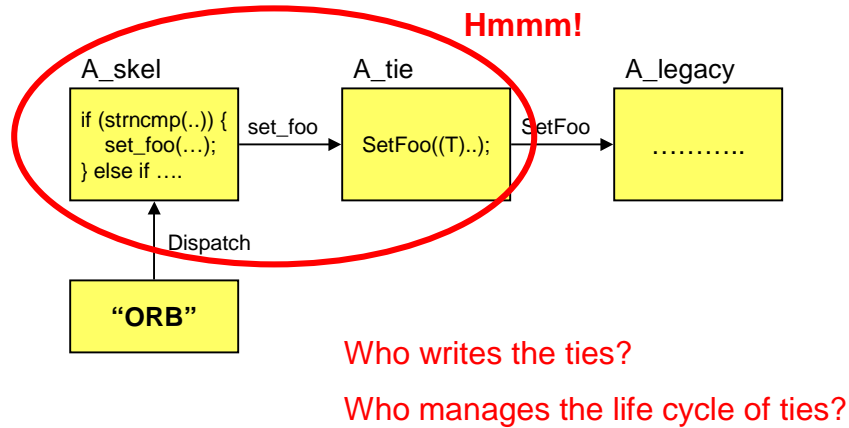
Corba & legacy code ...

IDL type	Corba C++ type	Legacy C++ type
long	CORBA::Long	long
boolean	CORBA::Boolean	XBool
float	CORBA::Float	float

IDL	Corba	Legacy
attribute long foo;	set_foo(...);	SetFoo(...);

NEC

Corba prescribed approach



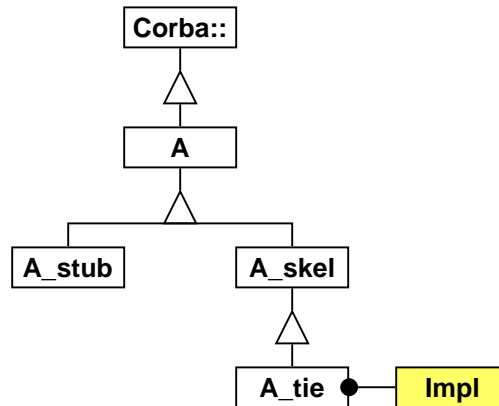
NEC

What do we really want?

- Delegation model
- Customizable **name** mapping
- Customizable **type** mapping
- Automatic code generation
- No unnecessary indirections (**ties**)
- **C++ as the IDL**

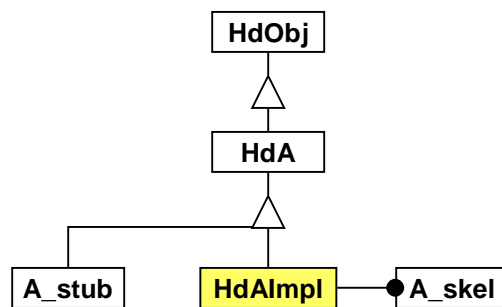
NEC

Corba: Stubs & Skeletons



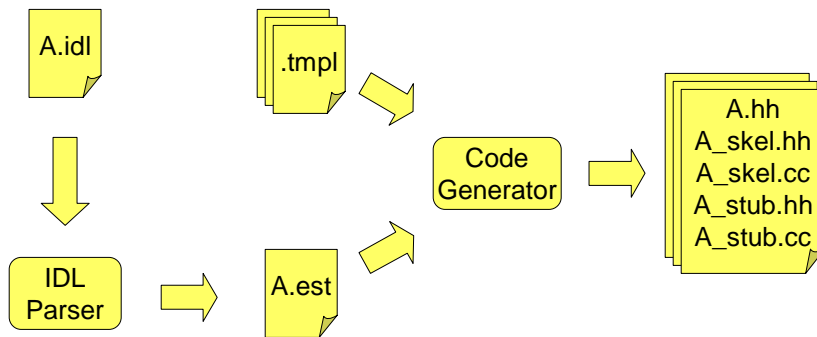
NEC

Heidi: Stubs & Skeletons



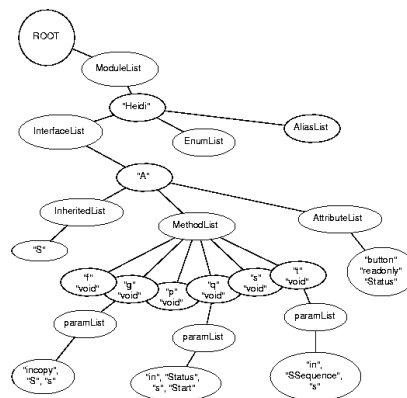
NEC

Template-driven IDL Compiler



NEC

Extended Syntax Tree



NEC

EST encoded in a Perl program

```
#!/usr/bin/perl
use Ast;
use JeevesUtil;
$ROOT = $n0 = Ast::New("Root");
#
# IDL:Heidi:1.0
#
$n1 = Ast::New("Heidi", "Module", $n0);
#
# IDL:Heidi/Status:1.0
#
$n2 = Ast::New("Status", "Enum", $n1);
@m = [ Start, Stop ];
$n2->AddProp("members", @m);
#
# IDL:Heidi/SSequence:1.0
#
$n2 = Ast::New("SSequence", "Alias", $n1);
$n2->AddProp("type", "sequence");
```

NEC

Enhanced Templates

- Text + substitution
- Directives for traversing EST
- Escapes
 - complex name mapping
 - complex type mapping (think strings & C++)

NEC

Templates ...

```
@foreach interfaceList \  
    -map interfaceName CPP::MapClassName  
@openfile ${interfaceName}.hh  
  
/* File ${interfaceName}.hh */  
class ${interfaceName} :  
  
    @foreach inheritedList -ifMore ',\' \  
        -map inheritedName CPP::MapClassName  
        virtual public ${inheritedName} ${ifMore}  
    @end inheritedList  
{  
    ...  
}
```

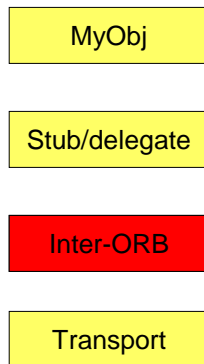
NEC

Example

<pre>/* file A.idl */ module Heidi { interface S; enum Status {Start, Stop}; typedef sequence <S> SSeq; interface A : S { void f(in A); void g(in copy S s); void p(in long l = 0); attribute Status button; }; };</pre>	<pre>/* file A.hh */ enum HdStatus {Start, Stop}; typedef HdList<HdS> HdSSeq; typedef HdListIterator ... class HdA : virtual public HdS { public: virtual void f(HdA*) =0; virtual void g(HdS*) =0; virtual void p(long l = 0) =0; virtual HdStatus GetButton() =0; virtual void SetButton(... ... virtual ~HdA() {} }; HD_GUID("A", 0xF129AC0196357CD10);</pre>
--	--

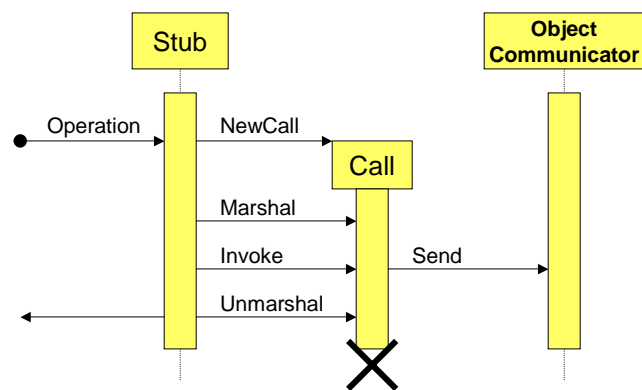
NEC

Customizing the ORB



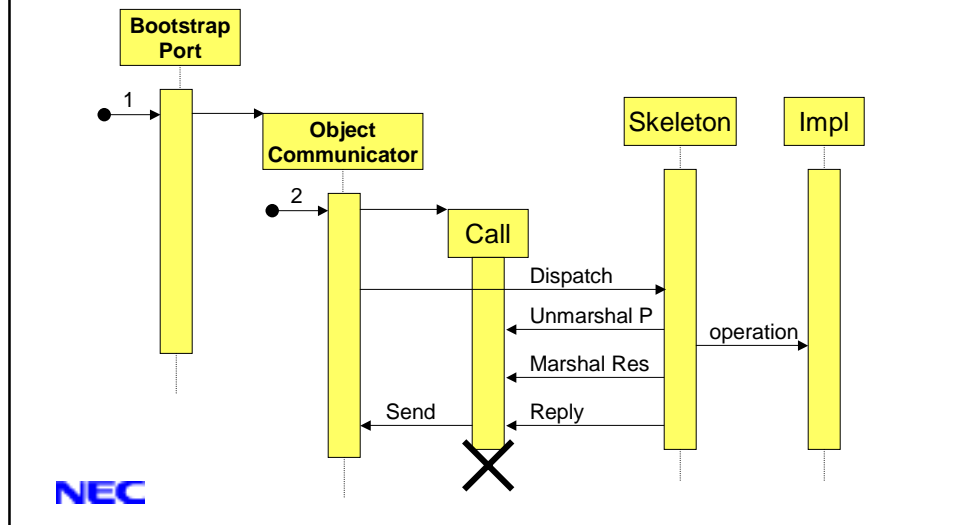
NEC

HEIDIRMI - Client Side



NEC

HEIDIRMI - Server Side



TclORB

- Needed a few simple GUI clients to a CORBA base network management system
- ~700 hundred lines of Tcl code
 - subset of GIOP/IIOP
 - pure tcl (no extension library)
- 2 weeks of coding

Related Work

- Customizable compilers
 - Flick
 - three stage compiler
 - recently added template language (SCML)
 - ORBacus (and most likely others)
 - two stage compiler
- Customizable ORBs
 - plenty! (TAO, dynamicTAO, QuarterWare, ...)
 - Spring (doors)

NEC

Conclusion

- Let IDL mappings be customizable
- Template driven IDL compiler
 - Separates parsing from mapping
 - Maintains IDL compiler as black-box
 - Automate “cookie-cutter” code generation
- Customizable Object Adapter
- Customizable protocols (like many others)
- Powerful approach
 - example TcIORB, HeidiRMI (C++, Java)

NEC

The end