Nathan Tung

CS111 Chapter 1 Outline #1 & 2

- Four common problems in systems
  - Emergent properties
    - Evident only in the system as a whole rather than in individual components
    - Example: jury behavior, Millennium Bridge synchronized footsteps weakness
  - Propagation of effects
    - Small change "propagates" and causes other larger changes/problems
    - "No small changes in a large system"
    - Example: Changing wheel proportions in a car
  - Incommensurate scaling
    - Not all components of a system follow the same scale, so parts break
    - Known to limit the range of size or speed a system can handle
    - Example: Galileo's giant, larger pyramids
  - Tradeoffs
    - Maximizing different things that might overlap/contradict
    - Waterbed effect: pushing a problem down causes another to arise
    - Example: Increasing circuit's clock rate increases power consumption and timing risks
    - Binary classification: misclassifying a property that doesn't have a direct measure via proxy
    - Example: Smoke detector, spam filter
- System (complex unity formed of diverse parts for a common plan or purpose)
  - A set of interconnected components for an expected behavior observed at the interface
  - Under discussion: system; not under discussion: environment; interactions: interface
  - Point of view for distinguishing system from components: purpose vs. granularity
  - Subsystem: system in one context but a component in another
  - Computer/information system stores, processes, or communicates information
- Signs of complexity
  - Lots of components
  - Lots of interconnections
  - Many irregularities
  - Long description (Kolmogorov complexity: measured by length of shortest specification)
  - Team of designers, implementers, maintainers
- Sources of complexity
  - Number of requirements, maintaining high utilization
  - Cascading and interacting requirements
    - Example: software update that requires higher version of something else, and so on
    - Principle of escalating complexity; more requirements → more complexity
    - More generality, requirement changes, or increased scale → more complexity
      - Solution: make local changes, or patches
  - Maintaining high utilization
    - Maximizing utilization (usage) of a rare or scarce resource
      - Example: single railroad that uses a switch to allow bidirectional travel
    - Law of diminishing returns: improving goodness requires more effort next time (complexity)
- Coping with complexity
  - Modularity
    - Analyze system as a collection of interaction subsystems, or modules
    - Divide-and-conquer of the system helps cut down debugging time
      - N statements will take N^2 debugging time
      - Modularization into K parts reduces debugging time by K
  - Abstraction
    - Separation of interface and specification from internals and implementation
    - Prevents unintentional or unnecessary interconnections
    - Robustness principle: tolerate inputs, be strict on outputs (fitting)
    - Safety margin principle: keep distance from edge to prevent drastic problems

        - Shake-out: check all inputs and reject anything less than perfect
        - Production: accept any input that can be interpreted reasonably
    - Layering
        - Module organization in which we build new mechanisms (upper) based on previously-complete mechanisms (lower) as modules
        - Example: high-level interpreter and machine language; a computer system's hardware/software
    - Hierarchy
        - Assemble groups of subsystems to create larger subsystems, and so on for a tree-like structure
        - Example: large organizations, army
        - N components means a maximum of N*(N-1) interactions; reduces potential interactions
    - Naming modules
        - Choosing an implementation (or binding) can be delayed to maintain flexibility
            - Indirection: name a feature instead of implementing it
- Computer systems have slightly different problems than other systems
    - Complexity not limited by physical laws
        - Noise limits analog components, but digital ones have no noise
        - Robustness principle: range of accepted analog values > range of analog values of output whenit means digital (static discipline)
            - Tolerant inputs, strict outputs
        - Software also has no speed limit
            - Abstractions help but are "leaky," as in overflow of an integer
        - Computer systems allow composition limited only by the designer's ability to understand it
            - Establish a self-limit
    - Rate of technological change is unprecedented
        - Prices of technology have hugely decreased
            - Incommensurate scaling rule: change in system parameter by 10 requires new design
            - Mistakes in rapidly changing designs cause complexity (due to less analysis/fine tuning)
        - By the time technology is user-friendly, that technology has been outpaced
        - Legal/judicial processes take forever to address new issues and technology
        - Communications today allow people to discover and adopt new technology
- Coping with complexity (again)
    - Modularity, abstraction, layering, and hierarchy aren't enough
        - Hard to choose the right one (for each of these)
    - Iteration
        - Build simple, working system, then evolve it in small steps towards its requirements
            - Small steps reduce risk of complexity; make system easy-to-change
        - Take small steps
        - Don't rush
        - Plan for feedback
        - Study failures
    - Iteration includes the risk of losing conceptual integrity
        - Bad-news diode: bad news spreads slower than good news, thus preventing fixes
        - Wrong modularity can be hard to change
            - Unyielding foundations rule: there are multiply modules that need to be changed
            - Designers may be reluctant to "waste time and effort" in a rework
        - Second-system effect: success can lead to overconfidence and a subsequent failure
    - Keep it simple (KISS)
        - Say no to complications, despite growing technology, competition, previous successes, etc.