Component Models and Platforms

Object and Component
Wiring Standards
CORBA, Java, COM+/CLR

Interprocess Communication

- For a long time, interoperability of software was limited to binary calling conventions at the procedural level.
  - Every operating system defines calling conventions, and all language implementations respect the calling conventions of their platforms.
  - However, traditional operating systems don't support procedural calls across process boundaries.
  - Instead, they provide a variety of mechanisms for interprocess communication (IPC) such as files, pipes, sockets, and shared memory.
  - Except for BSD-Unix style sockets, none of these mechanisms is portable across platforms.

Interprocess Communication (2)

- An advantage of all IPC mechanisms, with the exception of shared memory, is that they can easily be extended to work across networks right up to the internet.
  - This is a direct consequence of the traditional process model, in which each process creates the illusion of a separate virtual machine on a shared physical host.
- All these IPC mechanisms operate on the level of bits and bytes.
  - Implementing complex interactions on top of such mechanisms is painful and error-prone.

Remote Procedure Calls

- The idea of RPC is to use stubs at the ends of the remote procedure and the local caller.
  - The caller uses strictly local calling conventions and seems to call a local procedure.
  - In reality, it calls a local stub that marshals (serializes) the parameters and sends them to the remote end.
  - At that end, another stub receives the parameters, unmarshals (deserializes) them, and calls the true procedure.
  - The remote procedure itself, just as the caller, follows local calling conventions and is unaware of being called remotely.

Remote Procedure Call (2)

- The stubs are responsible for converting data values from their local representation to a network format (marshalling) and from the network format to the local representation (demarshalling).
- In this way, format differences, such as byte ordering of number representations, are bridged.
- Where the RPC "glue" is automatically generated, neither clients nor providers need to be aware of non-local calling.
  - This is an advantage as it simplifies the programming.
  - A disadvantage is that it hides the significant cost difference between a local, an interprocess, and an intermachine call.

From Procedures to Objects

- It is possible to implement method calls on top of the machinery that implements procedure calls.
  - CORBA ORBs are an example.
  - Microsoft's COM is also quite close to just using procedural calling conventions.
- Another possibility is to define a new virtual machine level with built-in support for method calls.
  - This is the approach followed by the Java virtual machine (JVM) and the .NET common language runtime (CLR).
  - However, a virtual machine can prevent or hinder interoperation beyond its own boundaries. Thus, both JVM and CLR provide special support for interoperation across VM boundaries.

Component Interfaces

- All current approaches uniformly define an interface as a collection of named operations, each with a defined signature and possibly a return type. The signature of an operation defines the number, types, and passing modes of parameters.
- Traditional object models (e.g. CORBA 2) define a one-to-one relationship between interfaces and objects. Each object provides the state and implementation behind one interface.
- Other approaches associate many interfaces with a single object (Java, CLR) or many interfaces with many part objects in a component object (COM, CORBA 3 CCM - CORBA Component Model).

Component Interfaces (2)

- Traditional approaches followed the DCE (distributed computing environment) lead and used an IDL (interface definition language).
  - Unfortunately not *the* IDL, as there are several competing proposals. In particular, OMG IDL and COM IDL are the two strongest competitors.
- There is no IDL for Java or CLR.
  - Instead of learning an IDL, programmers can view interface and other type definitions using the means of their programming language.
  - The so-called Java IDL is actually a CORBA ORB callable from Java combined with an IDL-to-Java compiler.

Naming and Locating Services

- COM uses globally unique identifiers (GUIDs)
  - Similar to DCE's UUIDs.
  - GUIDs are used to name a variety of entities uniquely, including interfaces (IIDs), groups of interfaces called categories (CATIDs), and classes (CLSIDs).
- In CORBA 2.0, globally unique Repository IDs have been introduced.
  - These can either be DCE UUIDs or strings similar to the universal resource locators (URLs) used on the worldwide web.
- Java fully relies on unique name paths established by nested named packages.

Naming and Locating Services (2)

- CLR provides similar qualified names to establish a readable naming scheme, but ultimately grounds all names in so-called strong names of assemblies – in essence, the public half of a private/public key pair that is, with very high probability, unique.
- That is, whereas repository IDs back individual names with unique IDs, CLR backs entire families of names with a single unique ID, as long as such a name family is co-published in a single assembly (a CLR software component).

Naming and Locating Services (3)

- Given a name, all services provide some sort of registry or repository to help locate the corresponding service.
- On top of this directory-like function, all approaches offer some degree of meta-information on the available services.
- The minimum that is supported by all is the runtime test of the types of the offered interfaces, the runtime reflection of interfaces, and the dynamic creation of new instances.

Compound Documents

- Among the first practical approaches to software components were compound document models, e.g. Microsoft's OLE and Apple's OpenDoc.
- The concept is simple. Instead of confronting users with many different applications, each with their own idea of what a document is, users deal only with documents.
  - If parts of a document need the support of different "applications," then it is the system's problem to find and start these where needed.
  - Embedded document parts can be manipulated in place, even if they are supported by different "applications."
- The ultimate example of a compound document is the web with objects embedded in HTML pages.

XML on the Wire

- XML (extensible markup language) is a relatively simple syntax used to create structured documents.
- The important qualifier *extensible* states that the notation allows for independent extensions.
- Only introduced in 1998, XML has succeeded as a common "on the wire" data format where numerous previous attempts have failed.
  - Partially, that is because of some interesting properties of XML.
  - Partially it is because of proper timing. The arrival of XML coincided with a growing number of areas that genuinely needed standardization at that level, most of which can be summarized as e-business.

XML on the Wire (2)

- XML is useful for representing any structured or semi-structured data. Any such data structured and formatted following XML rules is called an XML document.
- Unlike the schemas known from relational databases that prefer tabular data, XML uses tree-shaped data structures.
- XML is not ideal to contain unstructured bulk data.
  - Unless the data blocks are quite small, it is preferable to keep bulk data out-of-line and refer to it from within the XML-framed structured data.

XML on the Wire (3)

- XML is commonly used for messages, web pages, configuration data, and traditional documents.
- Even if such data is never produced or read by any other application than the one the schema was defined for, using XML can still be useful.
  - Web browsers support displaying and exploring XML documents.
  - There is also a wide range of tools that offer other forms of generic support for XML-based data, including parsers, editors, and schema checkers.
- XML syntax is similar to HTML except that:
  - Elements and attributes are strictly nesting and regular.
  - XML can use arbitrary tags (extensibility).

XML Schemas

- An XML schema provides rules to constrain XML documents by defining:
  - The set of elements that can appear at the top level.
  - For each element, the set of valid attributes and nested elements.
  - For each optional attribute of an element, the default value if the attribute is omitted.
  - The allowed ordering of peer elements, both at the top and at any nesting level.
  - The required and allowed number of appearances of an element, both at the top and at any nesting level
  - For each element, whether or not it can contain unstructured text (called “CDATA”).

XML Support Standards: XPath

- Symbolic navigation through XML documents is supported by XPath.
- XPath paths can be formed using a fairly rich set of syntactic forms.
  - Elementary paths look like the paths known from file systems, with tags separated by slashes ("/").
  - As XML elements can have any number of child elements, some or all of which may share the same tag, XPath allows for contents-based selection and the formation of result sets.
  - XPath even includes a small expression language which enables operations on sets of nodes, the processing of strings, and the manipulation of names and namespace names.
- XPath is used by several other support standards.

XML Support Standards: XSL

- The XML stylesheet language (XSL) augments XML documents with presentation information.
- The XSL transform (XSLT) standard enables the definition of transformation rules that transform an XML document into another form.
  - Originally intended to guide the transformation of XML documents into presentation forms, such as HTML, XSLT is actually quite general and can be used for many other transformation purposes.
  - For example, integration servers can use XSLT to transform SOAP (simple object access protocol) requests and replies to mediate between systems conforming to different XML schemas.

DOM and SAX Models

- The XML Document Object Model (DOM) provides an interface for programs to access and manipulate XML documents.
  - DOM essentially maps every element of an XML document to an object.
  - Each object has methods to access the element's attributes, locate the parent element, and enumerate the child elements – that is, to navigate the document.
- A general assumption when using DOM is that random access is a requirement.
  - In reality, many applications require a single linear scan of an XML document to extract required information.
  - In such cases it is more efficient to use an XML streaming model such as SAX (simple API for XML).

DOM and SAX Models (2)

- SAX works by first registering various event handlers with a SAX parser and then invoking the parser, which will call these handlers as it parses the XML document.
- The traversal order is depth-first.
- It is always possible to build a DOM-style tree of objects as a side-effect of the execution of SAX-style handlers. Therefore, we can view a SAX-style interface as being at a lower level than a DOM-style one.
- DOM- and SAX-style services are available for Java, COM+, and CLR.

SOAP

- XML documents can be used as self-describing messages. An interesting application of such individual messages is remote object invocation.
- SOAP (simple object access protocol) is an XML-based standard that enables invocations on remote objects, typically using HTTP (hypertext transfer protocol).
- The SOAP standard provides standard ways to:
  - Describe the addressee of an invocation.
  - Encode a wide range of typical programming data types into invocation messages.
  - Define what parts of a message must be understood or can be ignored.

SOAP (2)

- The SOAP standard defines namespaces for SOAP envelopes and for SOAP encoding and data.
- The SOAP body contains encoded typed values. Types can be simple, such as integers and strings; or complex, including compound types such as structs or arrays.
- References between values that form part of a single SOAP message can be encoded.

SOAP (3)

- Although objects and references between objects are supported, a SOAP message cannot contain references to message-external objects.
- However, a sender can pass URLs that can be used by the receiver to locate some suitable object at the sender's end.
  - Successive requests for the same URL will generally not yield the same object. Instead of saying "use this specific object," SOAP messages say "here is where an appropriate object can be found."
  - Communicating absolute object identities would create a fragile dependency of the communicated message on the sending party's computational state.
  - Such a dependency can be preferable for synchronous communication in tightly coupled computer clusters, but it is rarely useful in web-like, loosely coupled systems.

Web Services

- Unlike traditional web servers that serve web pages for consumption by people, web services offer computational services to other systems.
  - For example, an online bookstore could offer its database as a web service in such a way that other services (or websites targeting people) can build on it.
- Web services need to be described, much like interfaces required description using an IDL. A standard for this purpose, WSDL (web services description language), builds on XML and defines a framework to describe web services.
- UDDI (universal description, discovery, and integration) is a standard directory service to discover other web services. UDDI is itself a web service.

Web Services (2)

Discovery	UDDI	Itself a web service, UDDI serves as a directory for web services.
Description	WSDL, WSFL/XLANG	Given one or more web services, describe properties at a metalevel.
Access	SOAP	Given a web service instance, access it via messages.
Transfer	HTTP, SMTP	Transfer SOAP messages.
Transport	TCP/IP, UDP	Transport data.

The core web services protocol stack