ILU

(that's eye-loo!)

Inter-Language Unification

Bill Janssen, Mike Spreitzer

<janssen@parc.xerox.com>, <spreitze@parc.xerox.com>

Xerox Palo Alto Research Center

ILU info: ftp://ftp.parc.xerox.com/pub/ilu/ilu.html
This document: http://parcweb.parc.xerox.com/project/rpc/miscdoc/ILU/Generic-Talk-4.fm

ILU: Extensible Multilingual Generalized RPC

-- Neutral network interface specification language
(not least common denominator)

-- Multiple programming languages
(C, C++, Modula-3, Common Lisp, Python, Java)

-- Layered on a runtime-extensible set of RPC protocols & XDRs
(currently RFC 1831 RPC, XNS Courier RPC, IIOP, HTTP)

-- ... and a runtime-extensible set of transport protocols
(currently TCP/IP, UDP/IP; recently XNS SPP)

=> Interoperable with existing clients and servers

-- Multiple binding methods
(currently direct, RPC, buffered)

-- OS-portable
(currently POSIX, Windows; MacOS in the past)

-- a free OMG CORBA implementation ;-)

-- initial motivation: PARC internal needs
mainly: support prog. lang.s used at PARC

Bill Janssen, Mike Spreitzer
Xerox Palo Alto Research Center

ILU ISL Interface Specifation Language

Simple Types:

- boolean
- octet (a.k.a. byte)
- ISO Latin-1 & Unicode characters
- 16, 32, 64-bit signed/unsigned integers
- IEEE 32, 64, 128-bit floating point numbers

Constructed Types:

- Record
- Variable length sequence
- Multi-dimensional fixed-length array
- Enumeration
- Tagged union
- Optional (untagged union with NIL)
- Object (classical, multiple inheritance)

Exceptions -- termination model, possible associated value

Constants

Bill Janssen, Mike Spreitzer
Xerox Palo Alto Research Center

ILU Object Model

-- Classical object model - methods only, no data slots
-- Multiple inheritance (subtyping, really)

-- “Collectible” classes provide support for network
garbage collection

-- “Singleton” classes support existing RPC services

-- Methods may raise exceptions
-- Allowable exceptions must be specified

-- Method discrimination on first arg only
-- “Sibling” parameter modifier allows use of private protocols

-- Parameters may be In, Out, or InOut,
allowing multiple return values

-- ASYNCHRONOUS methods don’t wait for reply msg

-- FUNCTIONAL methods may be memo-ized

Sample ISL Interface

```
INTERFACE fs;  (* simple file server *)

TYPE UnixTime = CARDINAL;
TYPE Filename = ilu.CString;
TYPE ByteSeq = SEQUENCE OF BYTE;
TYPE OptionalString = OPTIONAL ilu.CString;
TYPE DirectoryListing = SEQUENCE OF ilu.CString;
TYPE FileInfo = RECORD
  name : ilu.CString,
  write-date : UnixTime,
  length : CARDINAL
END;

EXCEPTION NotFound : Filename;
EXCEPTION NotADirectory : Filename;
EXCEPTION OutOfBounds;

TYPE File = OBJECT BRAND "v1"
  METHODS
   FUNCTIONAL Info () : FileInfo,
   Contents (begin : CARDINAL, length : CARDINAL) : ByteSeq
     RAISES OutOfBounds END
  END;

TYPE Directory = OBJECT SUPERTYPES File END
  METHODS
   Files (pattern : OptionalString) : DirectoryListing
     RAISES NotADirectory END
  END;

TYPE Server = OBJECT
  METHODS
   FindFile (name : Filename) : File RAISES NotFound END
  END;
```

Programming Language Mappings

-- Introduce object system where lacking (i.e., C)

-- Embed multiple inheritance where lacking (i.e., M3)

-- Map to most natural expression in the language
(unless CORBA requires otherwise)

-- Work with both compiled and interpreted languages
(can omit stubs when interpreted)

-- Work with language's runtime
(kernel is parameterized by threading and synchronization primitives)

-- Several styles of main loop integration (if single-threaded)

-- Can support either user-level or kernel threads

RPC Protocols

-- Currently RFC 1831 RPC, XNS Courier, CORBA IIOP, Web HTTP

-- Others may be added at runtime

-- Clients can use different protocols for different servers, and multiple protocols for a single server

-- 'protocol meta-object' responsible for RPC messages, including choice of eXternal Data Representations, allows application-level extensibility

-- ILU layered on existing RPC (including XDR) standards: ILU constructs expressed as 'user constructs' in XDR

-- protocol meta-object responsible for layering

Transport Protocols

-- Extensible (at runtime) set

-- May be composed into 'transport stacks'; e.g.
`rfc1831rm|security|rfc1831rm|tcp`

-- Different transports stacks possible on different connections

-- 'transport stack meta-object' responsible for transporting opaque messages (byte sequences)

-- 'filter' transport layers for message processing
Currently record-marking, security, buffering
Soon compression

-- 'endpoint' transport layers for communication
Currently TCP/IP, UDP/IP, in-memory
Recently XNS SPP

-- security filter based on IETF Generic Security System API; allows arbitrary security policies to be used to protect messages

Bill Janssen, Mike Spreitzer
Xerox Palo Alto Research Center

Existing Clients & Services

-- Can write ILU client or server that interoperates with existing server or client

-- Want ISL to map *onto*, not *into*, other RPC interface languages (not always completely possible)

-- A ‘singleton’ object type has only one instance at a server (needed for existing non-OO interfaces)

Bill Janssen, Mike Spreitzer
Xerox Palo Alto Research Center

Binding Methods

-- Currently support both direct, in-memory, and RPC

-- Direct when both sides are in same address space, language
  Client uses true server directly, no stubs involved

-- In-memory for same-address-space, different languages
  Automatic foreign function calls
  Supports incremental multi-language development

-- RPC between address spaces
  User can pick the RPC protocol, or specify a new one

Bill Janssen, Mike Spreitzer
Xerox Palo Alto Research Center

CORBA and ILU

-- Similar object models

-- CORBA's OMG IDL has C++ deficiencies:

#include instead of principled import
definitions may occur outside of namespaces
C++ syntax
all objects are optional, but nothing else is optional

-- ILU extends CORBA with: ISL; network garbage collection; mappings for Common Lisp, Modula-3, Python, MOO, and TCL; large scalar types; support for existing network services; support for standard wire protocols, support for different languages in the same address space. Eventually pipes (bulk data transfer) and perhaps instance variables (with client-side cache protocol) and record inheritance.

-- ILU supports OMG object model, a useful subset (all but 'ANY') of OMG IDL, CORBA's ANSI C language mapping. No DII, Interface Repository, Implementation Repository. Partial BOA.

Current Status

2.0alpha7 is current release (18 April 1996)

Runs on most Unix systems, Windows 3, 95, & NT

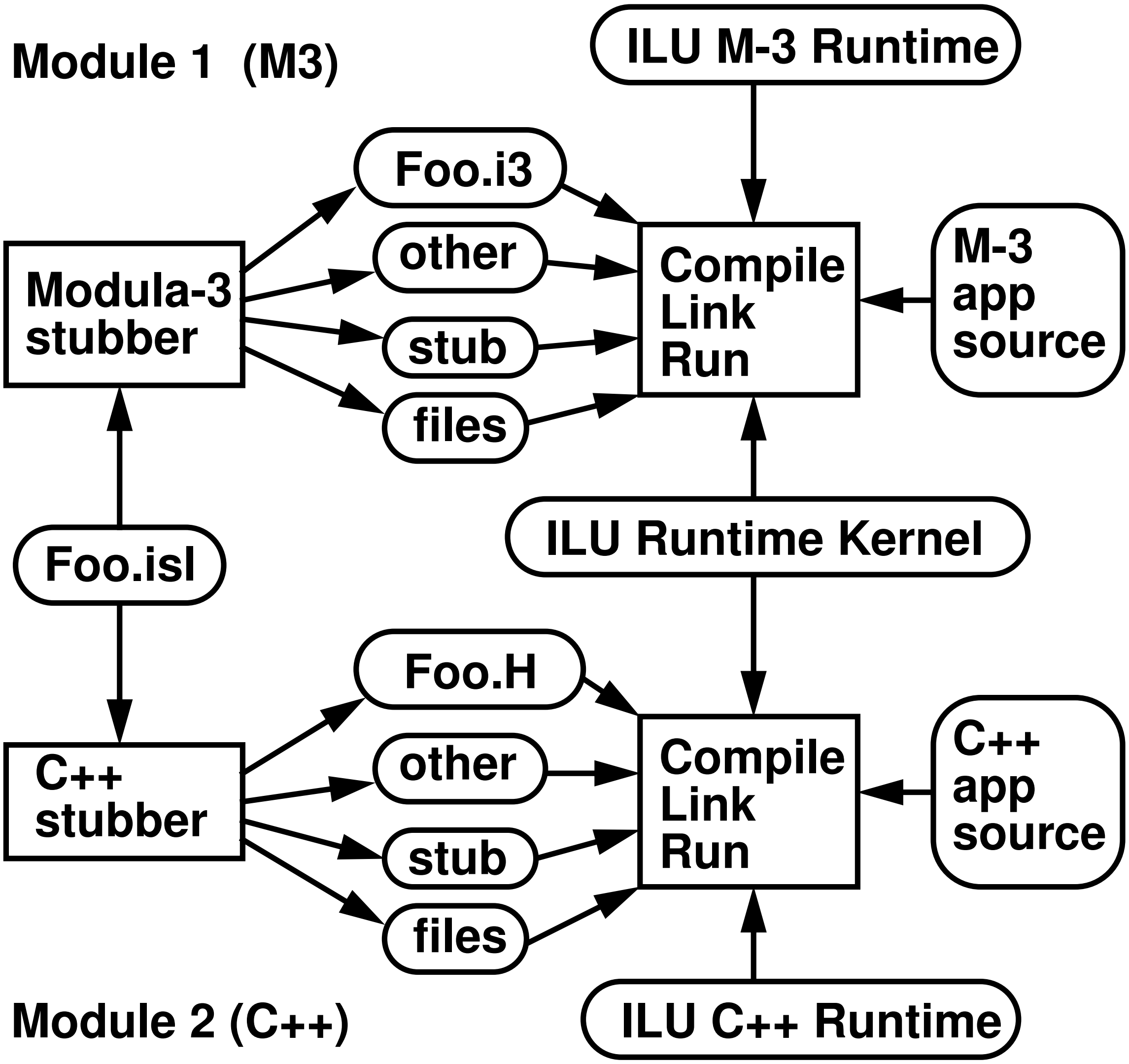
Seven people currently active in project

Available freely via anonymous FTP from PARC

All sources in Standard C; ISL parser uses GNU ‘bison’; OMG IDL parser uses Sun parser; configuration via GNU ‘autoconf’; makefiles use X Window System ‘imake’; documentation in plain text, Postscript, HTML, and GNU ‘texinfo’

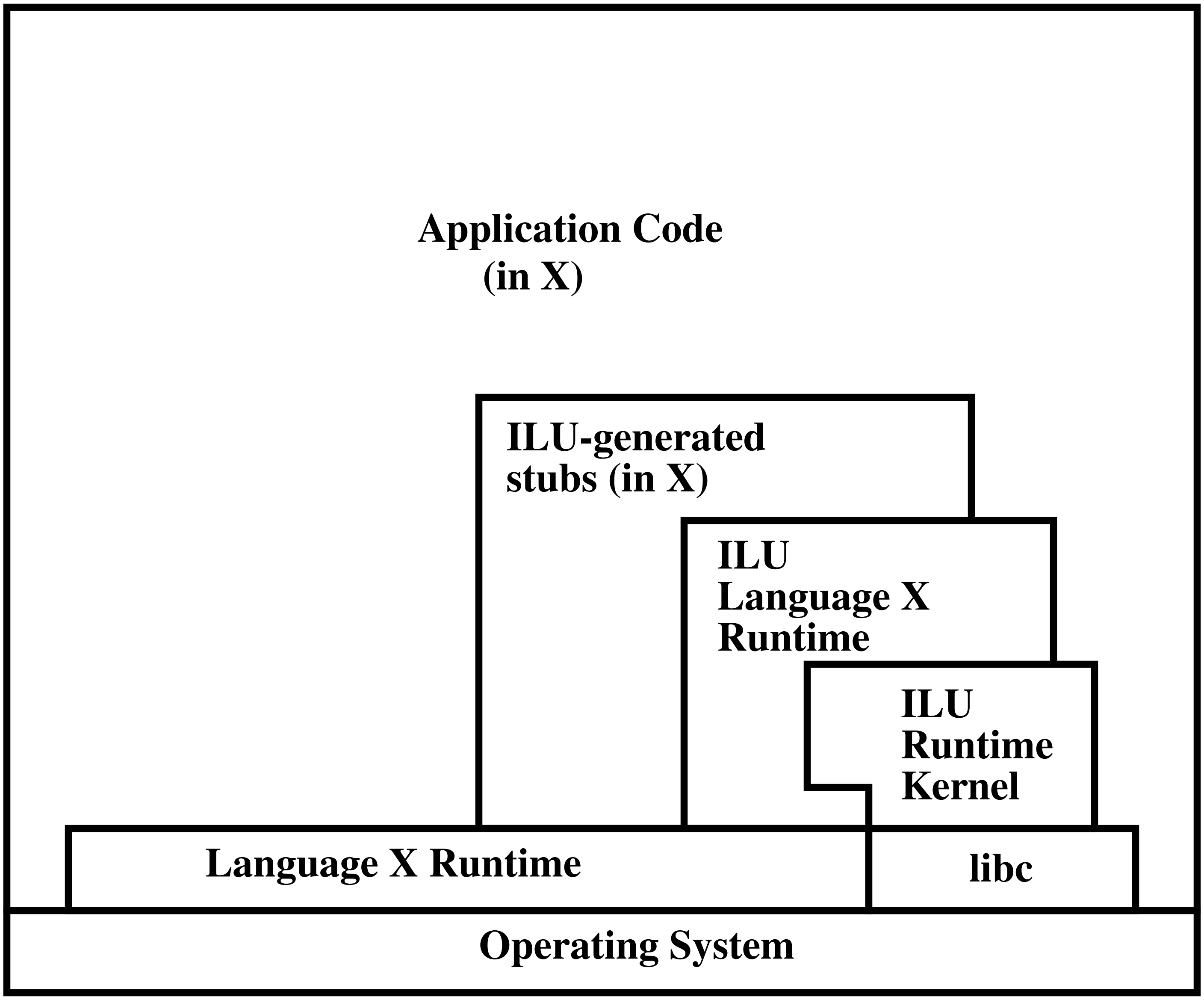
In use around the world, including WhereWereWe, Tivoli, TDB, Information Visualizer, Metro, and Bayou projects at PARC, the Stanford, University of Michigan, and University of Illinois Digital Library Initiative project, the USDAC Geoscope project, the Digital Creations ILU Responder for the Web

Currently under evaluation for the WWW Consortium’s HTTP-NG project

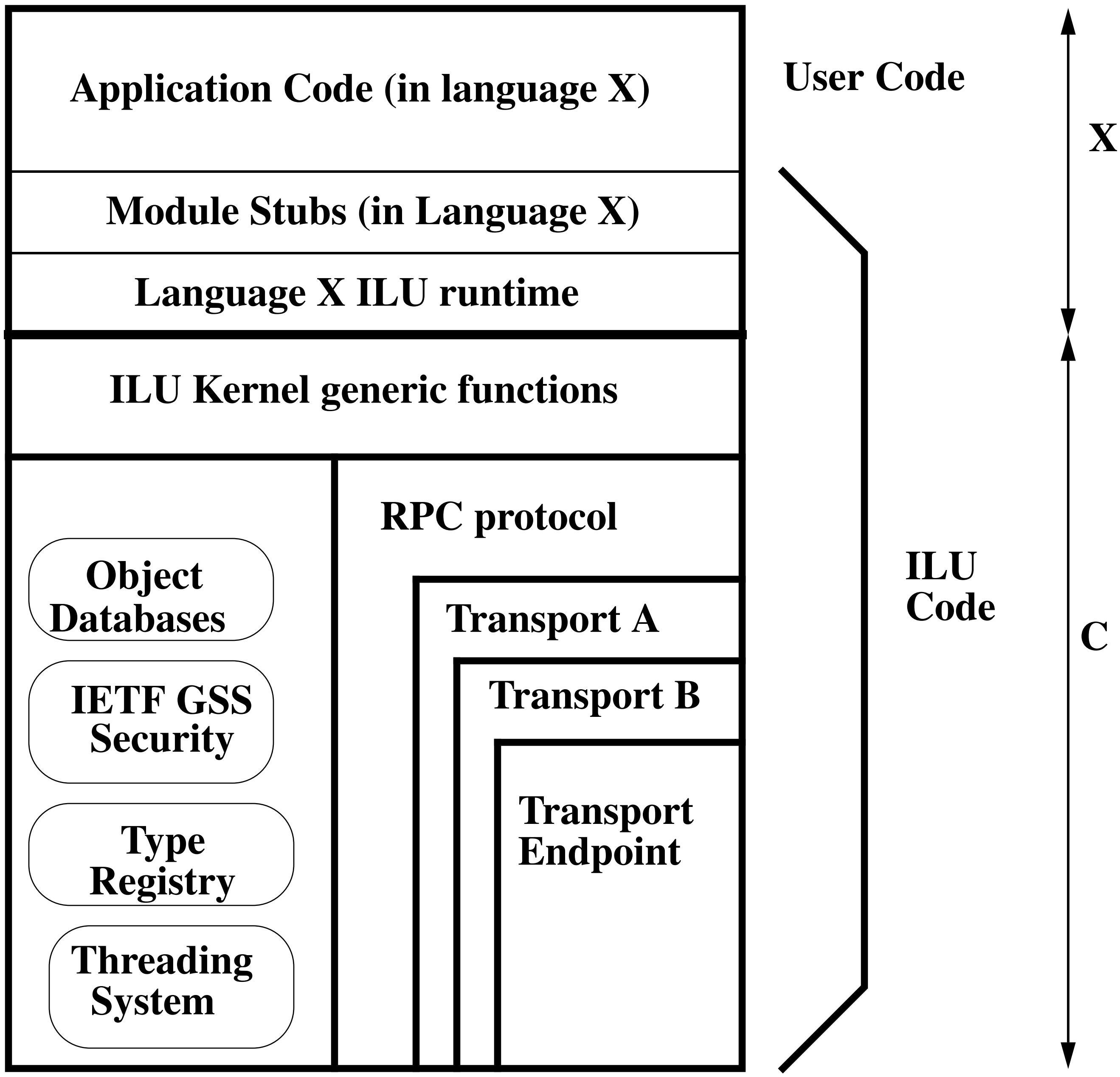


Two ILU modules in different languages

and different address spaces



Runtime structure of ILU client or server written in programming language X



An ILU Address Space