ISM 50 – Spring 2008
Final Study Notes

June 7, 2008

Disclaimer:
This note is NOT intended to cover all the material. Neither will this note provides all the related details. It only aims to help provide a general guidance for the review. The author has NO responsibility of any kinds by providing this note out of courtesy.

Please notice that the final is accumulative, so please review the material before midterm as well.

Messerschmitt Ch 5 – Client Server Computing [Lecture 10]

✦ Client-Server Architecture [pp140]

✦ Two-tier Client Server Architecture

→ client, server
→ storage or application logic may naturally reside at server side

✦ Three-Tier Client Server Architecture [pp144~146, Table 5.1]

→ shared data, application logic, presentation
→ reason: ensure the integrity of data, managing data by DBMS separately, security & reliability; data sharing among many applications

✦ Thin Clients

→ keeping application logic out of client ⟷ thin client, NC, Sun
→ Fat client, Microsoft

✦ Scalability [Lecture 10]

✦ need to serve multiple customers at the same time; growing customer base

→ result: many identical application servers; more than one databases storing different kinds of data

✦ Peer-to-Peer (P2P) [Lecture 12, slides 18, Table CS vs. P2P]
\textit{peer hosts}: desktop computers
\textit{compared w/ Client-Server (CS) architecture}: CS is of \textit{asymmetric inciting} whereas P2P is \textit{symmetric}.
\textit{what P2P good for}: file sharing, video conferencing, note that each peer could join/leave the network at any time

\textbf{Messerschmitt Ch 6 – Modularity and Layering [Lecture 12&13]}

\begin{itemize}
  \item Modularity
    \begin{itemize}
      \item decompose the system into smaller subsystems which reduces the complexity
      \item \textit{module}: subsystems after decomposition
    \end{itemize}
  \item Granularity [pp 162]
    \begin{itemize}
      \item determines the number of modules and the range of functionality of each
      \item fine or coarse
    \end{itemize}
  \item Hierarchy
    \begin{itemize}
      \item modules are themselves composed of internal modules
      \item enables to define different granularity as appropriate
    \end{itemize}
  \item Interfaces – actions, parameters, and returns
    \begin{itemize}
      \item Each module interacts with the others through its interface
      \item Interface demonstrates: (1) what functions are performed by this module; (2) how to invoke those functions
      \item interface vs. implantation
      \item \textit{Action}: simply something specific that the module does
      \item \textit{Parameters}: data that the server module expects to receive in the course of invoking that action, which customize that action
      \item \textit{Returns}: data that the client module expects to receive back
      \item *\textit{Encapsulation}: making a module’s implementation details inaccessible to other modules
    \end{itemize}
  \item Data types
    \begin{itemize}
      \item a way for modules to agree on the structure and interpretation of data
      \item a type is attached to a specific parameter or return, otherwise they are only bits
    \end{itemize}
  \item Protocols
    \begin{itemize}
      \item finite sequence of (atomic) actions required to achieve a higher level fencing
    \end{itemize}
\end{itemize}
refer to [Lecture 13, slides 17~27], the ATM example

The Layering Principle [pp172~173, Figure 6.5]

Each layer is a server to the layer above, providing a standard set of actions but not revealing the implementation. Thus, the implementation of one layer can be changed w/o affecting the layers above.

Each layer is a client to the layer below, utilizing its available actions in the course of providing its services to the layer above it.

Each layer is permitted to interact w/ only layers immediately above and below. Thus, a layer would hide the layers below it.

Layers of computing Infrastructure [pp 175, Table 6.5, Figure 6.7]

- Applications
- Application Components
- Middleware
- Operating System (OS)
- Networks
- Infrastructure: Middleware, OS, Networks

Data and Information in Layers

- Data↔Information, structure & interpretation;
- applications deal w/ information while infrastructure deals w/ data, separation of concerns
- Package of data = file (in storage context) or message (in communication)
- [pp 179, Figure 6.8], processing from information to data and then back
- Platform heterogeneity ⇒ middlewares perform the conversion of various platform-specific representations, eg. DBMS, DOM [pp181, Figure 6.10]

Spanning Layer [pp185, Figure 186]

- heterogeneous platforms lend some horizontal structure to each layer
- divides the infrastructure into quasi-independent subsystems
- eg. Internet Protocol (IP), Distributed Object Management (DOM)

Abstraction and Encapsulation

- Abstraction: generalization, ignoring or hiding details; make one more focusing on the important issues w/o distraction of details
- Encapsulation: the assurance that internal implementation details are invisible and inaccessible at the interface; again, separation of concerns
Messerschmitt Ch 7 – Computer and Communications Industries [Lecture 13]

- Infrastructure and Applications [pp 198]
  - *Infrastructure*: equipment, software available to and utilized by many networked applications - including the network, hosts, and infrastructure softwares
  - *Applications*: provide specific capabilities and features serving individual users

- Decomposition and Assembly (also covered in Ch 10)

- Components and Custom Development
  - *Components*: a subsystem that is purchased as a product from an outside company, the components must be accepted as is and the reminder of the system should be designed around it
  - Custom development: design a subsystem as a part of the whole system from scratch, can freely specify functionality and interaction

- Interoperability [Lecture 13]
  - the ability of components to interact properly to achieve some desired functionality

- Outsourcing [Lecture 13]
  - a subsystem design is contracted to an outside vendor

- System Integration [Lecture 13]
  - (1) Bring together subsystems, (2) make them work together, (3) to achieve a goal
  - requires: testing, making some modifications if necessary

- Products and Services [Lecture 13]
  - *Product*: customer installed and operated; often sold or licensed at a fixed price.
  - *Services*: Functionality provided over a wide-area network; often sold by subscription.
  - Four possibilities, \( \{\text{Product, Service}\} \times \{\text{Application, Infrastructure}\} \), eg: Microsoft Office (P, A), Windows Vista (P, I), Hotmail (S, A), Internet DNS (S, I)

- Bundled vs Unbundled Application
  - *Bundled*: an infrastructure provider bundles applications w/ their infrastructure, eg. AOL suite
  - *Unbundled*: only application service w/o providing an infrastructure, eg. Gmail
    - Adv. to user: reducing installation, integration and maintenance; contractual obligation for availability and quality; location independence
    - Adv. to provider: ongoing revenue stream supporting upgrade and maintenance; opportunity for price discrimination; usage-based revenue better aligned w/ user’s value proposition
Stovepipe and Integrated Infrastructure [pp 205, pp206 Figure 7.1 for an example]

- **Stovepipe**: a single supplier provides an all-encompassing application solution
- **Integrated Infrastructure**: an infrastructure benefiting all applications, splitting responsibility among suppliers therefore one obvious boundary would be between application and infrastructure
  
  eg. telephone network, broadcast television network; Internet

- pros and cons [pp208~209]

Vertical Integration and Diversification [pp 209~210, Table 7.3]

- For a company wishing to expand their product offerings, there are 2 ways
- **Vertical Integration**: a company makes rather than buys all the subsystems in its products, acquiring its previous suppliers; eg. IBM
- **Diversification**: a company provides products across different industry segments; eg. telecommunications service providers accumulating telephone, cable, wireless and data networking assets
  
  (Less Vertical Integration and more diversification [pp 211])

Venture Capital and Start-Ups [pp213]

- because the trend stovepipe→layering w/ open interfaces allows small firms to fully participate w/ complementary product w/o having to offer a total solution
- VCs diversify investment to avoid risks
- why successful? lower barriers to entry in software industry; agility of small firms; economic attractiveness to talents

Computing/Communications Convergence

- **Convergence**: once-independent product categories or whole industries become either competitive or complementary
- Computing vs. Communication industry
  
  - complementary: user demands networked computing applications, both computer and communication
  - competitive: Internet telephony
  - both industries have been driven away from stovepipe architecture

Standardization

- Why are they needed?
  
  - allow products/services from different suppliers to be interoperable at each interface

Reference Models and Interfaces

- decide decomposition of system, where interfaces fall; interfaces then defines the boundaries of competition
- based on reference model, functionality and interaction could then be specified
eg. the layered infrastructure architecture

De Facto and De Jure Standards

De Jure: sanctioned and actively promoted by some organization w/ jurisdiction, or by government; eg. GSM digital cellular phone, ISDN telephone interface

De Facto:

- Market power: dominant solution arising out of the market. eg. Windows OS
- Voluntary cooperation. eg. IP (internet protocols), Java

Standards Bodies

- an organization set up for the express purpose of promulgating standards. eg. ISO (International Organization for Standards)

Why do companies participate in standardization? [pp 218, Lecture 13 slide 93]

- Pool expertise in collaborative design
- Have influence on the standard
- Get technology into the standard
  - proprietary, expected financial benefits from patents
  - open, some standards are deliberately defined by leaving enough room for enhancement or extension (eg. MPEG, Motion Picture Experts Group), also companies could make profit from superior implementation of a given standard; (build up barriers to entry the market (segment))
- Reduced time to market
  - expertise
  - prototype implementation, early entry more profit

Open Standards

- Open Standard: a standard that is well documented, unencumbered by intellectual property rights and restrictions, and available to any vendor

Examples, and acronyms:

- ISO, International Organization for Standards,
  - standards body
- IETF, Internet Engineering Force
  - an example of which the standards process is ongoing and refined, extended continuously (a reason why internet technologies beats the OSI model proposed by ISO)
- MPEG, Motion Picture Experts Group
  - collaboratively design the standard
  - decoder and encoder
  - only specifies the formats of the bit stream representing the videos
  - leaves considerable room for supplies to improve coding and compression process, leave room for profit
- OMG, Object Management Group,
  - a voluntary cooperative standardization group, which is focused on object-oriented systems and enterprise computing. not a strict standard body but rather a group to recommend best technologies in relevant fields
Hayes command set
- for controlling modems
- originally was only a small part of Hayes modem products, yet later became the de facto standard for virtually all modems

Network effect & Lock-in
- Network effect: a network effect is a characteristic that causes a good or service to have a value to a potential customer which depends on the number of other customers who own the good or are users of the service. In other words, the number of prior adopters is a term in the value available to the next adopter. [Wikipedia]; externality
- eg. telephone, Microsoft
- may leads to Lock-in
- (open) standards reduce customer lock-in effect and customers can mix and match complementary products
- increase supplier lock-in, since suppliers need to address the issue of backward-compatibility limiting innovation freedom

Messerschmitt Ch 15 – Data Sharing

- DBMS
  - Database Management System
  - DBMS manages a large amount of data shared by applications, especially for organizational applications, the body of mission-critical data
  - Recall: 3-tier architecture, DBMS is typically the foundation of the 3rd tier

- Aggregation and sharing [pp 416, Figure 15.1]
  - Aggregation: two or more databases are accessed by a single application
  - Sharing: two or more applications access a common database

- Capabilities of DBMS - manage storage and processing and retrieval of information from one or more databases; maintain data integrity; access control
  - Persistence: mission-critical data must be preserved for a long time
  - Structured data model: only present to applications w/ a logic structuring of data, separation of concerns
  - Access control, Transaction support \(\Rightarrow \) data integrity

- Relational Database, table [Lecture 14]
  - Record, Field/Attribute, Keys

- SQL [Lecture 14]
SQL (Structured Query Language): forms an interface between applications and DBMS

Database Operations - SELECT, PROJECT, JOIN [pp 419, Figure 15.3]

 Markup Languages

- for documents, to represent to *internal structure* of document by *Tags*; database could only store them as a whole w/o internal structure then

 XML vs. HTML

- XML (eXtensible Markup Language)
- HTML (Hypertext Markup Language)

 Uses

- XML: emphasizes on the structure of documents
- HTML: emphasizes on formatting and presentation of documents

 Transaction Processing [Lecture 16, pp 429~430, Figure 15.8, Sidebar pp 429]

- *Transaction Processing*: the coordination of multiple resources and the shared access to common resources in a systematic and consistent way

- 4 important properties, ACID:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

 Data Warehouses

- Data warehouses vs. operational databases

  - *Data warehouse*: a decision support database that is maintained separately from the organization’s operational database
    - focusing on the modeling and analysis of data for decision makers, instead of daily operations or transaction processing
  - operational database

- OLAP vs. OLTP

  - OLTP (on-line transaction processing): daily operations on traditional RDBMS
    - customer-oriented; current, detailed data; update as main access pattern
  - OLAP (on-line analytical processing): data analysis and decision making based on data warehouse
    - market-oriented; historical, consolidated data; read-only w/ complicated queries as access pattern

 Data Mining (see slides) [Lecture 16]

- Knowledge Discovery: discover *valid, novel, potentially useful*, and ultimately *understandable patterns* in data
Application Areas

- Business: CRM, fraud detection, e-commerce, etc.
- Web: search engines, advertising, recommendations
- Government: crime detection
- Science: bioinformatics

Major data mining tasks

- classification, clustering

Messerschmitt Ch 11.2 & Ch 18

- Locating Things (Ch 11.2) [Lecture 15]

  Names

  - symbolic, character string representation
  - doesn’t locate directly; not unique
  - hierarchical names, easy to remember: soe.ucsc.edu, C:\Documents and Settings\My Document\whatever.doc

  Addresses

  - route or path to reach topologically location of the entity

  References

  - abstract representation of an entity
  - others could directly interact w/ reference w/o caring the actual address of the entity
  - eg. cell phone number is a reference, whereas wired phone number is an address

- Name services [pp 334 sidebar]

  Name service: a server maintaining a database of names and addresses.

  Domain Name System (DNS) [Lecture 17]

  - hierarchical structure of name servers
  - Domain Name: a name associated w/ a particular host that is easy for users to remember and has a hierarchical structure reflecting administrative responsibility

- IP addresses and host names [pp 497–498]

  - 4 bytes, $4 \times 8 = 32$ bits, total $2^{32}$
  - specifies the destination for routing packets
  - host name: info.sims.berkeley.edu; IP address: 128.55.156.273
  - DNS will translation host name into IP address
Hierarchy [pp 499]

- for the purpose of scalability, both host names and addresses are hierarchical
- Address hierarchical in topology
  - (network, host) form, “network of networks”
  - analogy: telephone number, country code, city code, local number
- Names hierarchical in administration
  - separated by “dot” in domain names
  - eg. info.sims.berkeley.edu where edu - the host is administrated by an educational institution; berkeley - name of the educational institution; sims - school with Berkeley; info - the particular host within the school
  - eg. ibm.com, geographically/topologically distributed yet under same domain name since all are administrated by the same organization

Routing in the Internet [pp 495~497]

- a packet is compose of: Header+Payload where Header provides the information of the packet’s destination
- **Routing**: the function of updating routing tables in packet switches, w/ the goal of forwarded packets reaching their destination while traversing the fewest communication links or encountering minimum congestion
- **Routers**: the packet switches that implementing routing
- **Packet forwarding**: In packet switches, the determination of an output link based on each packet header and a stored routing table
- **Routing Tables**: a table contains addresses and corresponding out links, consulted by the packet switches to determine the appropriate outlink for each packet

Time Division Multiplexing [Lecture 17]

- gives each connection the use of the link a fixed fraction of time
- fixed fraction of resources reserved for each connection
- **Circuit switching**

Statistical Multiplexing

- connections are not assigned fixed fraction of link
- sharing a communication link among packets from different sources and destinations, resulting more efficient use
- queuing up packets from multiple sources, then sending one after another [pp 495, Figure 18.1]
- Internet, **packet switching**

Layering of Network Architecture

- separation of concerns, again
- top-down order: Network layer, Link layer, Physical layer
Physical Layer

- convey bits over a wire

Link Layer Ethernet

- **Frame**: a set of bits, served as a distinct “package” of data
  - separation sequence to tell the beginning and end of a frame
  - error detection/correction bits

- Ethernet Medium Access Control Protocol
  - want to allow multiple hosts to share a single link
  - MAC (Medium Access Control) Protocol: avoids multiple hosts talking at the same time, collision avoidance

- Hubs and Switches
  - Hub: broadcasts frames on a link to all others connected
  - Switch: more intelligent, knows where is the destination of a frame, only forwards it to its destination

- **MAC Addresses**
  - unique identifier for each host

Network Layer

- Routing Table
- Packet Forwarding
- IP Addresses

IP Addresses vs. MAC Addresses

- IP Address: 4 bytes, $2^4 \times 8$ in total; Hierarchical; Changeable w/ location changes
- MAC Address: 6 bytes, $2^6 \times 8$ in total; Not Hierarchical; Not changeable

Encapsulation of IP packets within an Ethernet Frame

7OSI Layers

- top-down order: Application, Presentation, Session, Transport, Network, Link, Physical

Congestion Control vs. Flow Control

- **Congestion Control**: the avoidance of congestion by artificially reducing total originating traffic, link
- **Flow Control**: control of the rate at which a sender communicates data to match the rate a receiver is able to consume, receiver

Transport Protocols – TCP and UDP
TCP (Transmission Control Protocol)
- reliability: retransmission, ACK packet
- congestion control: “back-off”, when perceived congestion by loss of packets, reduces send rate by half, otherwise increase incrementally. “Additive Increase / Multiplicative Decrease”
- UDP (User Datagram Protocol): no retransmission, no congestion control. eg. audio streaming

ISP, NSP, Local Loop, Telephone Company Local Office [Lecture 17]
- ISP (Internet Service Provider)
- NSP (Network Service Provider): access to backbone networks
- Local Loop, Telephone Company Local Office [Lecture 17, slide 57]

Web Caching
- speed up web page loading by storing previously seen components locally

Messerschmitt Ch 9 – Applications and the Organization [Lecture 18]
- Buy vs. Make vs. Outsource [pp 277]
  - Buy:
    - + less time & cost;
    - + benefits of using a “standard” solution;
    - + support available;
    - - have to mold organization to application;
    - - no potential for competitive adv.
  - Make:
    - + most customizable;
    - + easier iteration between conceptualization and development;
    - - most risky;
    - - organizations may lack competency to make own application
  - Outsource:
    - + more opportunity for customizing than off-the-shelf;
    - + opportunity to reengineer the business process along w/ the application;
    - - contractor may share knowledge w/ competitors;
    - - contractor may have too much bargaining power

Application Lifecycle Model of Development [pp279~283]
- Conceptualization
  - establishes the basic objective. visions
Analysis
- describes what the application will do
- provides enough information for stakeholders to review
  - Stakeholders: management, end-users, operators and administrators, maintenance organization, suppliers and customers
- don’t make highly detailed specification
- describe scenarios in which it is used, use case

Architecture
- decomposes the application into subsystems
- defines the functionality, interaction and interfaces of subsystems

Development Evolution
- develops the details, incrementally

Testing
- a must
- if architected well, then can test subsystems independently since separation of concerns
- Alpha test: offline test of prototype
- Beta test: tested in intended environment w/ cooperative users

Deployment
- converts from previous processes if necessary (Cisco ERP, all at once), or incrementally
- train users (Frito-Lay HHC)
- data importation

Operations, Maintenance, and Upgrade
- continuous process
- maintain application, repair problems
- correct performance shortcomings (Cisco ERP)
- add features

Remarks
- ALM rarely followed precisely, may loop between stages
- Alternative: Rapid Iterative Prototyping, Cisco ERP case

Messerschmitt Ch 10 – Application Architecture
- Decomposition vs. Assembly
  - Decomposition: divide the architecture into interacting modules, implement and integrate them
  - Assembly: purchase subsystems (software components), assemble and integrate them

Object Oriented Programming (OOP) [pp305, Table 10.3]
Object attributes, behavior

- **attribute**: a numerical value or data that is externally visible, and may be changeable
- **behavior**: describes the manner in which its attributes change

- **Method**: an action available at the object interface to interact w/ other object
- **Interface**: the set of all methods and attributes of an object, usually accomplished by documentation as to the functionality of the object and its methods
- **Class**: that which is in common among a set of objects w/ the same interface and functionality

Software Reuse – Why is it important? How does OOP help?

- size and complexity of applications growing dramatically
- in order to contain costs, we need to be able to reuse pieces of software

Software Components

- reusable modules that can be bought from outside vendors

Component Assembly tools – what do they do?

- gluing/integrating components together

Visual vs. Scripting

- Scripting assembly: textual scripting language, eg. Tcl, Perl, Javascript

Software Frameworks – what do they do? examples?

- **Frameworks**: a preexisting architecture as well as components within that architecture to help developers
- **Use**: enables reuse, and ensures component interoperability
- eg. J2EE, Microsoft .NET

Case Studies

*Please read the case studies in the Reader and summarize answers for those question by yourself.*
Sun Case

- Why is the total cost of ownership (TCO) of a Windows PC much higher than the purchase price?
- What is a thin-client? Why might it have the potential to reduce the TCO?
- What are the drawbacks of having a thin-client vs. a traditional fat-client? The advantages?
- What selfish reasons does Sun have for advocating a thin-client model? Why does Microsoft prefer maintaining the dominance of the fat-client model?
- What is Java, and what advantages does it have over other languages?
- What is the N-tier Architecture?

MySQL

- What are the different segments of the database market? Which segment is MySQL strongest in? Which segment is the largest portion of the database market?
- Who are the three biggest suppliers of database management systems? What competitive advantages over the major DBMS suppliers does MySQL have in the Web Site data segment of the market?
- Why would large enterprises prefer to manage their mission-critical, enterprise-wide data with database software from one of the three major DBMS providers, rather than using MySQL’s product which is much cheaper?
- What was the open source movement? Who where part of it? In what areas? Who where threatened by it and why?
- What is a General Public License (GPL)? Why were MySQL’s customers willing to pay for the product, when they could get the product for free under a GPL?

Akamai

- Where are the bottlenecks in the Internet according to the case study?
- What is a Content Distribution Network (CDN)? What does it provide over ordinary web Caching?
- Where did Akamai locate its servers? What barriers to entry existed for a new entrant to build a CDN to compete with Akamai?
- How did EdgeSuite differ from Akamai’s FreeFlow product?
- Did Akamai choose to market its products with a direct sales force or through distribution partners? What are the advantages of each choice?
- Why did Akamai’s marketing strategy have to change when they transitioned from the Free Flow product to the Edge Suite product?
American Airlines (tentative)

- What do flight dispatchers do, and what information do they need to make their decisions? How did the dispatch automation package assist the flight dispatchers?

- What stages of development did the flight dispatch automation package go through?

- What were some of the benefits of good architecture and Object Oriented Programming in the context of the flight dispatch automation package?

- How did the flight dispatch automation package interact with AA's legacy systems?