Overview

During System Design major policy issues are addressed:
1. Concurrency
2. Hardware/Software Mapping
3. Persistent Data Management
4. Access Control
5. Software Control
6. Boundary Conditions

1. Concurrency
- Identify concurrent threads and address concurrency issues.
- Design goal: response time, performance.
- Threads
  - A thread of control is a path through a set of state diagrams on which a single object is active at a time.
  - A thread remains within a state diagram until an object sends an event to another object and waits for another event.
  - Thread splitting: Object does a nonblocking send of an event.

Concurrency Questions
- Which objects of the object model are independent?
- What kinds of threads of control are identifiable?
- Does the system provide access to multiple users?
- Can a single request to the system be decomposed into multiple requests? Can these requests be handled in parallel?

Implementing Concurrency
- Concurrent systems can be implemented on any system that provides
  - physical concurrency (hardware)
  - logical concurrency (software)

2. Hardware Software Mapping
- This activity addresses two questions:
  - How shall we realize the subsystems: Hardware or Software?
  - How is the object model mapped on the chosen hardware & software?
    - Mapping Objects onto Reality: Processor, Memory, Input/Output
    - Mapping Associations onto Reality: Connectivity
- Much of the difficulty of designing a system comes from meeting externally-imposed hardware and software constraints.
  - Certain tasks have to be at specific locations
Mapping the Objects
- Processor issues:
  - Is the computation rate too demanding for a single processor?
  - Can we get a speedup by distributing tasks across several processors?
  - How many processors are required to maintain steady state load?
- Memory issues:
  - Is there enough memory to buffer bursts of requests?
- I/O issues:
  - Do you need an extra piece of hardware to handle the data generation rate?
  - Does the response time exceed the available communication bandwidth between subsystems or a task and a piece of hardware?

Mapping the Subsystems Associations: Connectivity
- Describe the physical connectivity of the hardware
  - Often the physical layer in ISO’s OSI Reference Model
  - Which associations in the object model are mapped to physical connections?
  - Which of the client-supplier relationships in the analysis/design model correspond to physical connections?
- Describe the logical connectivity (subsystem associations)
  - Identify associations that do not directly map into physical connections:
    - How should these associations be implemented?

Connectivity in Distributed Systems
- If the architecture is distributed, we need to describe the network architecture (communication subsystem) as well.
- Questions to ask:
  - What are the transmission media? (Ethernet, Wireless)
  - What is the Quality of Service (QoS)? What kind of communication protocols can be used?
  - Should the interaction asynchronous, synchronous or blocking?
  - What are the available bandwidth requirements between the subsystems?

A Physical Connectivity Drawing

Hardware/Software Mapping Questions
- What is the connectivity among physical units?
  - Tree, star, matrix, ring
- What is the appropriate communication protocol between the subsystems?
  - Function of required bandwidth, latency and desired reliability
- Is certain functionality already available in hardware?
- Do certain tasks require specific locations to control the hardware or to permit concurrent operation?
  - Often true for embedded systems
- General system performance question:
  - What is the desired response time?

3. Data Management
- Some objects in the models need to be persistent
- A persistent object can be realized with one of the following mechanisms
  - Data structure
    - If the data can be volatile
    - Files
    - Cheap, simple, permanent storage
    - Low level (Read, Write)
    - Applications must add code to provide suitable level of abstraction
  - Database
    - Powerful, easy to port
    - Supports multiple writers and readers
### File or Database?
- **When should you choose a file?**
  - Are the data voluminous (bit maps)?
  - Do you have lots of raw data (core dump, event trace)?
  - Do you need to keep the data only for a short time?
  - Is the information density low (archival files, history logs)?
- **When should you choose a database?**
  - Do the data require access at fine levels of details by multiple users?
  - Must the data be ported across multiple platforms (heterogeneous systems)?
  - Do multiple application programs access the data?
  - Does the data management require a lot of infrastructure?

### Database Management System
- Contains mechanisms for describing data, managing persistent storage and for providing a backup mechanism
- Provides concurrent access to the stored data
- Contains information about the data ("meta-data"), also called data schema.

### Issues When Selecting a Database (1)
- **Storage space**
  - Database require about triple the storage space of actual data
- **Response time**
  - Mode databases are I/O or communication bound (distributed databases). Response time is also affected by CPU time, locking contention and delays from frequent screen displays

### Issues When Selecting a Database (2)
- **Locking modes**
  - *Pessimistic locking*: Lock before accessing object and release when object access is complete
  - *Optimistic locking*: Reads and writes may freely occur (high concurrency!) When activity has been completed, database checks if contention has occurred. If yes, all work has been lost.
- **Administration**
  - Large databases require specially trained support staff to set up security policies, manage the disk space, prepare backups, monitor performance, adjust tuning.

### Object-Oriented Databases
- Support all fundamental object modeling concepts
  - Classes, Attributes, Methods, Associations, Inheritance
- Mapping an object model to an OO-database
  - Determine which objects are persistent.
  - Perform normal requirement analysis and object design
  - Create single attribute indices to reduce performance bottlenecks
  - Do the mapping (specific to commercially available product). Example:
    - In ObjectStore, implement classes and associations by preparing C++ declarations for each class and each association in the object model

### Relational Databases
- Based on relational algebra
- Data is presented as 2-dimensional tables. Tables have a specific number of columns and and arbitrary numbers of rows
  - Primary key: Combination of attributes that uniquely identify a row in a table. Each table should have only one primary key
  - Foreign key: Reference to a primary key in another table
- SQL is the standard language defining and manipulating tables.
Data Management Questions
- Should the data be distributed?
- Should the database be extensible?
- How often is the database accessed?
- What is the size of typical and worst case requests?
- Do the data need to be archived?
- Does the system design try to hide the location of the databases (location transparency)?
- Is there a need for a single interface to access the data?
- What is the query format?
- Should the database be relational or object-oriented?

4. Access Control
- Discusses access control
- Describes access rights for different classes of actors
- Describes how object guard against unauthorized access

Access Control Questions
- Does the system need authentication?
- If yes, what is the authentication scheme?
  - User name and password? Access control list
  - Tickets? Capability-based
- What is the user interface for authentication?
- Does the system need a network-wide name server?
- How is a service known to the rest of the system?
  - At runtime? At compile time?
  - By Port?
  - By Name?

5. Decide on Software Control
- Control flow gives the order in which things can happen in the system
- deciding this depends on whether the things can happen
  - fairly independently and in parallel (threads/tasks) or
  - only in sequence in a given order (procedural) or
  - activities one at a time with their order determined by external events (event driven)

Guidelines for choosing control flow
- activities must occur in a fixed order with few time overlaps between activities
  - choose procedural control
    - activities may occur in different orders, as determined by external requests, but usually one activity at a time
  - choose event driven control (+ central controller)
    - activities are largely independent and can be time overlapped
  - choose threads

Procedure-Driven Control Example
Event-Based System Example: MVC
- Smalltalk-80 Model-View-Controller
- Client/Server Architecture

Centralized vs. Decentralized Designs
- Should you use a centralized or decentralized design?
  - Centralized Design
    - One control object or subsystem ("spider") controls everything
    - Change in the control structure is very easy
    - Possible performance bottleneck
  - Decentralized Design
    - Control is distributed
    - Spreads out responsibility
    - Fits nicely into object-oriented development

6. Boundary Conditions
- Most of the system design effort is concerned with steady-state behavior.
- However, the system design phase must also address the initiation and finalization of the system.
  - initialization
  - termination
  - failure

Boundary Questions
- Initialization
  - Describes how the system is brought from an non-initialized state to steady-state ("startup use cases").
- Termination
  - Describes what resources are cleaned up and which systems are notified upon termination ("termination use cases").
- Failure
  - Many possible causes: Bugs, errors, external problems (power supply).
  - Good system design foresees fatal failures ("failure use cases").

Summary
In this lecture, we reviewed the activities of system design:
- Concurrency identification
- Hardware/Software mapping
- Persistent data management
- Access Control
- Software control selection
- Boundary conditions

Each of these activities revises the subsystem decomposition to address a specific issue. Once these activities are completed, the interface of the subsystems can be defined.