Software Development and Design Processes

CMPT 145

Software Development and Design Processes

- Producing quality software is expensive
  - Money
  - Time
- Software design processes are methodologies to help keep costs down.
- Basic methodologies:
  - Ad hoc (e.g., first semester computer science)
  - Waterfall (e.g., second semester computer science)
  - Iterative and incremental
  - More advanced methodologies for larger projects

Why development process matters

- Your time is the most important resource you have.
- If you have a manager, your time will be managed.
- If you are working for yourself, you have to manage and plan your own time.
- Planning your work is essential to productivity.
- You cannot design and code at the same time. These are two different skills.
- Debugging your poorly planned software will take up most of your time.

Bad advice

How to **fail** at programming assignments:

- Don't read the assignment carefully.
- Start programming without a plan, and without a design.
- Start late.
- Write lots of code without testing any of it.
- Do all your testing after you've written all your code
- Assume everything will go perfectly.

Common Processes

- Development: A plan for the life-cycle of a software project
  1. Waterfall model
  2. Iterative and incremental model
  3. Test-Driven Design
- Design: A plan for one step in the development process
  1. Stepwise refinement
  2. Test-Driven Design

Waterfall model tasks

1. Requirements
   - Figure out what the software is supposed to do.
2. Design
   - A plan describing how to meet the requirements.
3. Implementation
   - Write the code using the plan.
4. System Testing and Verification
   - Show that the system meets the requirements.
5. Maintenance
   - Install, deliver, update and upgrade, etc

Waterfall model: Requirements step

- The requirements for a project are acquired (negotiated) in advance.
- A requirements document is produced.
- Written in natural human language (e.g., English)
- All parties (e.g., developers, clients) need to agree on this.

Waterfall model: Design step

- Design is a planning process.
  Not an implementation process.
- A design plan describes the functions and modules needed to achieve the requirements.
  - Each function is described by a specification: inputs, outputs, assumptions.
- For us, written in natural human language (e.g., English)
  - Mathematics is often used as well.

Waterfall model

- PRO: Simple strategy.
- CON: Only works for smallish projects, like first/second year assignments.
- CON: In real projects, requirements often change (especially corrections)

Waterfall model for students

1. Requirements.
   - Study the assignment/project description.
   - Ask for clarifications as necessary.
2. Design
   - A plan describing functions, scripts, modules needed.
3. Implementation
   - Write the code (functions, scripts, modules)
4. System Testing and Verification
   - Write test cases for everything
   - Show that the system meets the requirements.
5. Maintenance
   - Tidy up, and submit.

Waterfall model deadlines for students

1. Requirements.
   - Done within one day of assignment release.
2. Design
   - Done no later than halfway between release and due date.
3. Implementation
   - This will take 4 times longer than you planned.
   - Include time for all testing.
4. System Testing and Verification
   - Done at least one day before the due date.
5. Maintenance
   - Tidy up, and submit.

Exercise 1

Use the Waterfall model to plan the development of the Coupon Collector's problem.

(a) State the requirements as clearly as possible.
(b) Plan the design, in terms of functions, modules, data organization, and test cases.
(c) Based on the design, estimate the time to implement the design, including test cases.
(d) Assuming a successful implementation, estimate the time to demonstrate the system, and check the requirements were met.
(e) Estimate the time to make the project presentable to a marker.

Exercise 2

Use the Waterfall model to plan the development of the Self-Avoiding Random Walks problem.

(a) State the requirements as clearly as possible.

(b) Plan the design, in terms of functions, modules, data organization, and test cases.

(c) Based on the design, estimate the time to implement the design, including test cases.

(d) Assuming a successful implementation, estimate the time to demonstrate the system, and check the requirements were met.

(e) Estimate the time to make the project presentable to a marker.

Iterative and incremental model

1. Start with a minimal project with only a few of the requirements.
2. Apply the waterfall model
3. Deliver the small system, get feedback.
4. Repeat steps 2-3, adding more requirements each time.

Iterative and incremental model: pro and con

- PRO: Development and testing is shorter.
- PRO: Faster feedback is good.
- CON: Without careful planning, requirements added later may be inconsistent with earlier versions of the system.

Iterative and incremental model for students

- Obtain full requirements for project.
- Identify the core functionality.
- Apply the waterfall model to the core.
- Apply incremental model to the remaining requirements.
    1. Consider remaining requirements, one at a time
    2. Design the addition of the requirements.
    3. Test, test, test.
    4. Repeat steps 2-3, adding more requirements each time.
- Tidy up, and submit.

Planning and designing for programming assignments

- Identify deliverables and requirements.
- Identify concepts related to your lecture material.
- Break down larger scripts into manageable pieces, using step-wise refinement.
- Design and plan test cases that you can use to check your work as you go.
- Use your analytical skills to challenge your assumptions.
- Turn off anything you know distracts you from work.

Exercise 3

Use the Incremental and Iterative model to plan the development of the Coupon Collector's problem.

(a) State the requirements as clearly as possible.
(b) Identify a core functionality from the requirements.
(c) Plan the design of the core functionality, in terms of functions, modules, data organization, and test cases.
(d) Identify other requirements that could be planned, estimate the time to complete them.
(e) Assuming a successful implementation, estimate the time to demonstrate the system, and check the requirements were met.
(f) Estimate the time to make the project presentable to a marker.

Exercise 4

Use the Incremental and Iterative model to plan the development of the Self-Avoiding Random Walks problem.

(a) State the requirements as clearly as possible.
(b) Identify a core functionality from the requirements.
(c) Plan the design of the core functionality, in terms of functions, modules, data organization, and test cases.
(d) Identify other requirements that could be planned, estimate the time to complete them.
(e) Assuming a successful implementation, estimate the time to demonstrate the system, and check the requirements were met.
(f) Estimate the time to make the project presentable to a marker.

Version Control

- Software tools to help you manage content creation, esp. software
- Each *version* is a full backup, with documentation, of the state of your project.
- Maintains a documented history of your work on a project
    - Switch between versions instantly
    - Make comparisons between versions
    - Develop multiple versions simultaneously
    - Allows for coordinated collaborations in team projects

Why you need version control

- You'll make mistakes in design and implementation.
- You'll employ incremental development
- You'll want to have a backup system
- You'll want to make experimental changes, and revert them.
- Every professional software developer uses it.