How Do You Design?

*Where do architectures come from?*

**Creativity**

1) Fun!
2) Fraught with peril
3) May be unnecessary
4) May yield the best

**Method**

1) Efficient in familiar terrain
2) Not always successful
3) Predictable outcome (+ & -)
4) Quality of methods varies
Objectives

- Creativity
  - Enhance your skillset
  - Provide new tools
- Method
  - Focus on highly effective techniques
- Develop judgment: when to develop novel solutions, and when to follow established method

Engineering Design Process

- Feasibility stage: identifying a set of feasible concepts for the design as a whole
- Preliminary design stage: selection and development of the best concept.
- Detailed design stage: development of engineering descriptions of the concept.
- Planning stage: evaluating and altering the concept to suit the requirements of production, distribution, consumption and product retirement.
### Potential Problems

- If the designer is unable to produce a set of feasible concepts, progress stops.
- As problems and products increase in size and complexity, the probability that any one individual can successfully perform the first steps decreases.
- The standard approach does not directly address the situation where system design is at stake, i.e. when relationship between a set of products is at issue.
- As complexity increases or the experience of the designer is not sufficient, alternative approaches to the design process must be adopted.

### Alternative Design Strategies

- Standard
  - Linear model described above
- Cyclic
  - Process can revert to an earlier stage
- Parallel
  - Independent alternatives are explored in parallel
- Adaptive (“lay tracks as you go”)
  - The next design strategy of the design activity is decided at the end of a given stage
- Incremental
  - Each stage of development is treated as a task of incrementally improving the existing design
Identifying a Viable Strategy

- Use fundamental design tools: abstraction and modularity.
  - But how?
- Inspiration, where inspiration is needed. Predictable techniques elsewhere.
  - But where is creativity required?
- Applying own experience or experience of others.

The Tools of “Software Engineering 101”

- Abstraction
  - Abstraction(1): look at details, and abstract “up” to concepts
  - Abstraction(2): choose concepts, then add detailed substructure, and move “down”
    - Example: design of a stack class

- Separation of concerns
A Few Definitions... from the *OED* Online

- **Abstraction**: “The act or process of separating in thought, of considering a thing independently of its associations; or a substance independently of its attributes; or an attribute or quality independently of the substance to which it belongs.”
- **Reification**: "The mental conversion of ... [an] abstract concept into a thing."
- **Deduction**: “The process of drawing a conclusion from a principle already known or assumed; spec. in Logic, inference by reasoning from generals to particulars; opposed to INDUCTION.”
- **Induction**: “The process of inferring a general law or principle from the observation of particular instances (opposed to DEDUCTION, q.v.).”

Abstraction and the Simple Machines

- What concepts should be chosen at the outset of a design task?
  - One technique: Search for a “simple machine” that serves as an abstraction of a potential system that will perform the required task
  - For instance, what kind of simple machine makes a software system embedded in a fax machine?
    - At core, it is basically just a little state machine.
- Simple machines provide a plausible first conception of how an application might be built.
- Every application domain has its common simple machines.
Choosing the Level and Terms of Discourse

- Any attempt to use abstraction as a tool must choose a level of discourse, and once that is chosen, must choose the terms of discourse.
- **Alternative 1**: initial level of discourse is one of the application as a whole (step-wise refinement).
- **Alternative 2**: work, initially, at a level lower than that of the whole application.
  - Once several such sub-problems are solved they can be composed together to form an overall solution
- **Alternative 3**: work, initially, at a level above that of the desired application.
  - E.g. handling simple application input with a general parser.
Separation of Concerns

- Separation of concerns is the subdivision of a problem into (hopefully) independent parts.
- The difficulties arise when the issues are either actually or apparently intertwined.
- Separations of concerns frequently involves many tradeoffs
- Total independence of concepts may not be possible.
- Key example from software architecture: separation of components (computation) from connectors (communication)

The Grand Tool: Refined Experience

- Experience must be reflected upon and refined.
- The lessons from prior work include not only the lessons of successes, but also the lessons arising from failure.
- Learn from success and failure of other engineers
  - Literature
  - Conferences
- Experience can provide that initial feasible set of “alternative arrangements for the design as a whole”.
Patterns, Styles, and DSSAs

A DSSA is an assemblage of software components
- specialized for a particular type of task (domain),
- generalized for effective use across that domain, and
- composed in a standardized structure (topology) effective for building successful applications.

Since DSSAs are specialized for a particular domain they are only of value if one exists for the domain wherein the engineer is tasked with building a new application.

DSSAs are the pre-eminent means for maximal reuse of knowledge and prior development and hence for developing a new architectural design.
Architectural Patterns

- An architectural pattern is a set of architectural design decisions that are applicable to a recurring design problem, and parameterized to account for different software development contexts in which that problem appears.
- Architectural patterns are similar to DSSAs but applied “at a lower level” and within a much narrower scope.

State-Logic-Display: Three-Tiered Pattern

- Application Examples
  - Business applications
  - Multi-player games
  - Web-based applications
Model-View-Controller (MVC)

- Objective: Separation between information, presentation and user interaction.
- When a model object value changes, a notification is sent to the view and to the controller. So that the view can update itself and the controller can modify the view if its logic so requires.
- When handling input from the user the windowing system sends the user event to the controller; If a change is required, the controller updates the model object.
Sense-Compute-Control

Objective: Structuring embedded control applications

The Lunar Lander: A Long-Running Example

- A simple computer game that first appeared in the 1960's
- Simple concept:
  - You (the pilot) control the descent rate of the Apollo-era Lunar Lander
    - Throttle setting controls descent engine
    - Limited fuel
    - Initial altitude and speed preset
  - If you land with a descent rate of < 5 fps: you win (whether there's fuel left or not)
  - "Advanced" version: joystick controls attitude & horizontal motion
An architectural style is a named collection of architectural design decisions that
- are applicable in a given development context
- constrain architectural design decisions that are specific to a particular system within that context
- elicit beneficial qualities in each resulting system
- A primary way of characterizing lessons from experience in software system design
- Reflect less domain specificity than architectural patterns
- Useful in determining everything from subroutine structure to top-level application structure
- Many styles exist and we will discuss them in detail in the next lecture
Definitions of Architectural Style

- Definition. An architectural style is a named collection of architectural design decisions that
  - are applicable in a given development context
  - constrain architectural design decisions that are specific to a particular system within that context
  - elicit beneficial qualities in each resulting system.
- Recurring organizational patterns & idioms
  - Established, shared understanding of common design forms
  - Mark of mature engineering field.
  - Shaw & Garlan
- Abstraction of recurring composition & interaction characteristics in a set of architectures
  - Taylor

Basic Properties of Styles

- A vocabulary of design elements
  - Component and connector types; data elements
  - e.g., pipes, filters, objects, servers
- A set of configuration rules
  - Topological constraints that determine allowed compositions of elements
  - e.g., a component may be connected to at most two other components
- A semantic interpretation
  - Compositions of design elements have well-defined meanings
- Possible analyses of systems built in a style
Benefits of Using Styles

- Design reuse
  - Well-understood solutions applied to new problems
- Code reuse
  - Shared implementations of invariant aspects of a style
- Understandability of system organization
  - A phrase such as "client-server" conveys a lot of information
- Interoperability
  - Supported by style standardization
- Style-specific analyses
  - Enabled by the constrained design space
- Visualizations
  - Style-specific depictions matching engineers’ mental models

Style Analysis Dimensions

- What is the design vocabulary?
  - Component and connector types
- What are the allowable structural patterns?
- What is the underlying computational model?
- What are the essential invariants of the style?
- What are common examples of its use?
- What are the (dis)advantages of using the style?
- What are the style’s specializations?
Some Common Styles

- Traditional, language-influenced styles
  - Main program and subroutines
  - Object-oriented
- Layered
  - Virtual machines
  - Client-server
- Data-flow styles
  - Batch sequential
  - Pipe and filter
- Shared memory
  - Blackboard
  - Rule based
- Interpreter
  - Interpreter
  - Mobile code
- Implicit invocation
  - Event-based
  - Publish-subscribe
- Peer-to-peer
  - “Derived” styles
  - C2
  - CORBA