A Discipline of Software Design

Objectives

- To explain what design is and how various types of design deal with different aspects of a product
- To present design as problem solving and outline the roles of abstraction and modeling in design
- To place design in the software life cycle
- To survey software engineering design methods

Topics

- Software products and software design
- Abstraction and modeling
- Varieties of design
- Software design in the life cycle
- Software engineering design methods

Importance of Software Design

- We live in a designed world.
- Design is economically important and effects our quality of life.
- Software is becoming ubiquitous.
- The quality of software design has important consequences that software designers should be aware of and take seriously.

Software Products

A **software product** is an entity comprised of one or more programs, data, and supporting materials and services that satisfies client needs and desires either as an independent artifact or as essential ingredient in some other artifact.

Software Design Defined

Software designers do what designers in other disciplines do, except they do it for software products.

Software design is the activity of specifying the mature and composition of software products that satisfy client needs and desire, subject to constraints.

Design as Problem Solving

- An especially fruitful way to think about design is as problem solving.
- Advantages
  - Suggests partitioning information between problem and solution
  - Emphasizes that there may be more than one good solution (design)
  - Suggests techniques such as changing the problem, trial and error, brainstorming, etc.

Abstraction

Abstraction is an important problem-solving technique, especially in software design.

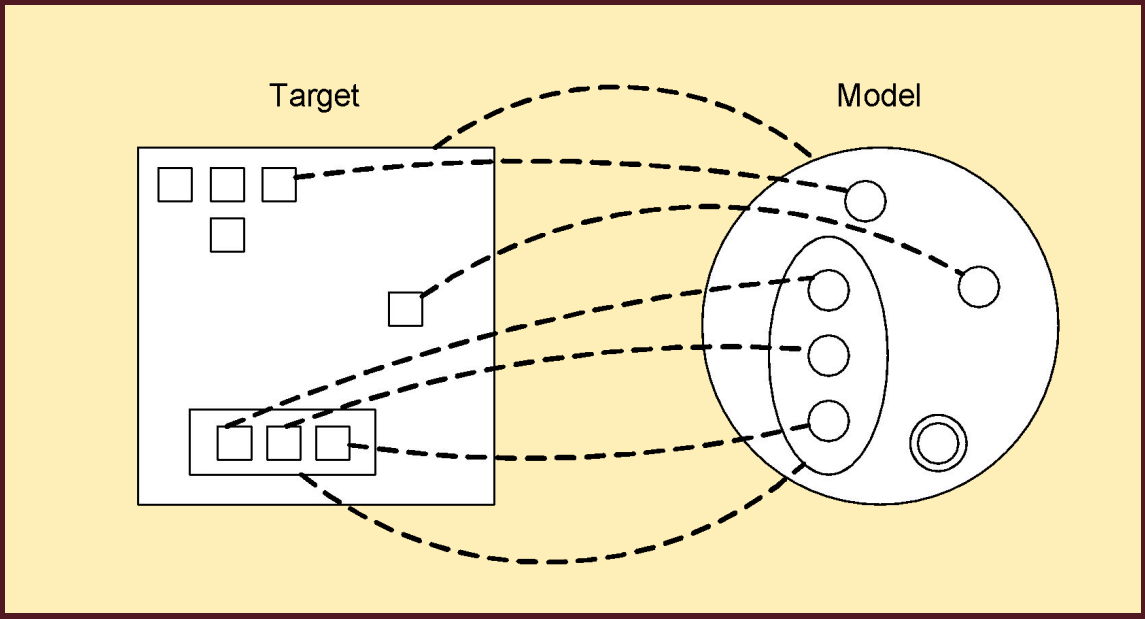
Abstraction is suppressing or ignoring some properties of objects, events, or situations in favor of others.

Importance of Abstraction

- Problem simplification
  - Abstracting allows us to focus on the most important aspects of a problem in (partially) solving it.
- Structuring problem solving
  - **Top-down strategy**: Solve an abstract version of the problem, then add details (**refinement**)
  - **Bottom-up strategy**: Solve parts of a problem and connect them for a complete solution

Modeling

A model represents a target by having model parts corresponding to target parts, with relationships between model parts corresponding to relationships between target parts.



Modeling in Design

- Modeling is used for the following purposes:
  - Problem understanding
  - Design creation
  - Design investigation
  - Documentation
- Modeling work because models abstract details of the target.
- Models can fail if important and relevant details are left out.

Static and Dynamic Models

A **static model** represents aspects of programs that do not change during program execution.

A **dynamic model** represents what happens during program execution.

- Static model examples include class and object models.
- Dynamic model examples include state diagrams and sequence diagrams.

Product vs. Engineering Design

- **Product designers** are concerned with styling and aesthetics, function and usability, manufacturability and manageability.
  - Industrial designers, (building) architects, interior designers, graphic designers, etc.
- **Engineering designers** are concerned with technical mechanisms and workings.
  - Structural, mechanical, chemical, and electrical engineers
- Design teams often include both product and engineering designers.

Software Product Design

Software product design is the activity of specifying software product features, capabilities, and interfaces to satisfy client needs and desires.

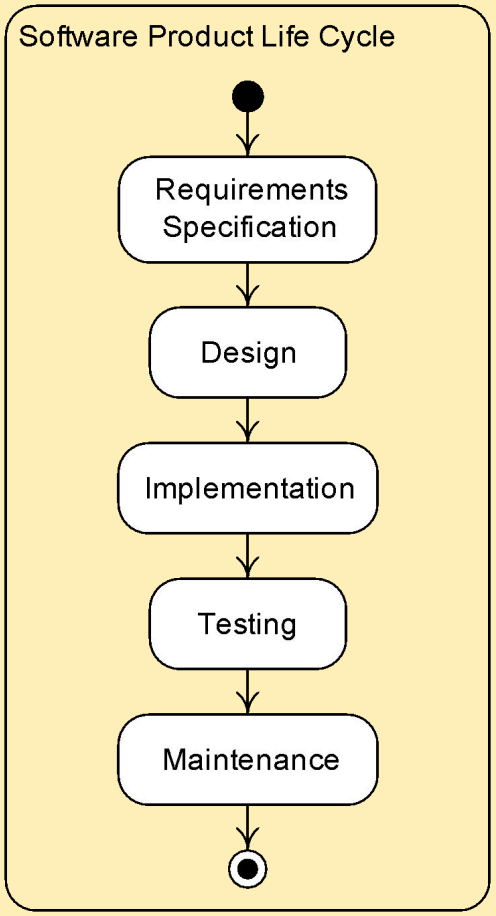
Requires skills in user interface and interaction design, communications, industrial design, and marketing

Software Engineering Design

Software engineering design is the activity of specifying programs and sub-systems, and their constituent parts and workings, to meet software product specifications.

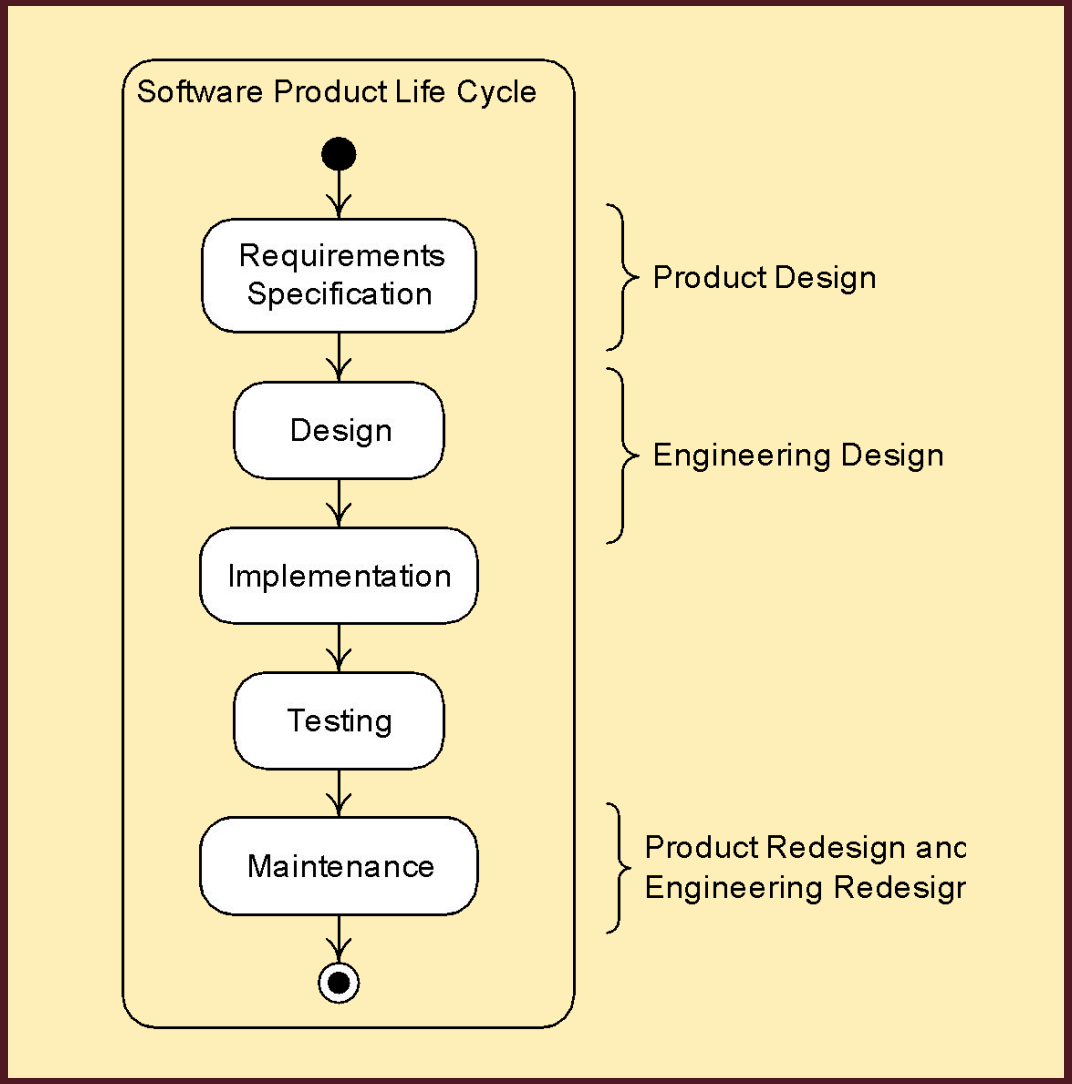
Requires skills in programming, algorithms, data structures, software design principles, practices, processes, techniques, architectures, and patterns

Waterfall Life Cycle Model



The waterfall model captures the logical, but not the temporal, relationships between software development activities.

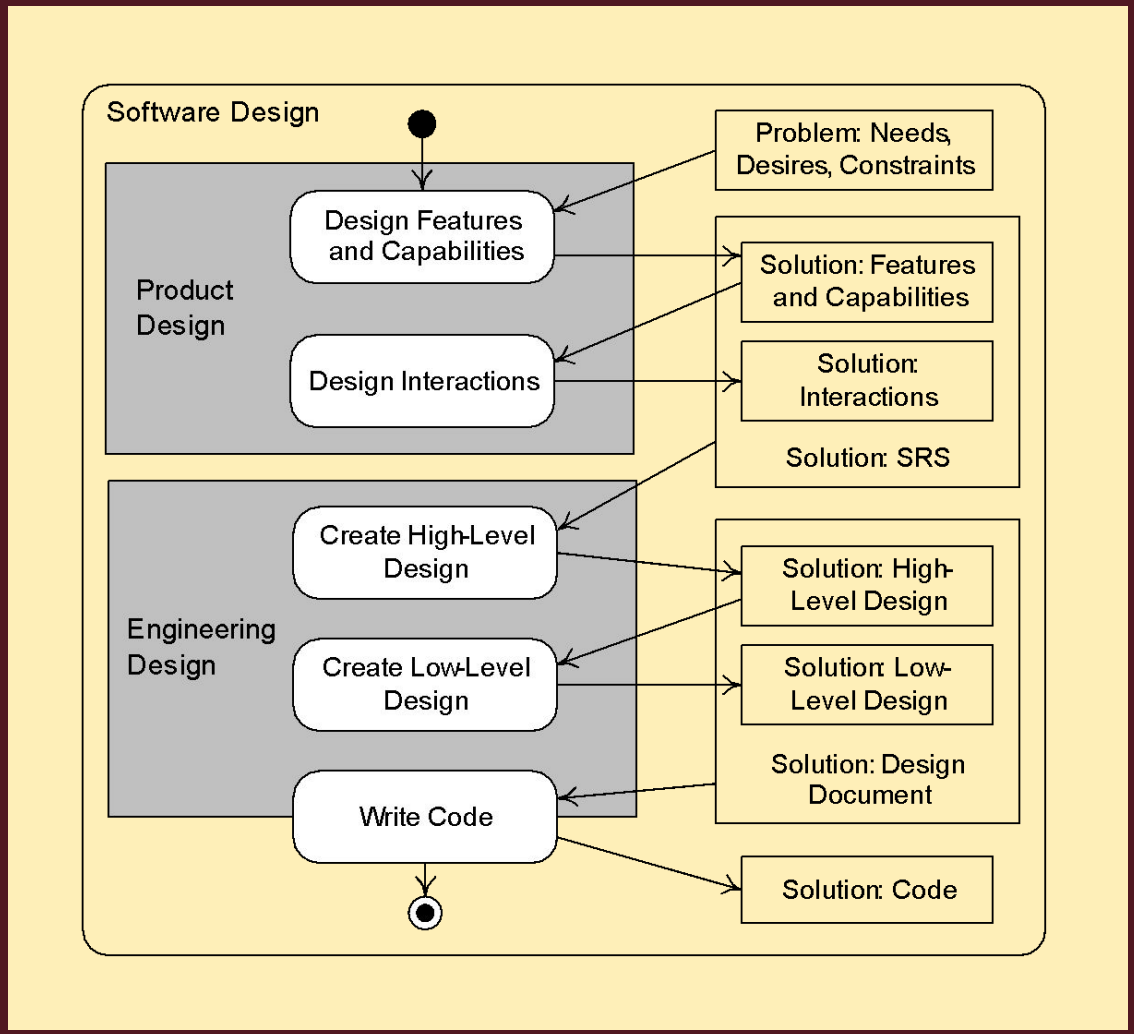
Design Across the Life Cycle



"What" Versus "How"

- Traditional way to make the distinction between requirements and design activities
- Not adequate because
  - Many "what" specifications turn out to be design decisions
  - Many "how" specifications turn out to be client or customer needs or desires
- Distinguish requirements from design based on problem solving: requirements activity formulates a problem solved in design

Design Problems and Solutions



Software Design Method

A **software design method** is an orderly procedure for generating a precise and complete software design solution that meets clients needs and constraints.

Design Method Components

- *Design Process*—A collection of related tasks that transforms a set of inputs into a set of outputs
- *Design Notations*—A symbolic representational system
- *Design Heuristics*—Rules providing guidance, but no guarantee, for achieving some end
- Design methods also use **design principles** stating characteristics of design that make them better or worse.

Design Method Timeline

1971 Niklaus Wirth introduces stepwise refinement.

1974 Stevens, Myers, Constantine introduce structured design.

Late 1970s to early 1980s Structured analysis and design methods are dominant.

Late 1980s Object-oriented analysis and design methods rise to prominence.

1995 UML 0.8 is released.

2004 UML 2.0 is released.

Summary

- Software design is important.
- Software design is best thought of as problem solving.
- Abstraction is a fundamental design technique.
- Modeling (which relies on abstraction) is a basic design tool.
- Software design comprises both product and engineering design.
- Product design occurs mainly in the requirements specification phase; engineering design mainly in the design and implementation phases.
- OO analysis and design methods are now dominant.