Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Contracts and Governance

XML Concepts and Techniques

XML Modeling and Storage

Summary and Directions
Architecture in IT
In the sense of information systems

- Important themes
  - Conceptualizing architecture
  - Enterprise architectures
  - Tiered architectures
  - Architecture as a basis for governance (next section)

- Not quite the same as conventional software architecture, though the topics are converging
Architecture Conceptually
As opposed to the description of a system via a blueprint

- How a system is organized
- An over-used, vaguely defined term
  - Software architecture
  - Standards, e.g., Berners-Lee’s “layer cake” and the networking standards
  - May include processes
    - That exercise the system
    - By which the system is built and maintained
    - By which the system is administered
- May include human organizations
Understanding Architecture: 1
Conception and description of a system

- Two main ingredients of a system
  - What: Components
  - How put together: Interconnections

- *Openness* entails specifying the interconnections cleanly
  - Physical components disappear
  - Their logical traces remain

- *Information environments* mean that the interconnections are protocols
Exercise: Examples of Architecture
Identify the main components and interconnections

- Buildings
- Plumbing
- Power systems
Understanding Architecture: 2

- Components and interconnections are not sufficient to characterize an architecture.
- Two additional ingredients go into an *architectural style*:
  - Constraints (hard requirements) on the components and interconnections.
  - Patterns involving the components and interconnections.
- *Openness* entails the constraints:
  - Do not apply on the physical components directly.
Exercise: Examples of Architecture
Identify the main constraints and key patterns

- Buildings
- Plumbing
- Power systems
Understanding Protocols

Protocols encapsulate IT interactions

- Interconnections over which information is the main thing that flows
  - Connect: conceptual interfaces
  - Separate: provide clean partitions among logical components
- Wherever we can identify protocols, we can
  - Make interactions explicit
  - Enhance reuse
  - Improve productivity
  - Identify new markets and technologies
- Protocols yield standards; their implementations yield products
Examples of Logical Architectural Components

Each logical component class serves some important purpose

- Power: UPS
- Network connectivity
- Storage: integrity, persistence, recovery
- Policy management
- Decision making
- Knowledge and its management

What are some products in the above component classes?
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Enterprise Architecture
Tiered Architecture
Web Architecture
Middleware
Deployment Architecture

Contracts and Governance

XML Concepts and and Techniques
The term *architecture* is used more broadly in IT settings

- The organization of an IT system
  - Bad usage: the implementation of a system
- The extensibility and modifiability of a system
  - Versus the system having no discernible structure
- Even the governance of a system, which inevitably accommodates the human organization where the system is deployed
IT and SOA Governance
The human administration of an IT system

- IT Governance: How IT resources are administered
- SOA Governance: How services are created, deployed, removed, ...
- Goes hand-in-hand with architecture
  - Incorporates
    - The stakeholders (human organization) of a system
    - The processes through which a system is updated or upgraded
    - Nontechnical aspects, such as flows of responsibility
  - Sometimes confused with architecture, but distinct
Key determinations

- Identifying a need is inherently creative
- Developing might involve merely configuring a COTS toolsuite
Exercise: Consider Kleenex Tissue Boxes in the Graduate Program Area
Enterprise Models: Information Resources

Capture static and dynamic aspects

- **Static**: snapshot of the enterprise’s resources
  - Databases and knowledge bases
  - Applications, business processes, and the information they create, maintain, and use
  which through explicit representations enable

- **Dynamic**: ways to add or modify enterprise resources
  - Integrity validation
  - Reusability
  - Change impact analysis
  - Software engineering: Automatic database and application generation via CASE tools
Enterprise Models: Rationales

- Capture (human) organizational structure
- Document business functions
  - Rationales behind designs of databases and knowledge bases
  - Justifications for applications and business processes
Enterprise Architecture Objectives

Enable governance to support the business objectives of the enterprise

- Accommodating *change* by introducing new instances and types of
  - Users
  - Ways of interaction (e.g., ongoing push toward mobility)
  - Applications
- Administering information resources
  - Preserving prior investments by interoperating with legacy systems
  - Upgrading resources
- Developing blueprints to guide resource and application installation and decommissioning
Exercise: Creating a Web Interface for the State Health Care System for the Public
Enterprise Architecture Observations

- IT is usually not a profit center
- Continual squeeze on IT resources
  - Demand for rapid development and deployment of applications
  - Demand for greater return on investment (ROI)
- Essential tension, involving stakeholders (users and suborganizations)
  - Empowerment ensures satisfaction of their needs
  - Ad hoc approaches complicate interoperability and governance
Enterprise Architecture Principles

Business processes should drive the technical architecture

- Define dependencies and other relationships among stakeholders (including users) and suborganizations of an organization
- Message-driven approaches are desirable because they decouple system components
- Event-driven approaches are desirable because they help make a system responsive to events that are potentially visible and significant to users
Architecture Modules: Applications

Typically, frontends of business processes

- Often directly visible to users
  - Application deployment
  - Data modeling and integrity
  - Business intelligence: decision support and analytics

- More technical but indirectly visible to users
  - Interoperation and cooperation
    - Ontologies: representations of domain knowledge
  - Component and model repositories
  - Business process management
Architecture Modules: Systems

Functionality used by multiple applications

- Middleware: enabling interoperation, e.g., via messaging
- Identity management, e.g., ID across a system to support Single Sign On
- Security and audit
- Accessibility
- Policy repositories and engines
Architecture Modules: Infrastructure

- Connectivity
- Platform: hardware and operating systems
- Storage
- System management
Functionalities in a Working Enterprise System

Compare with three-tier architectures

- Presentation: user interaction
  - A large variety of concerns about device constraints and usage scenarios

- Business logic
  - Application-specific reasoning
  - General rules

- Data management
  - Ensuring integrity, e.g., entity and referential integrity (richer than storage-level integrity)
  - Enabling access under various kinds of problems, e.g., network partitions
  - Supporting recovery, e.g., application, operating system, or hardware failures
Enterprise Functionalities

Bases for choosing among architectures

For three-tier architectures

► Size of implementations
► Organizational structure: who owns what and who needs what
► Staff skill sets
  ► User Interface: usability and design
  ► Programming
  ► Database
  ► Policy tools
► Products available in the marketplace
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Enterprise Architecture
Tiered Architecture
Web Architecture
Middleware
Deployment Architecture

Contracts and Governance

XML Concepts and Techniques
One-Tier and Two-Tier Architectures

- **One tier:** monolithic systems; intertwined in the code base
  - Historically the first (usually early approaches are monolithic)
  - Common in legacy systems
  - Difficult to staff, maintain, and scale up
- **Two-tier:** separate data from presentation and business logic
  - Classical client-server (or fat client) approaches
  - Mix presentation with application business rules
  - Change management
Three-Tier Architecture

- Presentation tier or frontend
  - Provides a view to user and takes inputs
  - Invokes the same business logic regardless of interface modalities: voice, Web, small screen, ...

- Business logic tier or middle tier
  - Specifies application logic
  - Specifies business rules
    - Application-level policies
    - Inspectable
    - Modifiable

- Data tier or backend
  - Stores and provides access to data
  - Protects integrity of data via concurrency control and recovery
Multitier Architecture
Also known as n-tier

- Best understood as a componentized version of three-tier architecture where
  - Functionality is assembled from parts, which may themselves be assembled
  - Supports greater reuse and enables greater dynamism
  - But only if the semantics is characterized properly
- Famous subclass: service-oriented architecture
  - Enterprise version: when the services are internal
  - General: some services are external
Architectural Tiers Evaluated
The tiers reflect logical, not physical partitioning

- The more open the architecture the greater the decoupling among components
  - Improves development through reuse
  - Enables composition of components
  - Facilitates governance, including scaling up of resources
  - Sets boundaries for organizational control
- In a narrow sense, having more moving parts can complicate administration
- But improved architecture facilitates administration through divide and conquer
XML-Based Information System

Let’s place XML in a multitier architecture
How About Database Triggers?

▶ **Pros**: essential for achieving high efficiency
  ▶ Reduce network load and materializing and serializing costs
  ▶ Leave the heavy logic in the database, under the care of the DBA

▶ **Cons**: rarely port well across vendors
  ▶ Difficult to introduce and manage because of DBA control
  ▶ Business rules are context-sensitive and cannot always be applied regardless of how the data is modified
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT
  Enterprise Architecture
  Tiered Architecture
  Web Architecture
  Middleware
  Deployment Architecture

Contracts and Governance

XML Concepts and Techniques
Web Architecture

Principles and constraints that characterize Web-based information systems

- URI: Uniform Resource Identifier
- HTTP: HyperText Transfer Protocol
- Metadata must be recognized and respected
  - Enables making resources comprehensible across administrative domains
  - Difficult to enforce unless the metadata is itself suitably formalized
Uniform Resource Identifier: 1

- URIs are abstract
- What matters is their (purported) uniqueness
- URIs have no proper syntax per se
- Kinds of URIs include
  - URLs, as in browsing: not used in standards any more
  - URNs, which leave the mapping of names to locations up in the air
Uniform Resource Identifier: 2

Good design requirements

- Ensure that the identified resource can be located
- Ensure uniqueness: eliminate the possibility of conflicts through appropriate organizational and technical means
- Prevent ambiguity
- Use an established URI scheme where possible
HTTP: HyperText Transfer Protocol

Intended meanings are quite strict, though not constrained by implementations

- Text-based, stateless
- Key verbs (and some others)
  - Get
  - Post
  - Put
- Error messages for specific situations, such as resources not available, redirected, permanently moved, and so on

ReST: Representational State Transfer
Representational State Transfer

ReST is an architectural style for networked systems that constrains the connectors

- Models the Web as a network of hyperlinked resources, each identified by a URI
- Models a Web application as a (virtual) state machine
- A client selecting a link effects a state transition, resulting in receiving the next page (next state) of the application
Characteristics of ReST

- Client-Server
- Statelessness: in terms of sessions
  - What is an advantage of statelessness?
  - Where is the session state kept then?
- Focus on resources being manipulated and their representations being transferred
- Uniform Interface: URIs, hypermedia
- Caching: responses can be labeled as cacheable
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT

Enterprise Architecture
Tiered Architecture
Web Architecture
Middleware
Deployment Architecture

Contracts and Governance

XML Concepts and Techniques
Middleware Practically
Components with routine, reusable functionality

- Abstracted from the application logic or the backend systems
- Any functionality that is being repeated is a candidate for being factored out into middleware
- Enables plugging in endpoints (e.g., clients and servers) according to the stated protocols
- Often preloaded on an application server
- Simplify programmer’s task and enable refinements and optimizations
Middleware Conceptually
Components that implement important architectural interfaces

Key examples: transaction and persistence

- **Explicit:**
  - Invoke specialized APIs explicitly
  - Ties application to middleware API: Difficult to create, maintain, port

- **Implicit:**
  - Container invokes the appropriate APIs
  - Based on declarative specifications
  - Relies on request interceptions or reflection
Containers
Distributed object management

- Architectural abstraction geared for hosting business components (objects)
  - Remote method invocation
  - Threading
  - Messaging
  - Transactions
  - Connection pooling
- Implementations for JEE and .NET
JEE Technology

Vernadat

- CORBA Client
- Java Applet in Browser
- Java Applications (Swing, AWT)
- Web Browser
- XML, HTML, HTTP (SSL)
- Servlet JSP
- Entity Bean
- Session Bean
- J2EE Connector
- EJB Server
- Java Message Service
- Java Naming and Directory Interface
- Operating System (Windows, Linux, Mac, Solaris…)
- Relational DBMS
- Legacy System

RMI, IIOP

Munindar P. Singh (NCSU)
.NET Technology

Vernadat
Basic Interaction Models

Interactions among autonomous and heterogeneous parties

- Adapters: what are exposed by each party to enable interoperability
  - Sensors ↔ information
  - Effectors ⇒ actions
- Invocation-based adapters
- Message-oriented middleware
- Peer-to-peer computing
Invocation-Based Adapters: Types
Realized in distributed objects, e.g., EJB, DCOM, CORBA

- **Synchronous**: blocking method invocation
- **Asynchronous**: nonblocking (one-way) method invocation with callbacks
- **Deferred synchronous**: (in CORBA) sender proceeds independently of the receiver, but only up to a point
Invocation-Based Adapters: Execution

Execution is best effort: application must detect any problems

- At most once
- More than once is
  - OK for idempotent operations
  - Not OK otherwise: application must check
Message-Oriented Middleware

- Main varieties
  - *Queues*: point to point, support posting and reading messages
  - *Topics*: logical multicasts, support publishing and subscribing to application-specific topics; thus more flexible than queues
  - *Blackboards*: Variant of topics wherein data items can be over-written and consumed

- Can offer reliability guarantees of delivery or failure notification to sender
  - Analogous to store and forward networks
  - Usually implemented over databases
A standardized callback for messages

- Accessed only via messages, not invoked
- No need for specialized interfaces, such as `home`, `remote`, ...
- Easy interface to implement: `onMessage()` as a callback
  - Programmer defines what actions to take on receipt of a message
  - Limited message typing
- Stateless: thus no conversations
Peer-to-Peer Computing

Traditionally, hard coded into specific applications, such as file sharing

- **Symmetric client-server**: (callbacks) each party can be the client of the other
- **Asynchrony**: while the request-response paradigm corresponds to pull, asynchronous communication corresponds to push
  - Generally to place the entire intelligence on the server (pushing) side
- **Federation of equals**: (business partners) when the participants can enact the protocols they like
  - Revisited from a cross-organizational perspective
  - Business protocols being defined in terms of business interactions, not low-level messaging
Enterprise Best Practices

- Enterprise Service Bus
  - Builds on top of messaging
  - Provide orchestration as a way to realize business processes
  - Shields programmer from transport and location considerations

- Asset repository
  - Building on directory services

- Data dictionary
  - Evolving into one based on ontologies
Outline

Challenges of Electronic Business

Specification Approaches

Commitments

Architecture in IT
  Enterprise Architecture
  Tiered Architecture
  Web Architecture
  Middleware
  Deployment Architecture

Contracts and Governance

XML Concepts and Techniques
Deployment Architecture: Web Server

- Frontend
- Supports HTTP operations
- Usually multithreaded
Deployment Architecture: Application Server

- Mediates interactions between browsers and backend databases: runs computations, invoking DB transactions as needed
- Provides a venue for the business logic
- Different approaches (CGI, server scripts, servlets, Enterprise JavaBeans) with tradeoffs in
  - Overhead: OS processes versus threads
  - Scalability
  - Security
Application Server as an Architectural Abstraction

Separates business logic from infrastructure

- Load balancing
- Distribution and clustering
- Availability
- Logging and auditing
- Connection (and resource) pooling
- Security

Separate programming from administration roles
Deployment Architecture: Database Server

- Holds the data, ensuring its integrity
- Manages transactions, providing
  - Concurrency control
  - Recovery

Transaction monitors can manage transactions across database systems, but within the same administrative domain.
Data Center Architecture

- Demilitarized zone (DMZ)
  - External router
  - Load balancer

- Firewall: only the router can contact the internal network
  - Internal network
  - Web servers
  - Application servers
  - Database servers