CSE 403 Wrapup

Software lifecycle

- Determines the order for tasks:
  - Requirements
  - Architecture
  - Design
  - Implementation
  - V&V: verification and validation
  - Delivery
  - Maintenance
- Goal: Perform work as early as practical
  - Costly to discover information or make changes late
  - Costly to make decisions too early
  - Costly to do tasks multiple times
- In CSE 403: iterative process

Requirements

- "What", not "how"
- Reflects user view, not developer view
- Understand the customer
  - Preferably better than they understand themselves
  - Seek transformational solutions (beware risk)
- Common technique: use case / scenario / story
- User interfaces
  - High-level concepts & metaphors
  - Low-level efficiency
- Get feedback early (example: paper prototype)

Architecture

- Divide and conquer (with simple interfaces)
- Modules for logical units of computation
  - Minimize coupling, maximize cohesion
- Draw it as a picture (maybe UML)
  - Key purpose: to communicate to others
- Interactions are part of the architecture too

Divide and conquer: Modularity, abstraction, specifications

- No one person can understand all of a realistic system
- Modularity permits focusing on just one part
- Abstraction enables ignoring detail
- Specifications and documentation formally describe behavior
- Helps to understand/fix errors
  - Or to avoid errors in the first place

Teamwork

- Dividing work
  - By module in the architecture
  - By task (PM, development, testing, …)
- Decisions
  - Get understanding and buy-in
- Communication
  - Specifications
  - Deadlines
  - Effective meetings
- Motivation, trust, and morale

Working in a team

- No one person can understand all of a realistic system
  - Break the system into pieces
  - Use modularity, abstraction, specification, documentation
- Different points of view bring value
- Work effectively with others
  - Sometimes challenging, usually worth it
- Manage your resources effectively
  - Time, people
  - Engineering is about tradeoffs
- Both technical and management contributions are critical

Process

Needed to keep your project under control:

- Specification
- Schedule (with measurable milestones)
- Source control
- Testing
- Automated build and test
  - use tools: formatters, linters, style checkers, bug finders, verification
- Bug database (and fix bugs first)

Testing & verification

- Goal: completely verify functionality
  - In practice: heuristics improve completeness
- Much cheaper than discovering errors later
- Use more tools: test input minimization
- Testing tips:
  - Be systematic
  - Test early and often
  - Tests are code too
  - Involve users
  - Can be fun!

Code reviews

- Another way to get feedback early
- Team members critique documents, code, etc.
- Greatly improves quality

- Identifies opportunities for refactoring
- Refactoring improves the design
  - Design quality has many facets, depends on task

- Don’t forget design reviews (and UI, etc.)

Design

- Design of classes: similar considerations to architecture
- Design patterns: the vocabulary of program development
  - Helps you design
  - Helps you communicate
- Don’t reinvent the wheel!

Getting it right ahead of time

- Design: predicting implications
- Example: understanding interconnections
- Understanding the strengths and weaknesses
- If you don’t understand a design, you can’t use it
- Documentation matters!

Documentation

- Everyone wants good documentation when using a system
  - Not everyone likes writing documentation
- What's obvious to you probably isn't obvious to others
- Documentation is an important part of a user interface (even if the user won't read it)
- "An undocumented software system has zero commercial value." –John Chapin (CTO of Vanu, Inc.)

Maintenance/evolution

- Maintenance accounts for most of the effort (often 90% or more) spent on a successful software system
- A good design enables the system to adapt to new requirements while maintaining quality
  - Think about the long term, but don't prematurely optimize
- Good documentation enables others to understand the design

Intellectual property

- Patent
- Trade secret
- Trademark
- Copyright
- License
- Contracts

Interviewing

- Know your audience
- Communicate about yourself
- Be competent
- Be honest (about yourself, knowledge, etc.)
- You are evaluating them too

What you have learned in CSE 403; what you will learn later

- Compare your skills today to a quarter ago
  - Bottom line: Your project would be easy for you
    - This is a measure of how much you have learned
- Your next project can be much more ambitious
- You will continue to learn
  - Building interesting systems is never easy
    - Like all worthwhile endeavors
  - Practice is a good teacher
    - Requires thoughtful introspection
    - Don't learn *only* by trial and error!

Course evaluation

- Please complete the course evaluation form online
  - Useful to future students
  - Useful to course staff
  - Useful to the department

Go forth and conquer

- System building is fun!
  - It's even more fun when you build them successfully
- Pay attention to what matters
  - Use the techniques and tools of CSE 403 effectively