Module 1: Introduction

- Scope
- Grading
- Policies
  - Especially, academic integrity
Scope of this Course

- Directed at computer science students
- Emphasizes concepts and theory
- Requires a moderate amount of work
- Fairly easy if you don’t let things slip
Electronic Business

- B2C: retail, finance
- B2B: supply chains (more generally, supply networks)
- Different perspectives
  - Traditionally: merchant, customer, dealmaker
  - Trends: collaboration among various parties; virtual enterprises; coalition formation

*Main technical consequence: interacting across enterprise boundaries or administrative domains*
Properties of Business Environments

- Traditional computer science deals with closed environments
- Business environments are *open*
  - Autonomy: independent action (how will the other party act?)
  - Heterogeneity: independent design (how will the other party represent information?)
  - Dynamism: independent configuration (which other party is it?)
    - Usually, also large scale
- Need flexible approaches and arms-length relationships
Autonomy

Independence of business partners

- Sociopolitical or economic reasons
  - Ownership of resources by partners
  - Control, especially of access privileges
  - Payments

- Technical reasons: opacity with respect to key features, e.g., precommit
  - Model components as autonomous to simplify interfaces “assume nothing”
  - Model components as autonomous to accommodate underlying exceptions
Heterogeneity

Independence of component designers and system architects

- Historical reasons
- Sociopolitical reasons
  - Differences in local needs
  - Difficulty of achieving agreement
- Technical reasons: difficulty in achieving homogeneity
  - Conceptual problems: cannot easily agree
  - Fragility: a slight change can mess it up
Dynamism

Independence of system configurers and administrators

- Sociopolitical reasons
  - Ownership of resources
  - Changing user preferences or economic considerations

- Technical reasons: difficulty of maintaining configurations by hand
  - Same reasons as for network administration
  - Future-proofing your system
Coherence

Think of this as an alternative to consistency

- There may be no state (of the various databases) that can be considered consistent
  - Maintaining consistency of multiple databases is difficult
  - Unexpected real-world events can knock databases out of sync with reality

- What matters is
  - Are organizational relationships preserved?
  - Are processes followed?
  - Are appropriate business rules applied?
Integration

Yields with one integrated entity

- Yields central decision making by homogeneous entity
- Requires resolving all potential inconsistencies ahead of time
- Fragile and must be repeated whenever components change

Obsolete way of thinking: tries to achieve consistency (and fails)
Locality and Interaction

A way to maintain coherence in the face of openness

- Have each local entity look after its own
  - Minimize dependence on others
  - Continually have interested parties verify the components of the state that apply to them

- Approach: replace global constraints with protocols for interaction
  - Lazy: obtain global knowledge as needed
  - Optimistic: correct rather than prevent violations
  - Inspectable: specify rules for when, where, and how to make corrections
Interoperation

Ends up with the original number of entities working together

- Yields decentralized decision making by heterogeneous entities
- Resolves inconsistencies incrementally
- Potentially robust and easy to swap out partners as needed

Also termed “light integration” (bad terminology)
Example: Selling

Update inventory, take payment, initiate shipping
- Record a sale in a sales database
- Debit the credit card (receive payment)
- Send order to shipper
- Receive OK from shipper
- Update inventory
Potential Problems

- What if the order is shipped, but the payment fails?
- What if the payment succeeds, but the order was never entered or shipped?
- What if the payments are made offline, i.e., significantly delayed?
In a Closed Environment

- Transaction processing (TP) monitors ensure that all or none of the steps are completed, and that systems eventually reach a consistent state.
- But what if the user is disconnected right after he clicks on OK? Did order succeed? What if line went dead before acknowledgment arrives? Will the user order again?
- The TP monitor cannot get the user into a consistent state.
In an Open Environment: 1

- Reliable messaging (asynchronous communication, which guarantees message delivery or failure notification)
- Maintain state: retry if needed
- Detect and repair duplicate transactions
- Engage user about credit problems

Matter of policies to ensure compliance
In an Open Environment: 2

- Not immediate consistency
- Eventual “consistency” (howsoever understood) or just coherence
- Sophisticated means to maintain shared state, e.g., conversations
Challenges

- Information system interoperation
- Business process management
- Exception handling
- Distributed decision-making
- Personalization
- Service selection (location and assessment)
Information System Interoperation

Supply chains: manage the flow of materiel among a set of manufacturers and integrators to produce goods and configurations that can be supplied to customers

- Requires the flow of information and negotiation about
  - Product specifications
  - Delivery requirements
  - Prices
Business Processes

- Modeling and optimization
  - Inventory management
  - Logistics: how to optimize and monitoring flow of materiel
Exception Conditions

Virtual enterprises to construct enterprises dynamically to provide more appropriate, packaged goods and services to common customers

- Requires the ability to
  - Construct teams
  - Enter into multiparty deals
  - Handle authorizations and commitments
  - Accommodate exceptions
- Real-world exceptions
- Compare with PL or OS exceptions
Distributed Decision-Making: 1

Manufacturing control: manage the operations of factories

- Requires intelligent decisions to
  - Plan inflow and outflow
  - Schedule resources
  - Accommodate exceptions
Automated markets as for energy distribution

- Requires abilities to
  - Set prices, place or decide on others’ bids
  - Accommodate risks
- Pricing mechanisms for rational resource allocation
Personalization

Consumer dealings to make the shopping experience a pleasant one for the customer

- Requires
  - Learning and remembering the customer’s preferences
  - Offering guidance to the customer (best if unintrusive)
  - Acting on behalf of the user without violating their autonomy
Service Selection

What are some bases for selecting the parties to deal with?

- Specify services precisely and search for them
  - How do you know they do what you think they do (ambiguity)?
  - How do you know they do what they say (trust)?

- Recommendations to help customers find relevant and high quality services
  - How do you obtain and aggregate evaluations?
Module 2: XML Representation

- Concepts
- Parsing and Validation
- Schemas
What is Metadata?

Literally, data about data

- Description of data that captures some useful property regarding its
  - Structure and meaning
  - Provenance: origins
  - Treatment as permitted or allowed: storage, representation, processing, presentation, or sharing

- Markup is metadata pertaining to media artifacts (documents, images), generally specified for suitable parsable units
Motivations for Metadata

Mediating information structure (surrogate for meaning) over time and space

- Storage: extend life of information
- Interoperation for business
- Interoperation (and storage) for regulatory reasons

General themes

- Make meaning of information explicit
- Enable reuse across applications: repurposing (compare to screen-scraping)
- Enable better tools to improve productivity

Reduce need for detailed prior agreements
Markup History

How much prior agreement do you need?

- No markup: significant prior agreement
- Comma Separated Values (CSV): no nesting
- Ad hoc tags
- SGML (Standard Generalized Markup Language): complex, few reliable tools; used for document management
- HTML (HyperText Markup Language): simplistic, fixed, unprincipled vocabulary that mixes structure and display
- XML (eXtensible Markup Language): simple, yet extensible subset of SGML to capture custom vocabularies
  - Machine processible
  - Comprehensible to people: easier debugging
Uses of XML

Supporting arms-length relationships

- Exchanging information across software components, even within an administrative domain
- Storing information in nonproprietary format
- Representing semistructured descriptions:
  - Products, services, catalogs
  - Contracts
  - Queries, requests, invocations, responses (as in SOAP): basis for Web services
Example XML Document

1. `<?xml version="1.0"?> <!-- processing instruction -->
   <topelem attr0="foo"> <!-- exactly one root -->
     <subelem attr1="v1" attr2="v2">
       Optional text (PCDATA) <!-- parsed character data -->
       <subsubelem attr1="v1" attr2="v2"/>
     </subelem>
   </topelem>
   <null_elem/>
   <short Elem attr3="v3"/>
</topelem>`
Exercise

Produce an example XML document corresponding to a directed graph
Compare with Lisp

List processing language
- S-expressions
- Cons pairs: car and cdr
- Lists as nil-terminated s-expressions
- Arbitrary structures built from few primitives
- Untyped
- Easy parsing
- Regularity of structure encourages recursion
Exercise

Produce an example XML document corresponding to

- An invoice from Locke Brothers for 100 units of door locks at $19.95, each ordered on 15 January and delivered to Custom Home Builders
- Factor in certified delivery via UPS for $200.00 on 18 January
- Factor in addresses and contact info for each party
- Factor in late payments
Meaning in XML

- Relational DBMSs work for highly structured information, but rely on column names for meaning.
- Same problem in XML (reliance on names for meaning) but better connections to richer meaning representations.
XML Namespaces: 1

- Because XML supports custom vocabularies and interoperation, there is a high risk of name collision
- A namespace is a collection of names
- Namespaces must be identical or disjoint
  - Crucial to support independent development of vocabularies
  - MAC addresses
  - Postal and telephone codes
  - Vehicle identification numbers
  - Domains as for the Internet
  - On the Web, use URIs for uniqueness
XML Namespaces: 2

```xml
<!— xml* is reserved -->
<?xml version="1.0"?>
<arbit:top xmlns="a URI" <!— default namespace -->
xmns:arbit="http://wherever.it.might.be/arbit–ns"
xmns:random="http://another.one/random–ns">
  <arbit:aElem attr1="v1" attr2="v2">
    Optional text (PCDATA)
    <arbit:bElem attr1="v1" attr2="v2"/>
  </arbit:aElem>
  <random:simple_elem/>
  <random:aElem attr3="v3"/>
</arbit:top>
```

© Munindar P. Singh, CSC 513, Spring 2010
Uniform Resource Identifier

- URIs are abstract
- What matters is their (purported) uniqueness
- URIs have no proper syntax per se

Kinds of URIs
- URLs, as in browsing: not used in standards any more
- URNs, which leave the mapping of names to locations up in the air

Good design: the URI resource exists
- Ideally, as a description of the resource in RDDL
- Use a URL or URN
RDDL

Resource Directory Description Language

- Meant to solve the problem that a URI may not have any real content, but people expect to see some (human readable) content
- Captures namespace description for people
  - XML Schema
  - Text description
Well-Formedness and Parsing

- An XML document maps to a parse tree (if well-formed; otherwise not XML)
  - Each element must end (exactly once): obvious nesting structure (one root)
  - An attribute can have at most one occurrence within an element; an attribute’s value must be a quoted string
- Well-formed XML documents can be parsed
XML InfoSet

A standardization of the low-level aspects of XML

- What an element looks like
- What an attribute looks like
- What comments and namespace references look like
- Ordering of attributes is irrelevant
- Representations of strings and characters

Primarily directed at tool vendors
Elements Versus Attributes: 1

- Elements are essential for XML: structure and expressiveness
  - Have subelements and attributes
  - Can be repeated
  - Loosely might correspond to independently existing entities
  - Can capture all there is to attributes
Elements Versus Attributes: 2

- Attributes are not essential
  - End of the road: no subelements or attributes
  - Like text; restricted to string values
  - Guaranteed unique for each element
  - Capture adjunct information about an element
  - Great as references to elements

Good idea to use in such cases to improve readability
Elements Versus Attributes: 3

```xml
<invoice>
  <price currency='USD'>
    19.95
  </price>
</invoice>

Or

```xml
<invoice amount='19.95' currency='USD'/>

Or even

```xml
<invoice amount='USD 19.95'/>
```