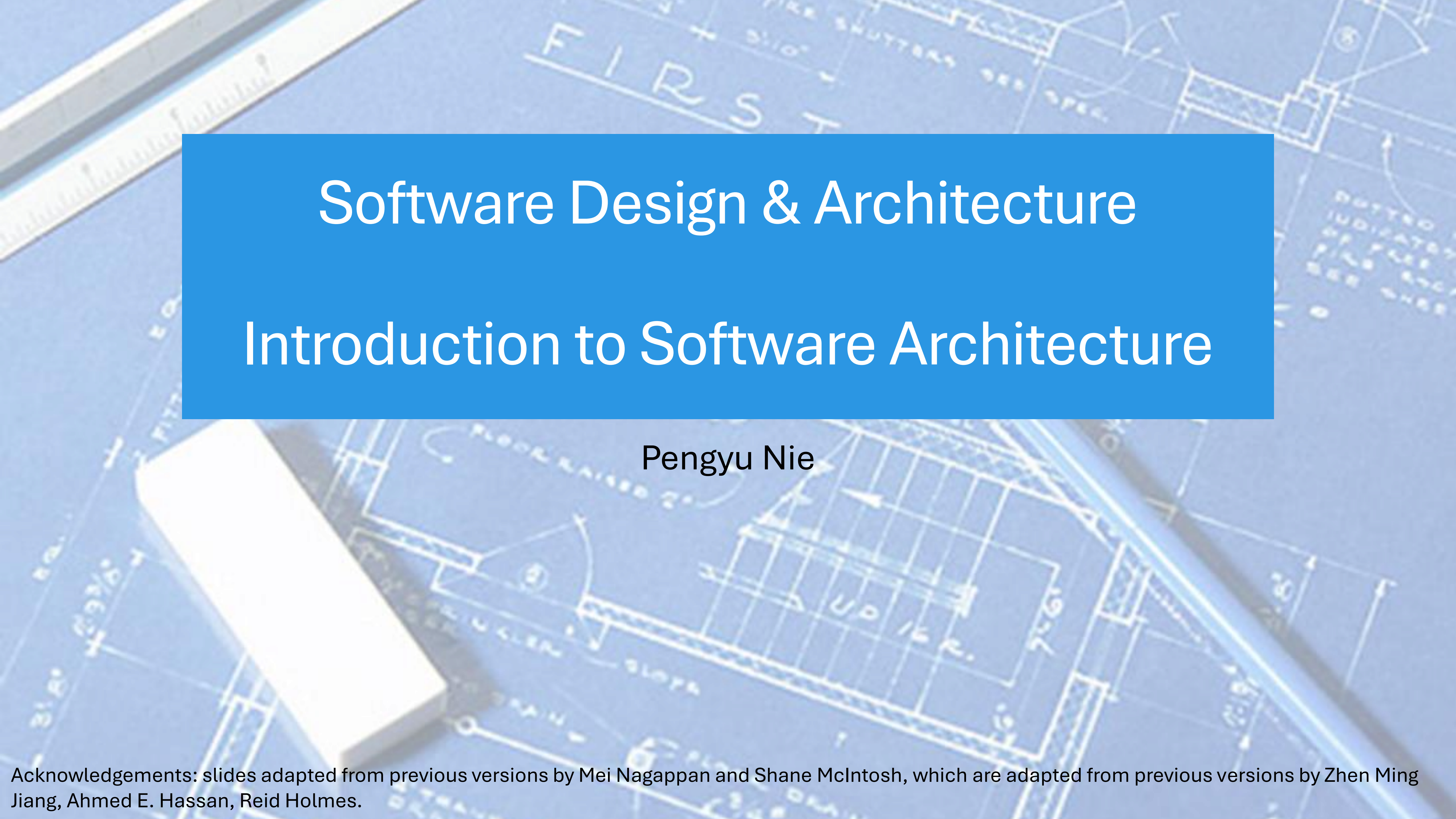


Software Design & Architecture

Introduction to Software Architecture

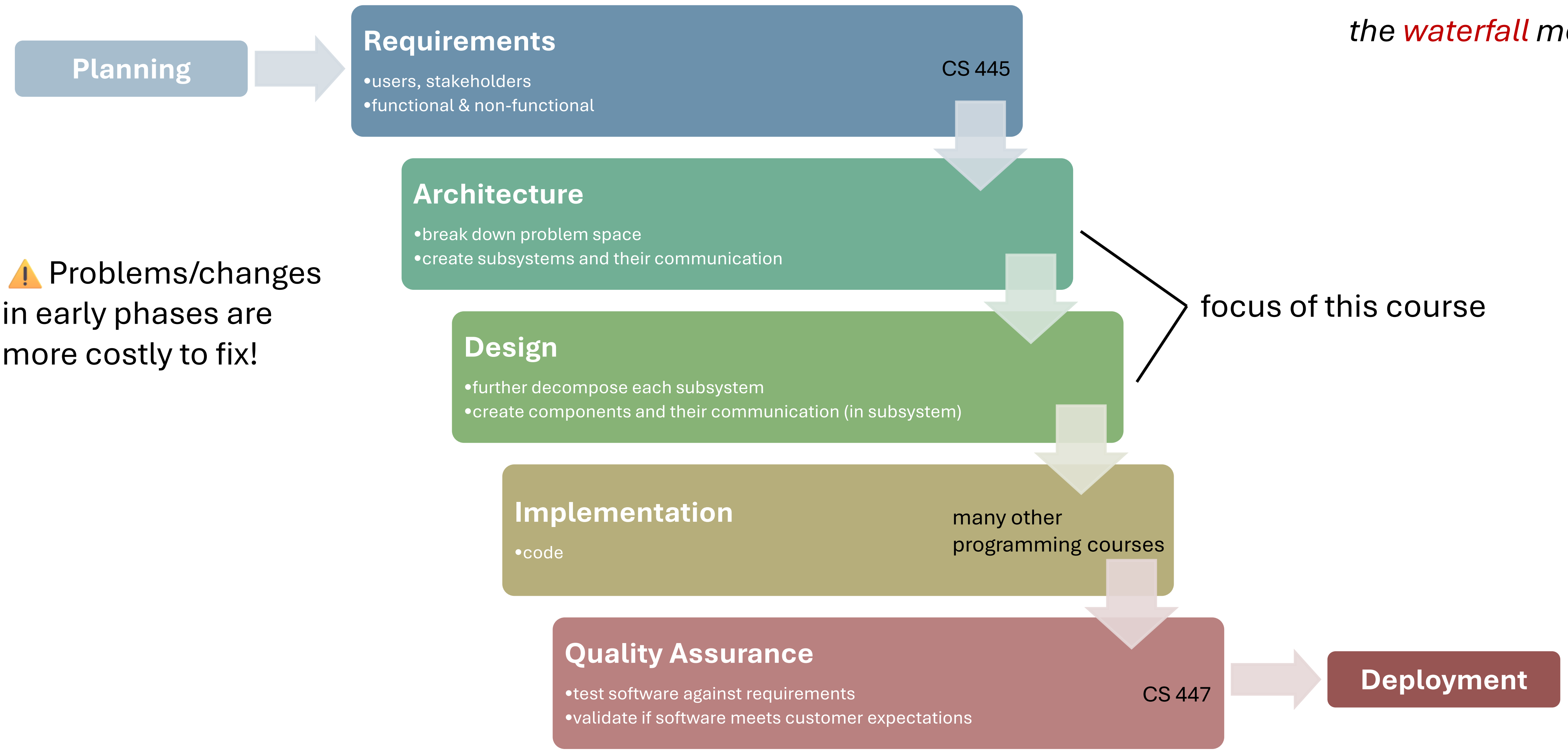
Pengyu Nie

Acknowledgements: slides adapted from previous versions by Mei Nagappan and Shane McIntosh, which are adapted from previous versions by Zhen Ming Jiang, Ahmed E. Hassan, Reid Holmes.

Agenda

- Software development lifecycle
- Software architecture: what and why
- Exercise: sketching software architecture
- Review P0 and P1 requirements

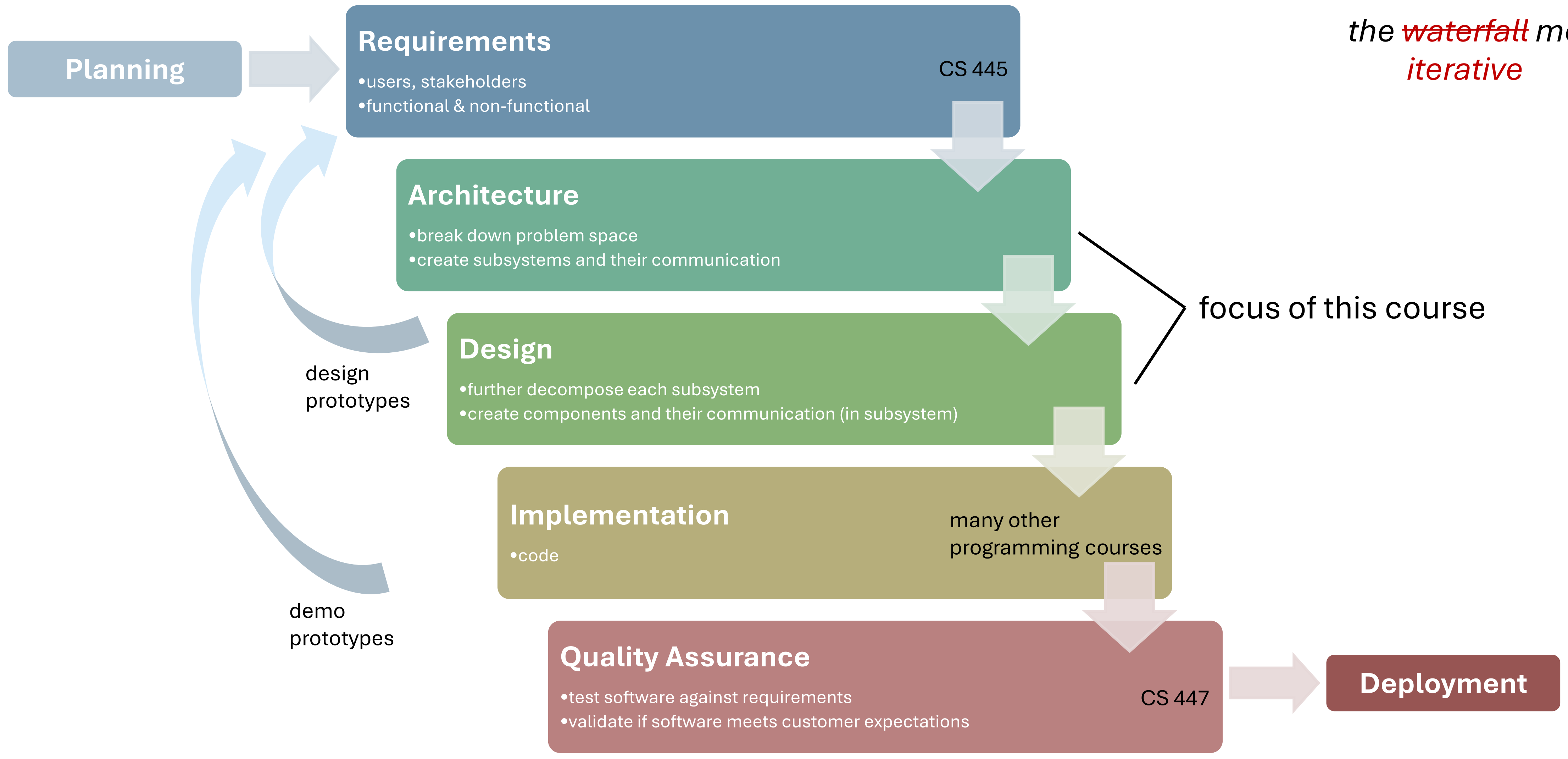
Software Development Lifecycle (SDLC) Phases


Planning
Requirements
•users, stakeholders
•functional & non-functional
CS 445
Architecture
•break down problem space
•create subsystems and their communication
Design
•further decompose each subsystem
•create components and their communication (in subsystem)
Implementation
•code
many other programming courses
Quality Assurance
•test software against requirements
•validate if software meets customer expectations
CS 447
Deployment
focus of this course

the waterfall model

⚠️ Problems/changes in early phases are more costly to fix!

Agile Software Development


Planning
Requirements
•users, stakeholders
•functional & non-functional
CS 445
Architecture
•break down problem space
•create subsystems and their communication
Design
•further decompose each subsystem
•create components and their communication (in subsystem)
Implementation
•code
many other programming courses
Quality Assurance
•test software against requirements
•validate if software meets customer expectations
CS 447
Deployment
design prototypes
demo prototypes
focus of this course

the ~~waterfall~~ model
iterative

Read more about SDLC and Agile in CS 346 notes: https://student.cs.uwaterloo.ca/~cs346/1251/course-notes/project-management/agile.html

What is Architecture?

“both the process and the product of planning, designing, and constructing buildings or any other structures”

-- Encyclopedia Britannica

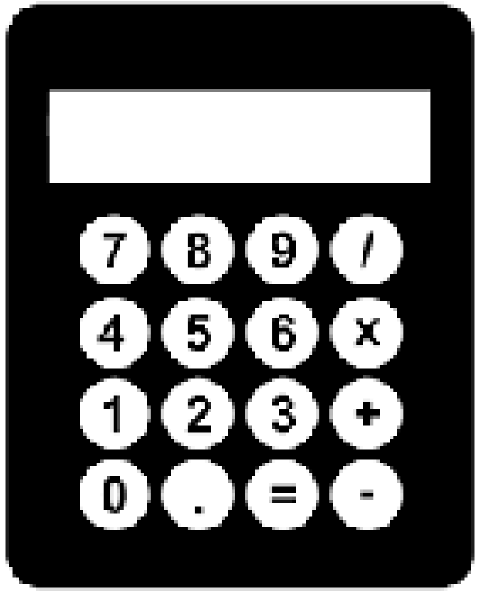
The Three Original Principles

- Durability: a structure should stand up robustly and remain in good condition
- Utility: a structure should be suitable for the purposes for which it is used
- Beauty: a structure should be aesthetically pleasing

-- *De Architectura* by Roman architect Vitruvius (1st century AD)

Why do we Need Architecture?

Why do we Need Architecture? (Software ver.)



The Architect

- Distinctive role
- Broadly trained
  - Requirements, design, implementation, use
- Has a keen sense of aesthetics
- Strong understanding of the domain

- What do these domain skills look like for buildings? For software?

Benefits of Architect

- What common benefits can software gain from an architect, that a building also gets from its architect?
  - Intellectual control and conceptual integrity
  - Experience
  - Management

Analogy to Building Architecture

- Architects focus on clients’ needs
- Iteration on a set of blueprints, refining when necessary
  - Intermediate plans, mockups, prototypes
- Created by specialists, not end users
- Structure induces properties (e.g., in a castle)
- Architects require broad training
  - Leverage lessons from past generations

Differences from Building Architecture

- What are the key differences between software architecture and architecture for buildings?
  - Age
  - Material
  - Delivery machanisms

Shortcomings of Analogy

- We have much more experience with buildings
- Buildings are physical artifacts; software is intangible
- Expertise in the software industry is less clearly differentiated (e.g., no “exception specialists”)
- Anyone can write software
- Deployment and operations are very different
- Changes are expected

Architecture

- Architecture is:
  - All about communication
  - What “parts” are there
  - How do the “parts” fit together
- Architecture is not:
  - About development
  - About algorithms
  - About data structures

What is Software Architecture?

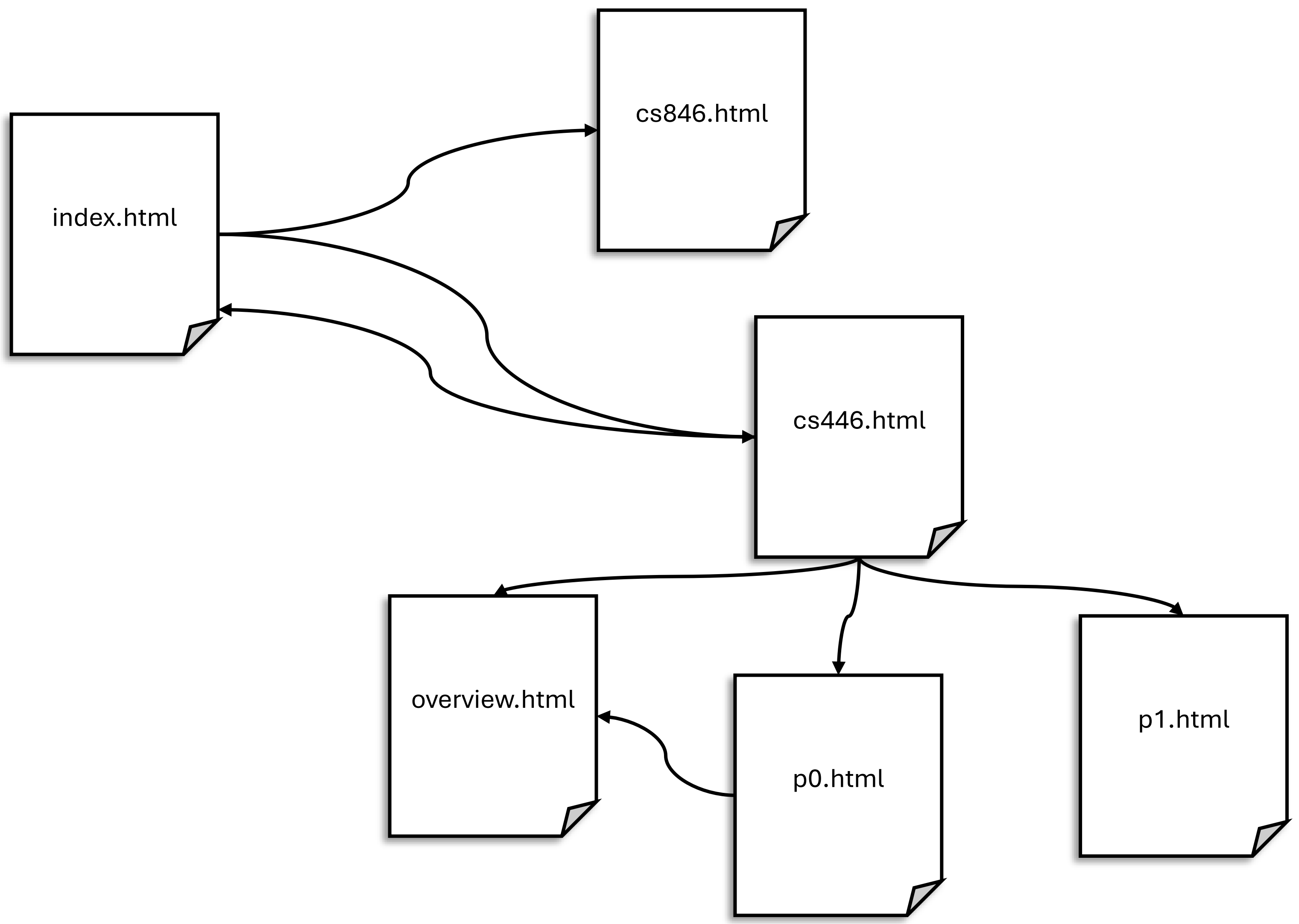
- The conceptual fabric that defines a system
  - All architecture is design but not all design is architecture
- Architectures capture three primary dimensions:
  - Structure
  - Communication
  - Non-functional requirements

What is Software Architecture? (Formal ver.)

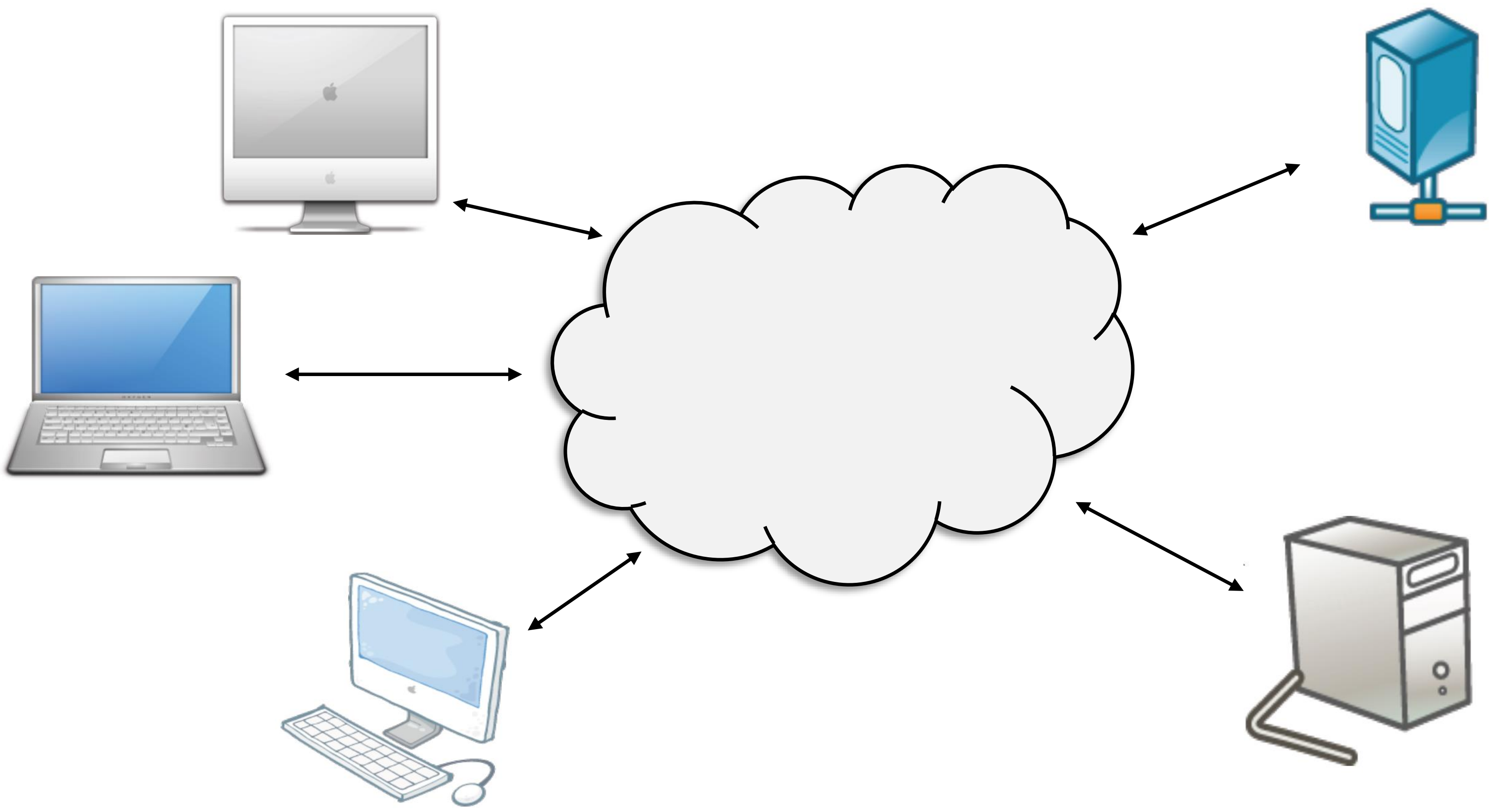
- “Architecture is the fundamental organization of a system, embodied in its [subsystems], their relationships to each other and the environment, and the principles govering its design and evolution”

-- ANSI/IEEE 1471-200

Logical Web Architecture



Physical Web Architecture



Exercise: Architectural Sketching

- Have your favourite draing tool launched
  - Microsoft whiteboard https://whiteboard.office.com
  - draw.io https://app.diagrams.net/
  - Mermaid (in plain text) https://mermaid.live/edit
- Target application: web browser (e.g., Chrome, Firefox)
- Task 1: List as many subsystems as you can think of.
  Use boxes to denote subsystems.

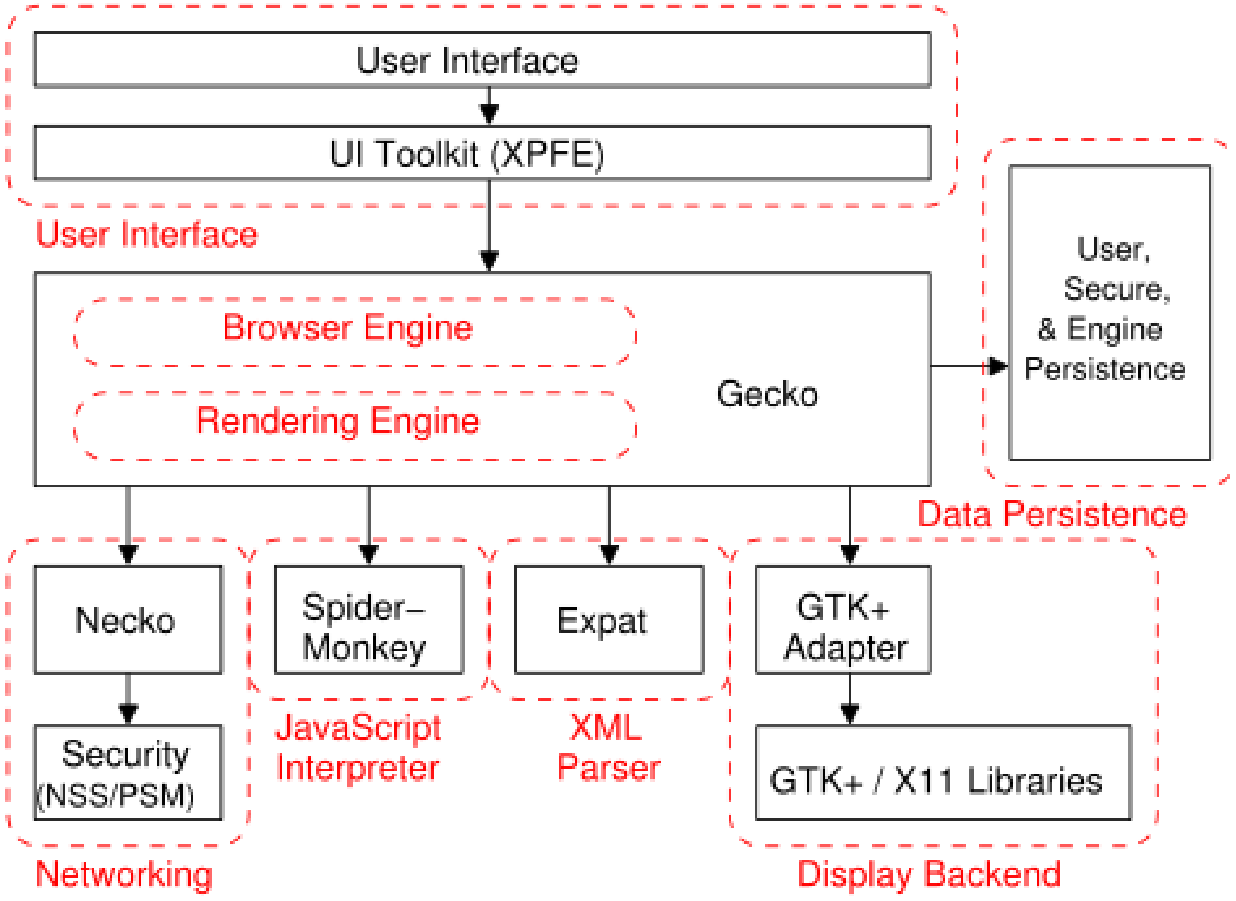
Instructor’s list of subsystems

- UI layer (to support multiple platforms)
- HTML/DOM engine
- CSS processor
- JS engine (to process client-side scripts)
- Networking (to enable “talking” to web servers)
- Bookmark manager
- Secure persistence (e.g., passwords, credit cards)
- History database
- Plugin manager

Exercise: Architectural Sketching (cont.)

- Continue the drawing from task 1
- Task 2: Connect subsystems that need to communicate.
  Use directed arrows to indicate control/data flow.

The Anatomy of Web Browsers



http://plg.math.uwaterloo.ca/~migod/papers/2007/emse-browserRefArch.pdf

Why is Software Architecture Important

- “Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.”

  -- Eoin Woods

- Architecture focuses on those aspects of a system that would be difficult to change once the system is built

Why is Software Architecture Difficult?

- “The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark.”

  -- Philippe Krutchen

- Young field
- High user expectations
- Software cannot execute independently

Specific Difficulties

- Complexity: grows non-linearly with program size
- Conformity: system is dependent on its environment
- Changeability: perception that software is easily modified
- Intangibility: not constrained by physical laws

Attacks on Difficulties

- High-level languages
- Development tools & environments
- Component-based reuse
- Development strategies
  - Incremental, evoluntionary, spiral models
- Emphasis on architecture and design
  - Design-centric approach taken from outset

Agenda

- Software development lifecycle
- Software architecture: what and why
- Exercise: sketching software architecture

- Review P0 and P1 requirements
  - https://pengyunie.github.io/cs446-1251/docs/project/p0/
  - https://pengyunie.github.io/cs446-1251/docs/project/p1/