Electronic Commerce Technologies

CSC 513

Spring 2011

Munindar P. Singh, Professor

singh@ncsu.edu

Department of Computer Science
North Carolina State University

Mechanics

- 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

Outline

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

Outline

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 Pertaining to Functionality

- 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?

Architectural Considerations

Architecture is motivated by additional considerations besides functionality

- Instance level, nonfunctional properties such as the availability of a specific service instance
  - What if the payments are made offline, i.e., significantly delayed?
- Metalevel properties such as the maintainability of the software modules and the ease of the upgradability of the system

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 operations
- 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 Operations

Modeling and optimization

- Inventory management
- Logistics: how to optimize and monitoring flow of materiel
- Billing and accounts receivable
- Accounts payable
- Customer support

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: Closed

Manufacturing control: manage the operations of factories

- Requires intelligent decisions to
  - Plan inflow and outflow
  - Schedule resources
  - Accommodate exceptions

Distributed Decision-Making: Open

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?

Outline

The Evolution of IT

- **Applications:** Control of computations hidden in code; integration a nightmare
- **Workflows:** Control abstracted out; integration still difficult
- **Standards-driven orchestration:** Integration improved; limited support for autonomy
- **Messaging:** Integration simplified by MoM and transformations; limited support for autonomy
- **Choreography:** Model conversations over messages; limited support for autonomy
- **Governance:** Administer resources via interactions among autonomous parties

Technical Service

- Generally, an abstraction of a computational object
  - Traditional, as in web or grid services
  - Improved: Abstraction of a "capability"
- Well encapsulated, i.e., a black box
- Interface defined at the level of methods or messages

Service Engagement

An aggregation of business relationships

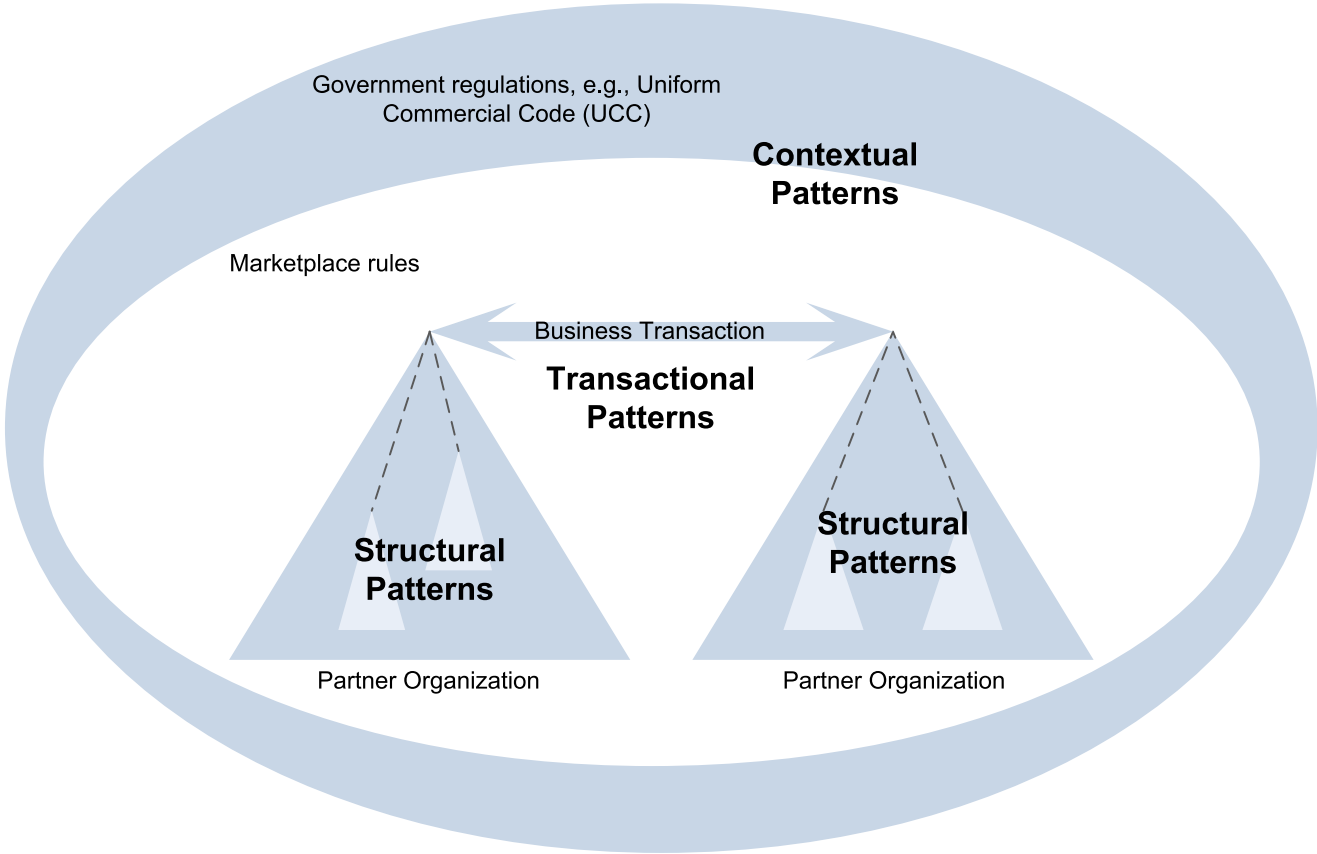
- Trillions of dollars worth of commerce conducted every year
- Characterized by
  - Independence of business partners
  - Coproduction
    - Participation by all, though not at the same level
    - Symmetric relationships: complementary capabilities and goals
  - Complex contracts among the partners
  - Participants are not black boxes

Business Service

Participant in a service engagement

- Characterized by transfer of value, not bits
- Typically long-lived with on demand enactments
- Instantiated on the fly
  - Unlike a product
  - Though may be constructed using products or about products

Conceptual Elements of a Service Engagement



- Transactional: main purpose and enactment, specifying value exchanged
- Structural: partnerships and contracts
- Contextual: setting of the engagement

Traditional Technical Approaches

Quite unlike a real-life service engagement

- Take participants flows (e.g., in BPEL, BPMN) as units of abstraction
  - Mix private policies and public interactions
  - Proprietary: may not be available for reuse
  - Context-laden: even when available, cannot be readily reused
- Focus on low-level (e.g., WS-CDL) or data-level meanings (e.g., OWL)
  - Ignore business-level significance of messages
  - Ambiguous; not verifiable

BPEL, BPMN, WS-CDL, OWL are well-known standards

A Real-Life Service Engagement

Operationally over-specified as interacting flows