Introduction to Large-Scale Software Design

17-423/723 Designing Large-Scale Software Systems

Lecture 1 Jan13, 2025

Today's Agenda

- Introduction and motivation for software design
- Course logistics

Learning Goals

After today's lecture, you will be able to:

- Describe what software design is and why it is important
- Describe the risk-driven approach to software design
- Describe the generate-communicate-evaluate (GCE) paradigm to designing software
- Describe the course logistics

Why design?

What is Design?

Dictionary

Definition

verb

noun

Synonyms Synonym Chooser Example Sentences Word History Phrases Containing Entries Near Show More ~

design 1 of 2 verb

de∙sign (di-'zīn ◄»)

designed; designing; designs

Synonyms of *design* >

transitive verb

- to create, fashion, execute, or construct according to plan : DEVISE, CONTRIVE
 design a system for tracking inventory
- 2 a : to conceive and plan out in the mind
 - he designed the perfect crime
- **b** : to have as a purpose : INTEND
 - she *designed* to excel in her studies
- c : to devise for a specific function or end a book *designed* primarily as a college textbook
 - a suitcase *designed* to hold a laptop computer

What is Design?

- (Verb.) To follow a process for planning how a product will work before it is implemented
- (Noun.) The result of a design activity, such as design documents, sketches, diagrams, which may serve as a plan for implementation



Design in Engineering Products



Software Design in Practice

- Systematic and deliberate design is less common in software industry
- Temptation is to start coding right away
 - "Why bother with design? It's fine as long as it works!"
- Even if you skip deliberate design, you will end up with an implicit or "default" design anyway
 - But most likely, the default design is not going to have desirable qualities of a successful product
 - As a product/system grows more complex, the cost of improving the design will also become more expensive
 - No deliberate design => higher development costs in the long term!

Discovering Issues Early in Design



US National Institute of Standards and Technology (NIST)

Consequences of Poor or "Default" Design



- Poor or lack of design can cause loss of customers, product failures, injuries/deaths, property/environmental damage, or financial loss
- Throughout the class, we will use case studies of software failures (as well as successes) to extract lessons and design principles

Risk-Driven Approach to Design

Reason #1: Design is less well-understood by software engineers

- Design is often considered "art" or "talent" rather than teachable skills
- This course is intended to change this perspective! We will teach you design principles and methods that will make you a better designer



Reason #2: Requirements change frequently

- Your design may become obsolete over time
- But you **can** design systems to be ready for such changes
- "Designing for change" will be a major theme throughout the class



"The client kept changing the requirements on a daily basis, so we decided to freeze them until the next release."

Reason #3: Tension between deliberate design vs. time-to-market

- Pressure to release your product as soon as possible
- Often used as an excuse to avoid any deliberate design



Reason #3: Tension between deliberate design vs. time-to-market

- Pressure to release your product as soon as possible
- Often used as an excuse to avoid any deliberate design
- But as the system scales, it must eventually deal with poor/lack of design!



Risk-Driven Approach to Design

- What is the "right" amount of design to do?
- Risk = (cost of failure)*(likelihood of failure)
- **Ask**: What are possible costs to my organization/stakeholders if my product/system fails?









Risk-Driven Approach to Design

- What is the "right" amount of design to do?
- Risk = (cost of failure)*(likelihood of failure)
- **Ask**: What are possible costs if my product/system fails?
 - The amount of design should be proportional to the level of risks
- Cost of failure: Examples
 - Loss of customers & revenue due to poor availability
 - Loss of customers & revenue due to poor usability
 - Extra development costs and project delays due to poor extensibility
 - Theft of sensitive information due to poor security
 - Injuries or loss of lives due to poor safety
- The goal is not to achieve a perfect product, but to <u>identify & eliminate</u> <u>"bad" designs that are likely to result in high-risk failures</u>

What does designing involve?

Types of Design

- Conceptual design
 - What are the key concepts in our system? What does the data model look like?
- Functional design
 - How are the key functionalities of the system implemented? What does the business logic look like?
- Architectural design
 - What are the key components in the system? What are the interfaces between components look like?
- Interaction design
 - How does the user interact with the system? How do we ensure that the system is easy and intuitive for them to use?
- Performance design
 - How do we scale the number of users? How do we ensure high service uptime?
- Security and reliability design
 - How do we protect the system against malicious actors? How do we make system resilient against possible failures in the network?

Quality Attributes

- Design is "easy" if the only goal is to build a functional system
- Quality attributes (QAs) are what makes design challenging
 - Often these are cross-cutting and in conflict with each other
 - Designing will frequently involve making trade-offs among these quality attributes
- QAs that we will study in this class include:
 - Extensibility
 - Reusability
 - Interoperability
 - Testability
 - Scalability
 - Robustness
 - Security
 - Usability
 - and others...





SOFTWARE DESIGN DECODED

66 Ways Experts Think

Marian Petre and André van der Hoek Illustrations by Yen Quach

• Generate, brainstorm, and explore a space of candidate design solutions



- Generate
- Communicate designs to team members & clients through design sketches, documentation & prototyping



- Generate
- Communicate
- Evaluate designs for various quality attributes & identify possible flaws



Generate-Communicate-Evaluate (GCE) Paradigm



We will cover a set of principles, techniques, and tools for generating, communicating, and evaluating designs w.r.t. various quality attributes

Generate-Communicate-Evaluate (GCE) Paradigm



Design is never "finished"; it's a continuous, iterative process!

- Develop skills for a career as a system architect/designer
 - "System design" questions are common in interviews for software engineering positions

- Develop skills for a career as a system architect/designer
 - "System design" questions are common in interviews for software engineering positions
- Role of AI/LLMs in software development
 - Coding will be more and more automated

ChatGPT Will Replace Programmers Within 10 Years

Predicting The End of Manmade Software



Adam Hughes · Follow Published in Level Up Coding · 12 min read · Feb 28



- Develop skills for a career as a system architect/designer
 - "System design" questions are common in interviews for software engineering positions
- Role of AI/LLMs in software development
 - Coding will be more and more automated
 - Role of software engineers will likely change & involve more high-level design
 - Design skills and knowledge will become increasingly more valuable

- Develop skills for a career as a system architect/designer
 - "System design" questions are common in interviews for software engineering positions
- Role of AI/LLMs in software development
 - Coding will be more and more automated
 - Role of software engineers will likely change & involve more high-level design
 - Design skills and knowledge will become increasingly more valuable
- For PhD & research-oriented students:
 - Software design is still relatively less understood by researchers
 - This course will discuss some open questions and opportunities in software design research

What do we mean by "large-scale"?

- Not just in scalability in the number of users & requests, but also in terms of:
 - Complexity in the problem domain
 - Diversity in functional requirements and quality attributes
 - Number of interacting subsystems and components
 - Period of system evolution over time (potentially indefinite)

Course Logistics

Course Staff



Instructor Eunsuk Kang eunsukk@andrew



TA Shivam Bansal shivamb@andrew

Office hours TBD; Watch out for announcement after class!

Communication

- Email us or ping us on Slack (invite link on Canvas)
- Post questions on Slack
- All announcements through Slack #announcements and Canvas
- Submissions through Canvas & Gradescope
- Other non-public materials (readings) on Canvas
- Please use #questions or #assignments and post publicly if possible; your classmates will benefit from your Q&A!

Disclaimer: Relatively New Class (Year 2)

- Expect rough edges & moving parts over the semester
- Please be flexible & patient!
- We welcome your candid feedback! Let us know:
 - Topics that you'd like to us cover in more/less detail
 - Concepts that could be better explained
 - Tools/techniques that you'd like us to cover
 - An assignment that takes too much (little) time or is too hard (easy)
 - Readings that are too boring/obscure/hard to follow
 - And anything else, really!
- You have a chance to shape the future offerings of this course!

Course Learning Goals

After taking this course, you will be able to:

- Design software systems for various quality attributes, including reusability, extensibility, interoperability, robustness, scalability, testability, security, usability
- Explain how to adapt a software design process to fit different domains, such as robotics, web apps, mobile apps, and medical systems
- Identify, describe, and prioritize relevant requirements for a given design problem
- Generate viable design solutions
- Apply appropriate abstractions & modeling techniques to communicate and document design solutions
- Evaluate design solutions based on their satisfaction of common design principles and trade-offs between quality attributes

Course Philosophy

- Hands-on Experience in a collaborative project
- Growth mindset & learning from failures
 - Learning from real-world case studies
 - Learning from your own mistakes in the project
- Active student participation
 - We encourage you to ask questions and participate in class discussions
 - Wrong answers support learning! (see growth mindset)

This is a Software Engineering Class!

- Focused on engineering judgment
- Arguments, trade-offs, and justification, rather than a single correct answer
- The answer will often be: "It depends..."
- Practical engagement, building systems, testing, automation
- Strong teamwork component

Pedagogical Principles in this Course

- Spaced Practice / Interleaving of Topics
 - Practicing newly learned material is most effective when you started to forget about it
 - Based on educational research, topics won't be covered in a single lecture, but spread throughout the course
 - Connections between different topics will become more clear this way!
- Active Learning
 - Lectures are structured with many in-class discussions, think-pair-share, and other activities
 - Research shows that active learning is more effective, when when it doesn't feel like that See <u>https://www.pnas.org/doi/10.1073/pnas.1821936116</u>

Recitations

- Additional exercises to supplement concepts from the lectures
- Cover hands-on topics, such as tools, frameworks, best practices,
- First recitation this Friday!
- Participation in recitations will also be considered as part of "Class Participation" grade (see later on Grading)

Grade Breakdown

- 20% Homeworks
- 50% Project
- 20% Midterm & final
- 10% Class participation
 - $\circ \quad \text{In-class discussions} \\$
 - Participation in recitation activities
 - Exit tickets

Participation: Exit Tickets

- Goal: Recall and summarize the key ideas from the lecture
 - Also to help us understand how well we conveyed those ideas
- In the last 5~10 min of each class, we will ask you to answer a couple of quick questions about that day's content (on Canvas)
- Not graded on correctness; any on-topic & "reasonable" answers will be accepted
- 3 free passes throughout the semester
 - Let us know if you have any exceptional circumstances (illness, travel, interviews, etc.,)

Team Project

- Goal: Gain experiences designing, implementing, and iteratively improving a complex software system
- Six milestones over the semester
 - M1: Initial design and specification
 - M2: Initial prototype implementation
 - M3: Iterative design for changeability & interoperability
 - M4: System integration
 - M5: Iterative design for robustness
 - M6: Final design report & presentation
- More details about the project next lecture

Individual Homeworks

- Goal: Practice applying design principles and techniques not covered by the project milestones
- We expect that these will take no more than 2~3 hours to complete

Grading: Tokens

- Individual & team tokens: 7 each
- Use 1 token to submit a homework assignment/project milestone 1 day late
- Use 3 tokens to redo a homework/milestone
- Unused individual tokens at the end of the semester will count towards your participation grade

Participation: In-class Discussions

- Most lectures will involve case studies & discussions
- Please don't hesitate to contribute your ideas and experiences! Remember, there's no one "correct" answer to problems in this class

Use of Generative AI

- You are free to use generative AI (e.g., ChatGPT) for homeworks/project
 - We are interested to explore the potential utility of LLMs for software design!
 - Generate design alternatives, generate documentation, synthesize code given a specification, etc.,
- But you are **<u>NOT</u>** allowed to use it for **exams** or **exit tickets**
- It's your responsible to check the quality of the output from an LLM
 - These models will sometimes hallucinate and generate superficial, bogus output
- In your submissions, clearly document how you've used these tools

Looking Ahead

Next 2~3 weeks: Foundational techniques and tools for design

Domain & design modeling, quality attributes & trade-offs, generating design ideas, design review, design processes

Second half of the course: Designing for quality attributes

Design for change, interoperability, reuse, scalability, robustness, security, AI,...

Slack Introductions

- Before Friday's recitation, introduce yourself on #social channel:
 - Your (preferred) name
 - In 1~2 sentences, your software engineering background and goals (e.g., coursework, internships, work experience)
 - Your favorite programming languages, tools, or frameworks
 - One topic you are particularly interested in learning during this course?
 - A hobby or a favorite activity outside school

Exit ticket!

• See Canvas

Summary

- Systematic design is common in many engineering disciplines.
- Decisions to invest in design should be driven by the amount of risks in the product or system.
- Designers generate, communicate, and evaluate design solutions.
- Designing is a continuous, iterative process.