Interoperability & Middleware

David S. Rosenblum
ICS 221
Winter 2001

Interoperability

- The ability of independently-developed components to interact and cooperate with each other
- The ability of components to exchange data and services
- For example ...

American electrical plug

European electrical outlet

Why Is Interoperability Important?

- Interoperability is becoming a dominant (if not the dominant) challenge for software engineering
  - Component-based software engineering
  - Increased levels of software reuse
  - Exploiting legacy applications
  - The World Wide Web
  - Distributed software engineering
    - Engineering of distributed software
    - Distributed engineering of software
    - Engineering of software for automated distribution and deployment

Interoperability and Software Architecture

- Connectors
  - What kind of connector is needed to allow A to interoperate with B?
  - What are its properties?

The connector must resolve ***architectural mismatch*** *between A & B*

Architectural Mismatch (Garlan, Allen, Ockerbloom 1995)

- *Architectural mismatch* refers to a mismatch between assumptions made by different components about the structure of the system and the nature of the environment in which they operate

Assumptions Leading to Architectural Mismatch (I)

- Assumptions about the nature of the components
  - substrate on which component is built
  - control model
  - data model
- Assumptions about the nature of the connectors
  - protocols
  - data model

© 2001 David S. Rosenblum

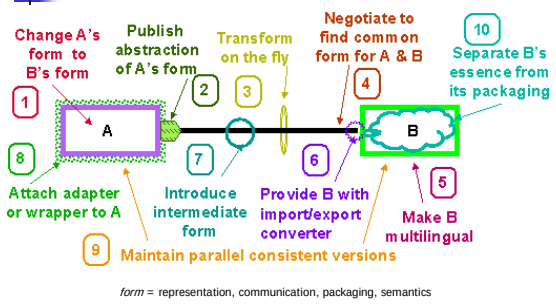
Assumptions Leading to Architectural Mismatch (II)

- Assumptions about the global configuration
  - topology
  - presence of certain components or connectors
  - absence of certain components or connectors
- Assumptions about the system construction process
  - order in which elements are instantiated
  - order in which elements are combined

Syntactic and Semantic Interoperability

- Syntactic compatibility only guarantees that data will pass through a connector properly
- Semantic compatibility is achieved only when components agree on the meaning of the data they exchange
- Example: American & European electricity
- Example: UNIX pipes
  - Syntactic compatibility established by making all data ASCII
  - Semantic compatibility is not universal
    - Line-oriented data? Field-oriented lines? Field separators?

Approaches to Achieving Interoperability (Shaw)



form = representation, communication, packaging, semantics

Approaches 1 & 2

1. Change A's form to B's form
   - Complete rewrite of A or B
   - Use standard architecture-specific frameworks
     - Component & middleware standards (OLE, ActiveX, COM, DCOM, CORBA, JavaBeans, OpenDoc)
     - Client/server builders (e.g., Powerbuilder)
     - Class frameworks (MFC)
     - UNIX pipes
2. Publish abstraction of A's form
   - Uniform distribution format
     - Adobe Acrobat, MIME, Rich Text Format (RTF)
   - Open interfaces
   - Introspection/reflection
   - Views and projections
     - This is commonly found in databases

Approach 3

3. Transform on the fly
   - Data filters
     - Convert big-endian to little-endian within communication protocol
   - Mediators
     - Encapsulate connection policies in separate integration components (Sullivan and Notkin)
     - Provide agents that synthesize information across heterogeneous data sources (Wiederhold)
   - Scripts and other externally-imposed controls
     - Event-based integration (SoftBench, ToolTalk, Castanet)
     - Scripting languages: Tcl, Perl, Javascript
     - Multimedia browsers (Mosaic, IE, Netscape)

Approaches 4 & 5

4. Negotiate common form
   - Modem protocol negotiation
5. Make B multilingual
   - Parts capable of interacting in different forms
     - Cross-platform parts
     - "Fat binaries"
     - Portable UNIX code

© 2001 David S. Rosenblum

Approach 6

6. Provide import/export converters
   - Standalone conversion tools
     - Graphic format converters (GIF/JPG/PS/PBM/PCX/TIFF)
   - Incoming/outgoing converters
     - Conversion plug-ins (MS Word/WordPerfect, MS Powerpoint/Harvard Graphics)
     - Procedure call conversions, Ada "pragma interface"
     - Marshaling/unmarshaling of data
     - Object serialization/deserialization

Approach 7

7. Introduce intermediate form
   - Exchange representations
     - Interface description language (IDL)
   - Standard distribution forms
     - RTF, PDF, HTML, external data representation

Approaches 8 & 9

8. Use wrapper
   - Wrappers and filters
     - Adapters
       - CORBA Basic Object Adapter (BOA)
       - Adapters and adapter classes in Java and JavaBeans
     - Web browsers
   - Emulation
     - WinTel emulators for Mac, SunOS
     - X Windows emulators for WinTel
9. Parallel consistent versions
   - Maintain two synchronized versions
     - Must constrain both A & B to match assumptions

Approach 10

10. Separate B's essence from its packaging
    - Separate decisions about internal functionality from decisions about "packaging" or interaction style with rest of system
      - Both these decisions are currently made by component provider
      - Both these decisions must be currently understood by system integrator

Example of Separating Packaging from Essence

- Component provider provides essence
  - compute the 200-day moving average of a stock price
- System integrator provides packaging
  - synchronous procedural invocation
  - and/or asynchronous event-based update
  - and/or ActiveX control
  - and/or CORBA IDL-based stub/skeleton interface
  - ...

Distributed Object Technology and Middleware

- An layer of software that resides between applications and the network in order to facilitate interoperability of distributed application components
- An important enabling technology for interoperability in distributed software systems
- Marriage of client/server technology with object-oriented design and analysis
- The "objects" can be anything from data structures to million-line legacy systems

© 2001 David S. Rosenblum

Current Technology

- CORBA
- COM+/DCOM and their ancestors
- ~~OpenDoc~~
- ~~SOM~~
- ODP
- Enterprise JavaBeans
- MOM/Event Messaging Middleware
- Publish/Subscribe Middleware

Middleware and Architecture (I)

- Reconciling architectural models with component interoperability standards and standard design notations
  - Explicit connectors
  - Architectural style
  - Research project: Architectural modeling in UML
  - Research project: ARABICA and ROBUSTA beanboxes for JavaBeans

Middleware and Architecture (II)

- Reconciling architectural models with middleware infrastructures
  - Software engineering principle encourages architects/designers to defer implementation decisions
  - But many design decisions constrain the choice of implementation technologies
  - Research project: Architectural support for *middleware-induced styles*

Middleware-Induced Styles (Di Nitto & Rosenblum 1999)

- The choice of a specific middleware can have an impact on the architecture of the software system
  - and vice versa
- An intuitive example
  - A three-tier client/server architecture implemented on top of an event-based middleware...

Modeling Middleware-Induced Styles in ADLs

- Focus of Paper
  - Language constructs for defining styles
  - Language constructs for creating architectures starting from styles
  - Deficiencies in languages' expressive power
  - Restrictions in the semantics

Findings

- We could not find a consensus on the semantics provided for ADLs
- None of the languages we studied suits our requirements
- Why? Most ADL research has ignored
  - Issues of implementation conformance
  - Mapping to middleware technologies
    - An exception is [Dashofy, Medvidovic, Taylor 99]

© 2001 David S. Rosenblum

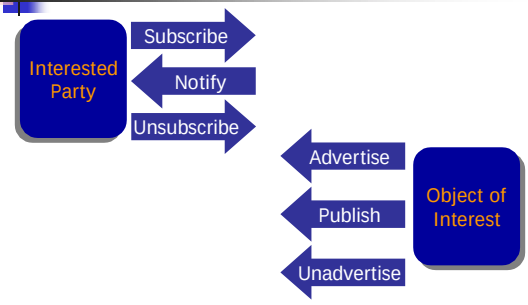
Interoperability at Internet Scale

- Huge numbers of hosts and users
- Vast numbers of events
- Heterogeneity
- Increased importance of
  - Network latency
  - Autonomy
    - Resource accounting
    - Security
  - Mobility

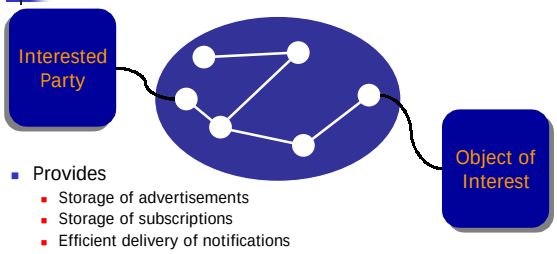
Implications of Internet Scale on Event Notification

- Event interaction becomes too burdensome for applications to manage on their own
- Thus, a separate event notification service is required
- But, event technologies originally designed for LANs don't scale

SIENA (Carzaniga, Rosenblum, Wolf 2000)



SIENA Event Notification Infrastructure



- Provides
  - Storage of advertisements
  - Storage of subscriptions
  - Efficient delivery of notifications
- Research challenge: what is an appropriate architecture for this infrastructure?

Architectural Advantages of Event-Based Interoperability

- Loose coupling between components
  - Publishers need not know identity/location of subscribers
  - Subscribers need not know identity/location of publishers
- Scalability
  - Load on publishers and subscribers is constant as number of publishers and subscribers grows
  - But the load on the pub/sub infrastructure grows …

© 2001 David S. Rosenblum