Final Study Notes

December 8, 2008

☐ Please notice that the final is accumulative, so please review the material before midterm as well.
☐ This note could be considered only as a reference. It won’t cover all the details. Please keep this in mind when using.

Messerschmitt Ch 5 – Client Server Computing

- Client-Server Architecture [pp140]
- 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 and reliability; data sharing among many applications
- Thin Clients
  - Keeping application logic out of client \( \Rightarrow \) thin client, NC, Sun
  - Fat client, Microsoft

Messerschmitt Ch 6 – Modularity and Layering

- Modularity
  - Decompose the system into smaller subsystems which reduces the complexity
  - Module: subsystems after decomposition
- Granularity [pp 162]
  - Determines the number of modules and the range of functionality of each
  - Fine or coarse - grained
- Hierarchy
  - Modules are themselves composed of internal modules
  - Enables to define different granularity as appropriate
- Interfaces – actions, parameters, and returns
  - Each module interacts with the others through its interface
  - Interface demonstrates: (1) what functions are performed by this module; (2) how to invoke those functions
  - Interface vs. Implementation
– **Action**: simply something specific that the module does
– **Parameters**: data that the server module expects to receive in the course of invoking that action, which customize that action
– **Returns**: data that the client module expects to receive back
– **Encapsulation**: making a module’s implementation details inaccessible to other modules

• 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

![Diagram: Layered Architecture]

• Data and Information in Layers
  – Data ↔ Information, structure and interpretation;
  – Applications deal w/ information while infrastructure deals w/ data, separation of concerns
  – **Package of data** = file (in storage context) or message (in communication)
  – Platform heterogeneity ⇒ middlewares perform the conversion of various platform-specific representations
eg. DBMS, DOM [pp 181, Figure 6.10]

• 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

- **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 remainder 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**: the ability of components to interact properly to achieve some desired functionality

- **Outsourcing**: a subsystem design is contracted to an outside vendor

- **System Integration**
  - (1) Bring together subsystems, (2) make them work together, (3) to achieve a goal
  - Requires: testing, making some modifications if necessary

- **Products and Services**
  - *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: {Product, Service} × {Application, Infrastructure}
  - eg: Microsoft Office (P, A), Windows Vista (P, I), Hotmail (S, A), Internet DNS (S, I)

- **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 [pp 208 209]

- **Vertical Integration and Diversification** [pp 209 210, Table 7.3]
  - 2 ways to expand product offerings
  - **Vertical Integration**: makes rather than buys all the subsystems in its products, acquiring its previous suppliers; eg. IBM
  - **Diversification**: provides products across different industry segments; eg. telecommunications service providers accumulating telephone, cable, wireless and data networking assets
  - Why prefer less Vertical Integration? Disadv.?
    - + Competition
    - + Mix and match components
    - + Reduce lock-in
    - - Needs integration work!
  - Trend: Less Vertical Integration and more diversification [pp 211]

- **Computing/Communications Convergence** (not discussed explicitly in class)
  - **Convergence**: once-independent product categories or whole industries become either competitive or complementary
  - Computing vs. Communication industry
    - Complementary: user demands networked computing app, both computer and communication required
    - 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 (not discussed in in class)

- **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 (API)
    - Voluntary cooperation. eg. IP (internet protocols), IEEE, IETF

- **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 16]
  - Pool expertise in collaborative design
- Have influence on the standard
- Get technology into the standard
  * Proprietary: expected financial benefits from patents
  * Non-proprietary (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
  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,
  - IETF, Internet Engineering Force
  - MPEG, Motion Picture Experts Group
  - OMG, Object Management Group,
  - Hayes command set: for controlling modems

• Network effect and Lock-in
  - (Open) standards reduce customer lock-in: can mix and match complementary products
  - Increase supplier lock-in: 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 apps, especially for organizational apps, 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, Lecture 15]
  - *Aggregation*: two or more databases are accessed by a single application
  - *Sharing*: two or more applications access a common database

• Capabilities of DBMS (not discussed in details)
  - Manage storage, processing and retrieval of information from one or more databases
  - Maintain data integrity
  - Access control

• Relational Database, table [Lecture 15]
  *Record, Field/Attribute, Keys*

• SQL [Lecture 15]
  - SQL (Structured Query Language): forms an interface between applications and DBMS
  - Database Operations -SELECT, PROJECT, JOIN [pp 419, Figure 15.3]
Messerschmitt Ch 9 – Applications and the Organization [Lecture 19]

- **Buy vs. Make vs. Outsource [pp 277]**
  - **Buy:**
    + Less time and 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;
    - May lack competency to make own application
  - **Outsource:**
    + More opportunity for customizing than off-the-shelf;
    + Opportunity to re-engineer 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
    Enough information for stakeholders (all the people involved) to review
    Not highly detailed, use case
  - **Architecture**
    Decomposes the app into subsystems
    Defines the functionality, interaction and interfaces of subsystems
  - **Development Evolution**
    Develops the details, incrementally
  - **Testing**
    A must
    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 (probably from old app)
  - **Operations, Maintenance, and Upgrade**
    Continuous process
    Maintain app, 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

Ignored here since we didn’t cover those topics.
Messerschmitt Ch 18 & Lecture Slides [17, 18]

- Domain Name System (DNS) [Lecture 18]
  - IP addresses are inconvenient to memorize
  - *Domain Name*: a name associated w/ a particular host that is easy for users to remember
  - Hierarchy in Addr vs Names
    - IP addresses hierarchical in topology
    - Names hierarchical in administration

- Layering of Network Architecture

  ![Network Architecture Diagram]

- Physical Layer: convey bits over a wire
- Link Layer
  - *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 (MAC)
    - Allows multiple hosts to share a single link
    - While 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*: info for routing, (Address, Next Hop) pair as entry
  - *Routing*: the process of updating the Routing Table
    - Goal: forwarding packets closer to dest. with traversing fewest links or encountering less congestion
  - *Packet Forwarding*: transmitting each packet on the appropriate output link, based on Routing Table
  - IP Addresses (see below)
• IP Addresses vs. MAC Addresses
  – IP Address: 4 bytes, $2^{4\times8}$ in total; Hierarchical; Changeable w/ location changes
  – MAC Address: 6 bytes, $2^{6\times8}$ in total; Not Hierarchical; Not changeable

- Encapsulation of IP packets within an Ethernet Frame
- 7 OSI Layers (refer figure above, Layered Network Architecture)
  Top-down order: Application, Presentation, Session, Transport, Network, Link, Physical
- Transport Protocols – TCP and UDP
  – TCP (Transmission Control Protocol) Reliability: Re-transmission, ACK packet
  – Congestion Control: “back-off”, when perceived congestion by loss of packets, reduces send rate
  – Port Numbers: sorting packets to various app on a single host
  – UDP (User Datagram Protocol)
    No re-transmission, no Congestion Control.
    eg. audio streaming
  – ISP, NSP, Local Loop, Telephone Company Local Office [Lecture 18]
    * ISP (Internet Service Provider)
    * NSP (Network Service Provider): access to backbone networks

* Local Loop, Telephone Company Local Office [Lecture 18]
– Web Caching: speed up web page loading by storing previously seen components locally
Case Studies

**Note.** You are supposed to summarize your own answers to the Case Studies. Since Professor has spent considerable time in class discussing them, you shall be able to do that by yourself. Therefore, they will not be covered in this notes.