Service-Oriented Computing
CSC 450 and CSC 750

Munindar P. Singh, Professor
singh@ncsu.edu

Department of Computer Science
North Carolina State University

Fall 2017

Bio Highlights and Humble Bragging

- ▶ Students
    - ▶ Graduated PhD: 25; MS: 27
    - ▶ Inaugural Alumni Hall of Fame: Nirmit Desai, Pınar Yolum
    - ▶ Inaugural Faces of Computer Science (EB2 hall): Chris Hazard
    - ▶ Associate Editors: Amit Chopra, Michael Maximilien, Pınar Yolum
    - ▶ CGS MS Thesis Award: Payal Chakravarty; also, nominee: Anup Kalia
    - ▶ Dept awards, 2017 (3 out of 5): Nirav Ajmeri, Hui Guo, Pradeep Murukannaiah
    - ▶ Dept awards, 2016 (1 out of 5): Pradeep Murukannaiah
- ▶ NCSU Internal
    - ▶ Alumni Distinguished Graduate Professor
    - ▶ Research Leadership Academy
    - ▶ Outstanding Research Achievement Award
- ▶ External
    - ▶ Fellow, Association for the Advancement of Artificial Intelligence
    - ▶ Fellow, Institute of Electrical and Electronics Engineers
    - ▶ Editor in Chief
        - ▶ ACM Transactions on Internet Technology
        - ▶ IEEE Internet Computing

My Goal and Request for Your Help

- ▶ Introduce you to deep concepts, some years in the making in the research and advanced development community
- ▶ Introduce you to critical thinking
- ▶ Boost your confidence in taking on technical challenges
    - ▶ You might hesitate to take on otherwise
    - ▶ Your peer group might find overwhelming
- ▶ Offer free advice (worth every penny^{SM}) about your
    - ▶ Education
    - ▶ Career
- ▶ How you can help
    - ▶ Don't take ethically dubious actions
    - ▶ Stay engaged
    - ▶ Communicate with me personally, especially about
        - ▶ Explanations and motivations
        - ▶ Improvements to the course, in general
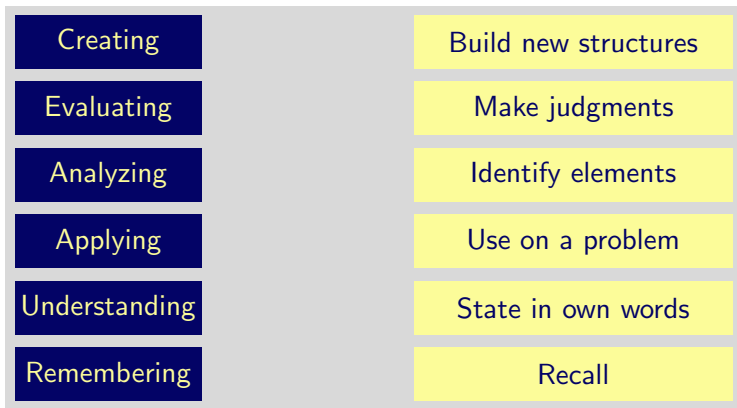
Mechanics

- ▶ Scope
- ▶ Grading
- ▶ Policies
    - ▶ Especially, academic integrity
    - ▶ Don't help; don't take help; don't collude

Scope of this Course

- ▶ Directed at computer science students
- ▶ Addresses service-oriented computing, emphasizing how it differs from typical forms of computing
  - ▶ Emphasizes concepts and theory
  - ▶ Involves tools in assignments
- ▶ Requires a moderate amount of work
  - ▶ Fairly easy if you don't let your tasks slip

Bloom's Taxonomy of Learning Domains (Cognitive)

I emphasize the upper categories

Creating	Build new structures
Evaluating	Make judgments
Analyzing	Identify elements
Applying	Use on a problem
Understanding	State in own words
Remembering	Recall

▶ http://www.nwlink.com/~donclark/hrd/bloom.html

Highlights

- ▶ Visions for the Web
- ▶ Open environments
- ▶ Services
- ▶ Main themes of this course
  - ▶ Information and its meaning
  - ▶ Decentralization of computation
  - ▶ Applying standards and tools where appropriate

Historical View on Computer Architectures

Central $\Rightarrow$ Client-Server $\Rightarrow$ Peer-to-Peer $\Rightarrow$ Multiagent Systems

The Web

- ▶ Initially, and still primarily, designed for people
    - ▶ Focuses on presentation
    - ▶ People provide the meaning: no formal representation
- ▶ Low-level interactions
    - ▶ Client-server
    - ▶ Stateless
    - ▶ Focus on procedures across websites that cause avoidable coupling

The Service Web

- Support interactions among providers and consumers
    - Beyond presentation to representation of content
    - From plain representation of content to richer meanings
    - Ways of structuring complex activities
    - Ways for independent components to interoperate
    - Ways to share contextual information

Service Composition

- ▶ Providers build and launch services
- ▶ The services are understood by prospective consumers
- ▶ Consumers can discover and select suitable services
- ▶ Consumers put selected services together to create value and possible launch their compositions as new services

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
  - ▶ Challenge: how to cut across different perspectives

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
- ▶ Requirements
    - ▶ Going from one locus of control to multiple loci of control
    - ▶ Supporting flexible interaction and arms-length relationships

Open Environments Can Arise Outside of Business
Exercise: think of some examples

- ▶ Collaborative scientific computing
- ▶ Natural disaster recovery
- ▶ . . .

Autonomy

Independence of business partners

- ▶ Sociopolitical or economic (commonsense) reasons
  - ▶ Ownership of resources by partners
  - ▶ Control, especially of access privileges
  - ▶ Payments
- ▶ Technical reasons: opacity with respect to key features, e.g., precommit
- ▶ Encapsulate: Model components as autonomous to
  - ▶ Simplify interfaces "assume nothing"
  - ▶ Accommodate any 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
Fitting well with each other: 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
Becoming one

Yields with *one* integrated entity

- ▶ Yields central decision making by one homogeneous entity
- ▶ Requires resolving all potential inconsistencies ahead of time
  - ▶ Freeze Org policies into computational system
- ▶ 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: (to the extent possible) 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
Working together

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
Scenarios that would lead to inconsistency

- ▶ 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 goods arrive damaged?

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 the 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
  ▶ Impossibility of ensuring consistency and progress in an open setting

In an Open Environment: 1

- ► Fundamental need
  - ► Model the rules of encounter among the parties
  - ► Matter of policies to ensure compliance
  - ► Engage user about credit problems
- ► Underlying assumptions and approach
  - ► Reliable messaging (asynchronous communication, which guarantees message delivery or failure notification)
  - ► Maintain state: retry if needed
  - ► Detect and repair duplicate transactions

In an Open Environment: 2

▶ Sophisticated means to maintain shared state, e.g., conversations
  ▶ Coherence of local states
  ▶ Not immediate consistency, as traditional databases promise
  ▶ Eventual "consistency" (howsoever understood)

Challenges

- ▶ Information system interoperation
- ▶ Business operations
- ▶ Exception handling
- ▶ Distributed decision-making
- ▶ Personalization
- ▶ Service selection (location and assessment)

Information System Interoperation: Supply Chains

The flow of materiel and goods from manufacturers and integrators to customers

- ▶ Flow of information among these parties
- ▶ Domain independent, e.g., Universal Business Language
    - ▶ Delivery requirements
    - ▶ Prices
- ▶ Domain dependent
    - ▶ Product specifications

Business Operations
Modeling and optimization

- ▶ Typical emphasis on internal operations
  - ▶ Inventory management
  - ▶ Logistics: how to optimize and monitor flow of materiel
  - ▶ Billing and accounts receivable
  - ▶ Accounts payable
  - ▶ Customer support
- ▶ More interesting situations in cross-organizational settings
  - ▶ Rules of encounter: normative and economic mechanisms
  - ▶ Handling exceptions

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
- ▶ Economic mechanism (e.g., pricing, penalties) 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
Accommodating and benefiting from dynamism

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?