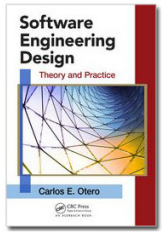
CHAPTER 1: INTRODUCTION TO SOFTWARE ENGINEERING DESIGN

SESSION II: OVERVIEW OF SOFTWARE ENGINEERING DESIGN

Software Engineering Design: Theory and Practice
by Carlos E. Otero



SESSION'S AGENDA

- Software Engineering Design
  - ✓ Why study software engineering design?
- Software design challenges
  - ✓ Requirements volatility
  - ✓ Processes
  - ✓ Technology
  - ✓ Ethical and Professional issues
  - ✓ Managing design influences
- Software design process
  - ✓ Software architecture
  - ✓ Detailed design
  - ✓ Construction design
  - ✓ HCI design

SOFTWARE ENGINEERING DESIGN

- In the previous session, we presented a general motivation for software engineering design and important concepts for designers. These included:
  - ✓ Design for large-scale, complex, systems
  - ✓ The software engineering life-cycle
  - ✓ A micro-process for problem solving, and
  - ✓ A holistic (macro) process for problem solving

- In the previous session, design was introduced as a systematic and intelligent process for generating, evaluating, and specifying designs for devices, systems, or processes.

- In software engineering, design provides blueprints that capture how software systems will meet their required functions and how they will be shaped to meet their intended quality.

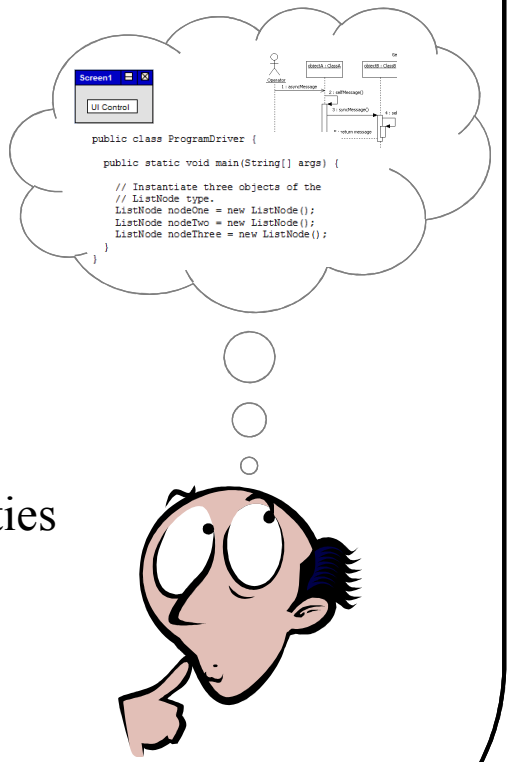
SOFTWARE ENGINEERING DESIGN

- Formally, software engineering design is defined as:
  - ✓ (1) The process of identifying, evaluating, validating, and specifying the architectural, detailed, and construction models required to build software that meets its intended functional and non-functional requirements; and,
  - ✓ (2) the result of such process.

- In the software industry, the term design is used interchangeably to describe both the *process* and *product* resulting from such process.
  - ✓ From the *process perspective*, it describes the phase, activities, tasks, and interrelationship between them required to model software's structure and behavior before construction.
    - This is concerned mostly with the project management's aspect of development.
  - ✓ From the *product perspective*, it describes artifacts resulting from design activities, e.g., class diagram.
    - This is concerned mostly with the technical aspects of development.

SOFTWARE ENGINEERING DESIGN – PRODUCT DEVELOPMENT

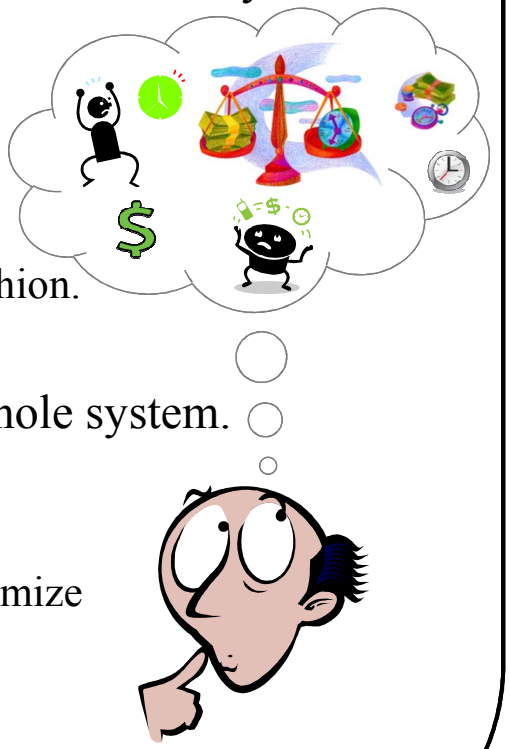
- From the product development's perspective, studying software design is important, mainly, because:
  - ✓ Requirements are mapped to conceptual models of software
  - ✓ Provide models that represent structure and behavior
  - ✓ Main components and interfaces are identified
  - ✓ Quality of future system is born
    - Modularization, cohesiveness, coupling, etc.
    - Provide means to evaluate quality attributes, e.g., usability.
  - ✓ Solutions can be reused in other projects



- Design forms the foundation for all other development activities
  - Construction
  - Testing
  - Maintenance

SOFTWARE ENGINEERING DESIGN –PROJECT MANAGEMENT

- From the project management's perspective, studying software design is important, mainly, because:
  - ✓ Good software design help minimize effects of requirements' volatility
    - This can minimize impact on schedule and cost.
  - ✓ Increases efficiency in human resource allocation.
    - Designs decompose systems so that teams can work on different parts of the problem in parallel.
    - By decomposing system, teams can work in distributed fashion.
    - These can have positive effects in schedule and cost.
  - ✓ Shields the project from having "one guy" owning the whole system.
    - Multiple personnel owns multiple components.
      - If one leaves the company, it is easier to replace…
    - Documented designs can be used by new personnel to minimize the learning curve.



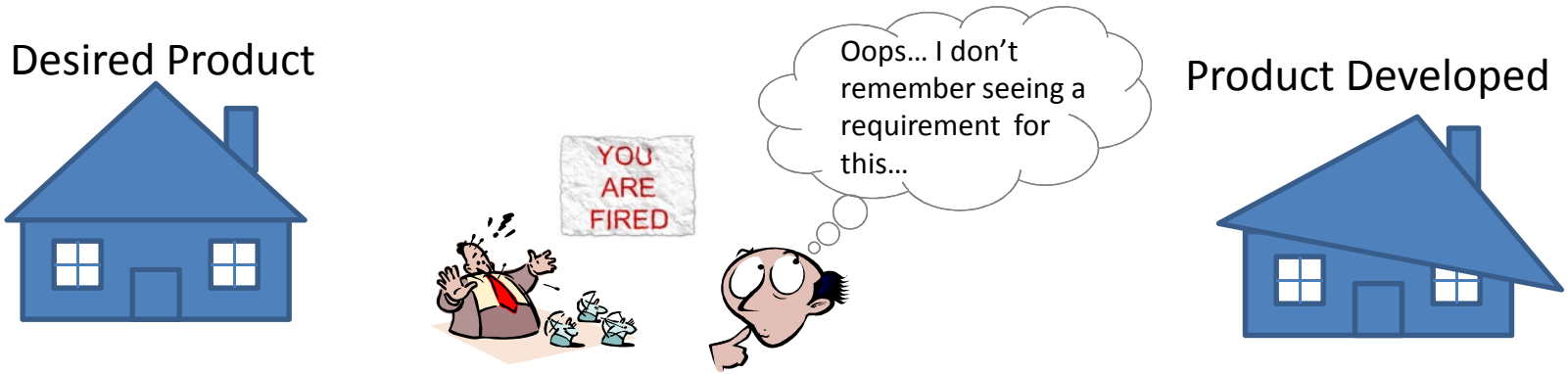
- Overall, design helps improve project management.
  - ✓ Planning, organization, staffing, and tracking

SOFTWARE DESIGN CHALLENGES

- Today, the software design phase has evolved from an ad-hoc and sometimes overlooked phase to an essential phase of the development life-cycle.
- The increasing complexity of today's software systems has created a set of particular challenges that makes it hard for software engineers to meet the continuous customer demand for higher software quality.
- These challenges have prompted software engineers to pay closer attention to the design process:
  - ✓ To better understand, apply, and promulgate well-known design principles, processes, and professional practices.
- The major design challenges include:
  - ✓ Requirements volatility
  - ✓ Inconsistent development processes
  - ✓ Fast, and ever-changing technology
  - ✓ Ethical and professional practices
  - ✓ Managing design influences
- To understand these better, we need to explore them in more depth…

Software Design Challenge #1 – Requirements Volatility

- Requirements' volatility is a major reason for the complexity of software projects.
  - ✓ The common view is that software can be modified "easily"
    - Requirements are not fully specified before design and construction
    - Requirements are changed after design and construction
  - ✓ Consider if we take this approach for the development of a house!



SOFTWARE DESIGN CHALLENGE #1 – REQUIREMENTS VOLATILITY

- In software projects, although much effort is put into the requirements phase to ensure that requirements are complete and consistent, that is rarely the case.
  - ✓ This makes the software design phase the most influential one when it comes to minimizing the effects of new or changing requirements.
  - ✓ This forces designers to create designs that provide solutions to problems at a given state while also anticipating changes and accommodating them with minimal effort.

- Because of requirements' volatility, software engineers must have a strong understanding of the principles of software design and develop skills to manage complexity and change in software projects.

SOFTWARE DESIGN CHALLENGE #2 – THE PROCESS

- Software engineering is a process-oriented field. In the design phase, processes involve a set of activities and tasks required to bridge the gap between requirements and construction. For example:
  - ✓ Architectural and detailed designs
  - ✓ Design reviews
  - ✓ Establishing quality evaluation criteria
  - ✓ Establishing design change management and version control
  - ✓ Adopting design tools
  - ✓ …
- The problem is that in many cases, a company's design process
  - ✓ is not well established,
  - ✓ is poorly understood,
  - ✓ is approached with minimalistic expectations,
  - ✓ is focused on one form of design, e.g., user interface, while ignoring others, or,
  - ✓ is simply, not done at all!

SOFTWARE DESIGN CHALLENGE #3 – THE TECHNOLOGY

- The technology for designing and implementing today' software systems continues to evolve to provide improved capabilities. For example,
  - ✓ Modeling languages and tools
  - ✓ Programming languages
  - ✓ Integrated development environments
  - ✓ Design strategies, patterns, etc.
  - ✓ …
- As new technologies emerge, software designers are required to assimilate and employ them all at the same time.
  - ✓ In some cases, old and new technology needs to coexist in the same project!
- This creates a demand for capable designers that can assimilate new concepts and technology quickly and effectively.
  - ✓ This is challenging because of the time required for both learning new technology and completing a project on-time, while making sure that the new technology interoperates well with old legacy systems.

SOFTWARE DESIGN CHALLENGE #4 – ETHICAL AND PROFESSIONAL PRACTICES

- Designers create blueprints that drive the construction of software.
  - ✓ Tight schedules can create external pressures to deviate from the formal design process to get the product out the door.
  - ✓ In some cases, this can have catastrophic consequences
- To correctly and responsibly execute the design phase, designers are required to exert strong leadership skills to:
  - ✓ Influence and negotiate with stakeholders
  - ✓ Motivate the development team
  - ✓ Enforcing ethical guidelines
  - ✓ Evaluate the social impacts of their designs in the public domain or in safety-critical systems
  - ✓ Follow and enforce the ethical and professional practices
- This is a challenge when attempting under numerous pressures from different stakeholders, e.g., management, customer, peers, etc.

SOFTWARE DESIGN CHALLENGE #5 – MANAGING DESIGN INFLUENCES

- Besides negative pressures, designs are shaped by other influences from stakeholders, the development organization, and other factors (e.g., the designers' own experiences).

- These influences can have cyclical effects between the system and its external influences, such that external factors affect the development of the system and the system affects its external factors [1].

- Managing these influences is essential for maximizing the quality of systems and their related influence on future business opportunities.

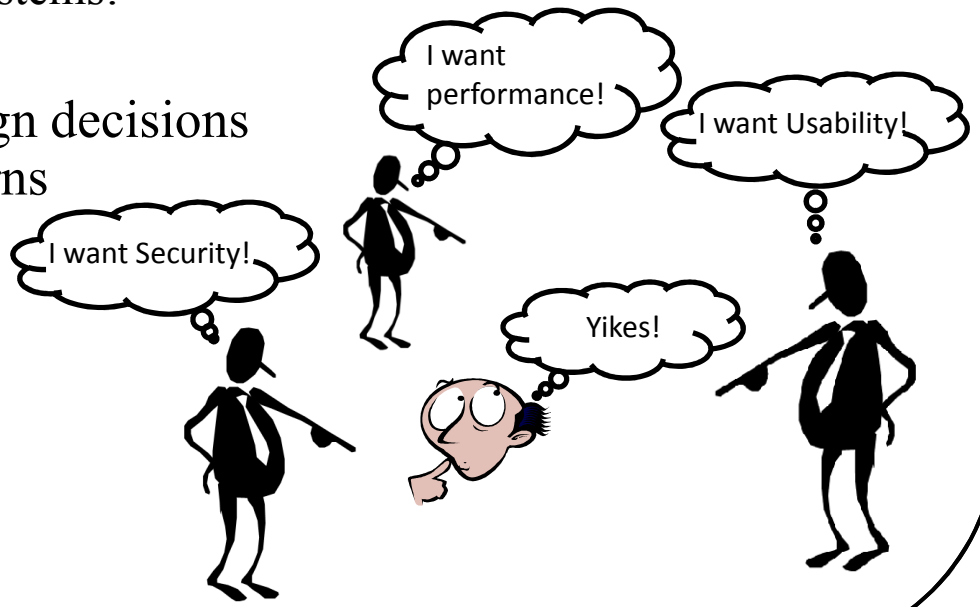
- Of specific importance are design influences that come from
  - ✓ the system's stakeholders and,
  - ✓ its developing organization

SOFTWARE DESIGN CHALLENGE #5 – MANAGING DESIGN INFLUENCES

- Software projects can have a multitude of stakeholders, each with specific wants and needs that influence the software design.
  - ✓ Some conflicting with each other!
  - ✓ Each stakeholder believes he/she is correct.

- This requires some design trade-offs to satisfy each customer.
  - ✓ Difficult to do in large-scale systems!

- This is difficult to do since design decisions need to accommodate all concerns without negatively affecting the project.



Software Design Challenge #5 – Managing Design Influences

- The developing organization also influences designs significantly.
  - ✓ Consider software development across site boundaries!
    - Consider coordinating design efforts
    - Consider conducting peer reviews
    - Consider developing code from design
    - Consider managing version control
  - ✓ In this case, design must support such development!

- At the same time, designs can influence the developing organization to enter new areas of business.
  - ✓ Consider a design that supports plug-ins to enhance capabilities of the software. Someone may realize that existing functionality can be enhanced via plug-in to solve a problem in a different domain, giving the developing organization a competitive advantage!

- Managing these influences is challenging because it requires designers to span out of the technical domain to have keen interest in the organization as a whole.

SOFTWARE DESIGN PROCESS

- Up until now, we have spent quite a bit of time discussing properties of software design. We have also defined software design from the *process perspective*, which referred to the phase, activities, tasks, and interrelationship between them required to model software's structure and behavior before construction.
  - ✓ However, we have not formally described what goes on within the design phase.
  - ✓ Let's examine in depth the software design process…

- A software design process is a set of activities and controls that specify how resources work together for the production of software design artifacts.
  - ✓ Two major activities of the software design process include:
    - Software Architecture
    - Detailed Design

SOFTWARE DESIGN PROCESS

- Software Architecture
  - ✓ Corresponds to a macro design approach for creating models that depict the quality and function of the software system.
  - ✓ Provides black-box models used to evaluate the system's projected capabilities as well as its expected quality.
  - ✓ Designed using multiple perspectives, therefore, allows different stakeholders with different backgrounds to evaluate the design to ensure that it addresses their concerns.
  - ✓ It provides the major structural components and interfaces of the system.
  - ✓ It focuses on the quality aspects of the system before detailed design or construction can begin.
  - ✓ They serve as important communication, reasoning, and analysis tool that supports the development and growth of the system [1]
  - ✓ Lays the foundation for all subsequent work in the software engineering life-cycle.

SOFTWARE DESIGN PROCESS

- Detailed Design
  - ✓ Whereas software architecture deals with the major structural components and interfaces of the system, detailed design focuses mostly on the internals of those components and interfaces.
  - ✓ Begins after the software architecture activity is specified, reviewed, and deemed *sufficiently* complete.
  - ✓ Builds on the software architecture to provide a *white-box* approach to design the structure and behavior of the system.
  - ✓ Refines the architecture to reach a point where the software design, including architecture and detailed design, is deemed sufficiently complete for the construction phase to begin.
  - ✓ Focuses on functional requirements, whereas the architecture focuses mostly on non-functional, or quality, requirements.
  - ✓ Two important tasks of the detailed design activity include:
    - Interface Design
    - Component Design

SOFTWARE DESIGN PROCESS

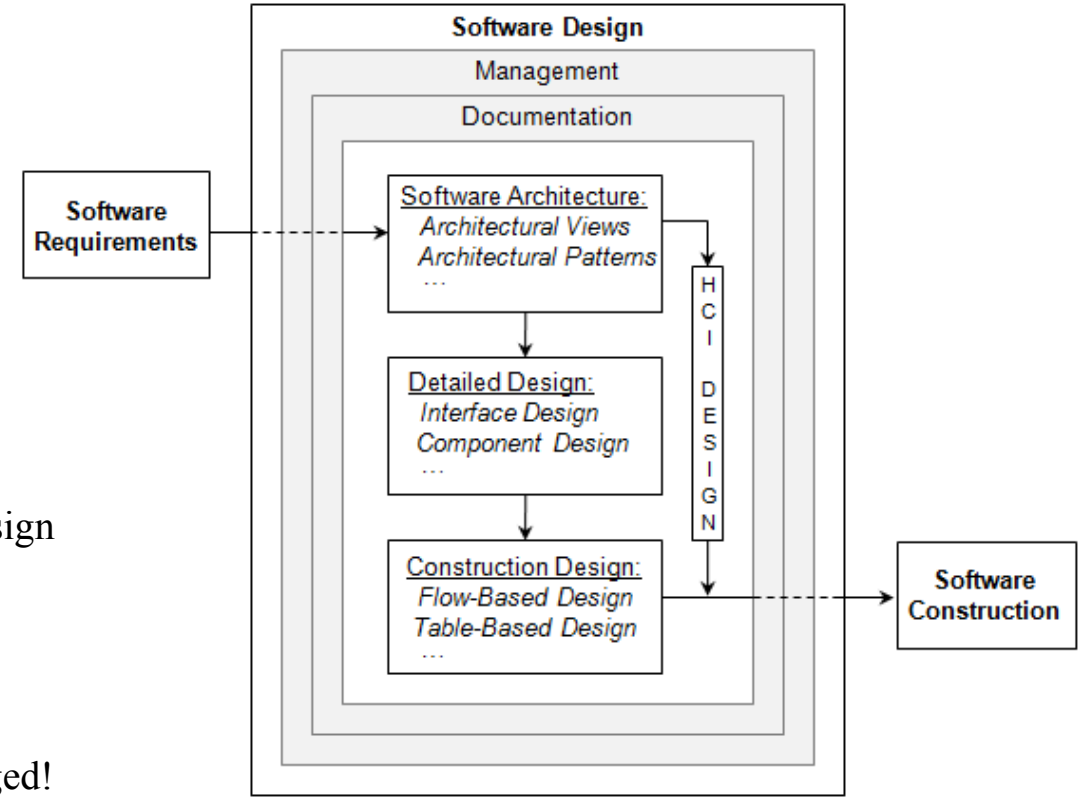
- Interface design
  - ✓ Refers to the design activity that deals with specification of interfaces between components in the design.
  - ✓ Provide a standardized way for accessing services provided by software components.
  - ✓ Allow for multiple efforts to occur in parallel, as long as interfaces are obeyed, therefore, it is one of the first tasks during detailed design.
  - ✓ Can be performed for both internal interfaces and external interfaces, e.g., XML messaging specification for communication across the network.
- Component design
  - ✓ During architecture, major components are identified. During component design, the internal design of (the structure and behavior of) these components is created.
  - ✓ In object-oriented systems, using UML, component designs are typically in the form of class diagrams, sequence diagrams, etc.
  - ✓ When creating these designs, several design principles, heuristics, and patterns are often used in professional practice.
  - ✓ Sometimes referred to as component-level design.

SOFTWARE DESIGN PROCESS

- Software architecture and detailed design are well-known activities of the software design phase, however, other important activities are required for supporting the creation of architectural and detailed designs.
  - ✓ Therefore, when scoping the software design process, the effort required for these other activities must be considered!
- Other important activities that require time and effort include:
  - ✓ Design documentation
  - ✓ Management activities
- An often neglected activity when scoping design efforts is the design that occurs during construction. This form of design is important to model complex functions or routines identified during detailed design.
  - ✓ Construction design is a form of detailed design that occurs mostly during the construction phase.
  - ✓ Construction design is important to reason, communicate, and document the solution to complex programming problems.
- Another important design activity is the Human-computer interface design.
  - ✓ Design activity where general principles are applied to optimize the interface between humans and computers.

SOFTWARE DESIGN PROCESS

- The software design phase has several activities, each having one or more tasks.
  - ✓ Architecture
    - Identifying views
    - Identifying patterns
    - …
  - ✓ Detailed Design
    - Interface design
    - Component design
    - …
  - ✓ Construction Design
    - Flow-based design
    - Table-based design
    - …
  - ✓ Human-computer Interface Design
    - …



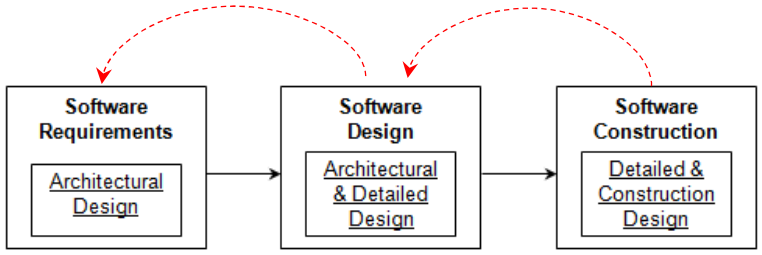
- Each design activity needs to be
  - ✓ *Documented*, and
  - ✓ All processes need to be managed!

- **We will spend the rest of the semester** learning how to carry out these activities (including tasks), how to document these efforts, and how to manage them!

SOFTWARE DESIGN PROCESS

- In the previous slide, it is seen that architectural designs are elaborated through detailed design, which are further elaborated through construction design. In practice, design activities are not carried out in such orderly and controlled fashion.
  - ✓ Quite a bit of iteration occurs between these activities.

- In a perfect world, all design work would take place during the design phase. In practice, the distribution of design activities varies throughout the software development life-cycle.
  - ✓ In most projects, most of the design efforts occur during the design phase, but some design inevitably occurs in other life-cycle phases, e.g.,
    - Requirements
    - Construction
  - ✓ In some cases, architectural design efforts occur during the requirement's phase.
  - ✓ In some cases, detailed design occurs during the construction phase.
  - ✓ These cases will become evident as we cover these activities during the course.



WHAT'S NEXT…

- In the next session, we will continue the focused discussions on software engineering design. Specifically,
  - ✓ Roles of software designers
    - Systems Engineer
    - Software architect
    - Component designer
    - User interface designer
  - ✓ Design principles
    - Modularization
    - Abstraction
    - Encapsulation
    - Cohesion and coupling
    - Separation of interface and implementation
    - Sufficiency and completeness
  - ✓ Design strategies
    - Object-oriented vs. structured design
  - ✓ Practical design considerations

REFERENCES

- [1] Bass, Len, Paul Clements, and Rick Kazman. Software Architecture in Practice, 2d ed. Boston: Addison-Wesley, 2003.