Software Engineering

CS305, Autumn 2020

Week 15

Class Progress… (last week)

Software Quality

- What is quality? General and software-specific definition.
- Metric for judging quality (COQ)
- Why improve quality?
- Approaches and Implementation guidelines for continuous improvement of quality: TQM, ISO, and CMM

Project Management

- Steps/activities in project management
  - Effort estimation and techniques – FP, COCOMO

Class This Week..

- Agile Methodologies
- Revision

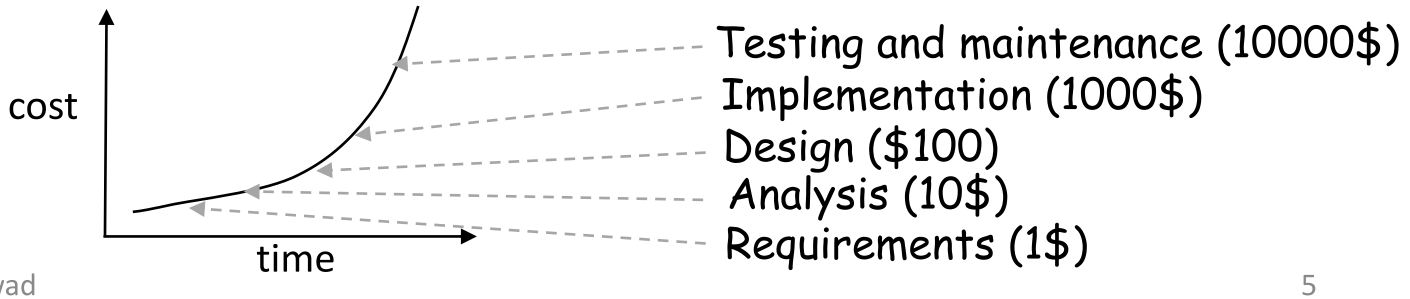
Agile Development Methodology

- Another type of software development methodology heavily based on testing.
- Also called Test Driven Development (TDD)
  - Recall PA1 that briefly introduced you to TDD:
    - Developed test specs based on SRS.
    - Implemented test specs (test cases and test suites) – Functional Testing (Black-Box testing)
- A group of software developers published the manifesto for Agile Software Development in 2001.
  - They had met to discuss lightweight software development processes

Why Lightweight Software Development?

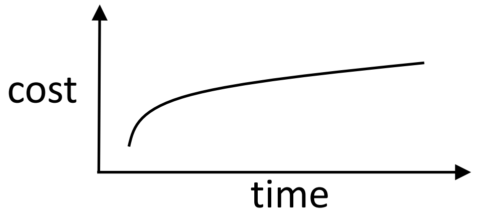
- Recall waterfall model:
  - A phase in the process stared only after the previous phase ended. Phases: `Requirements -> Design -> Implementation -> Testing -> Maintenance`
  - Very old (70s, some concepts date back to 50s), Not flexible w.r.t changing requirements and design
  - Good at catching errors early, which is important considering Boehm's observation of the cost of change:

Cost grows exponentially with time



Why Lightweight Software Development Method ? (Contd..)

- *What if the cost remained flat?*
  - Possible because of improvements in technology and tools:
    - punch cards for inputs and batch processing in job submission vs. faster compilation and execution times
    - assembly vs. high-level programming languages
    - slow vs. fast hardware
    - IDEs, Cloud, many more…



- Because of the shorter turnaround time, you can let time answer questions and resolve uncertainties inherent in software development. What this means….

Agile Methodology

- Delay investing in resources / plans that might never be used / realized. Ambiguity and volatility are inevitable

  There is value in waiting

- Implement upfront

  Focus on code rather than the design

  Deliver working software quickly and adapt quickly

- Get feedback and iterate

  Prioritize People over Processes (esp. customer)

- Focus on Simplicity (of design, implementation..)

  Does not mean create inadequate software.
  "Look for the simplest thing <u>that works</u>"

Xtreme Programming (XP)

XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements ***-Kent Beck***

- 4 Attributes: lightweight, humanistic, disciplined, software development
- Guidelines and *Principles*:
  1. Write tests *(to get feedback)*
  2. Restructure code often (*to simplify, to show courage*)
  3. Talk to fellow programmers and customers often *(communicate)*

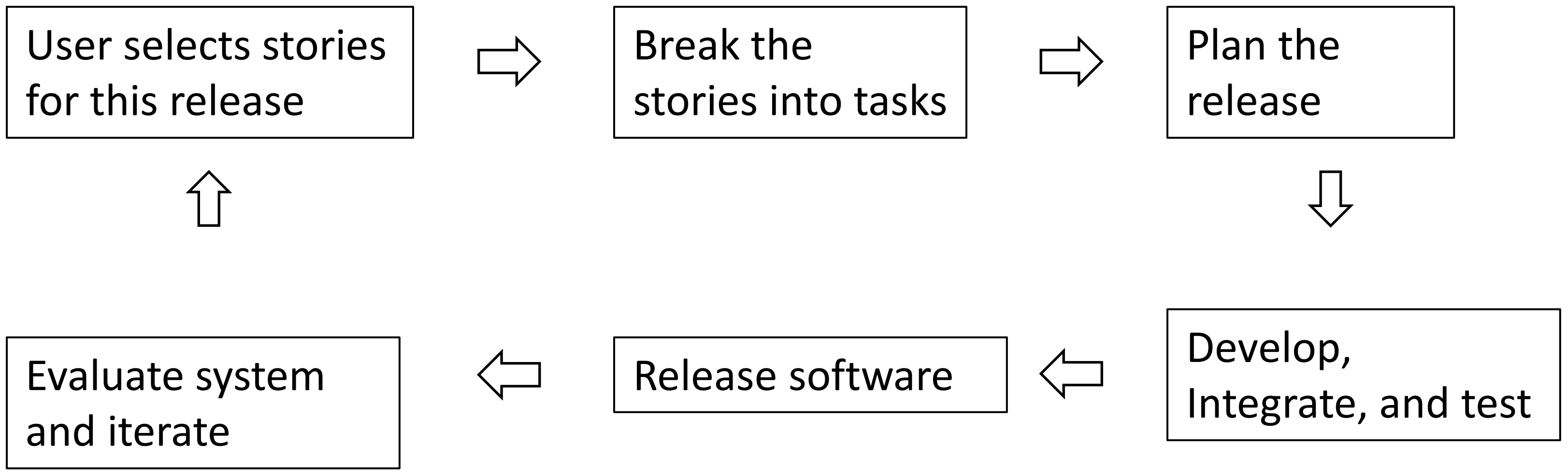
When you have to throw away code that doesn't work/unnecessary

XP in Practice

- Incremental planning
- Small releases
- Simple design
- Test first
- Refactoring
- Pair programming
- Continuous Integration
- On-site customer

Incremental Planning

- Assumes that the requirements are recorded on story cards, use cases, or scenarios.
- First, pick story (stories) for this release


User selects stories for this release
Break the stories into tasks
Plan the release
Develop, Integrate, and test
Release software
Evaluate system and iterate

Small Releases

- Rather than focusing on a big release consisting of a lot of stories, focus on small releases
  - Helps deliver business value faster => builds customer confidence
  - Gives rapid feedback and hence, adapt quickly to changing requirements
  - Reduces risks and gives a sense of accomplishment to developers

Simple Design

- Simple enough to just meet the requirements
  - No duplicated functionality
  - Fewest possible classes and methods
    - So adapting / changing is easier

Test-First Development

- If there is a feature, write test case for the feature and test before writing the feature itself
  - Do this for unit tests as well
  - You see that test fail initially (obviously). As you add more functionality, tests start passing.

Refactoring

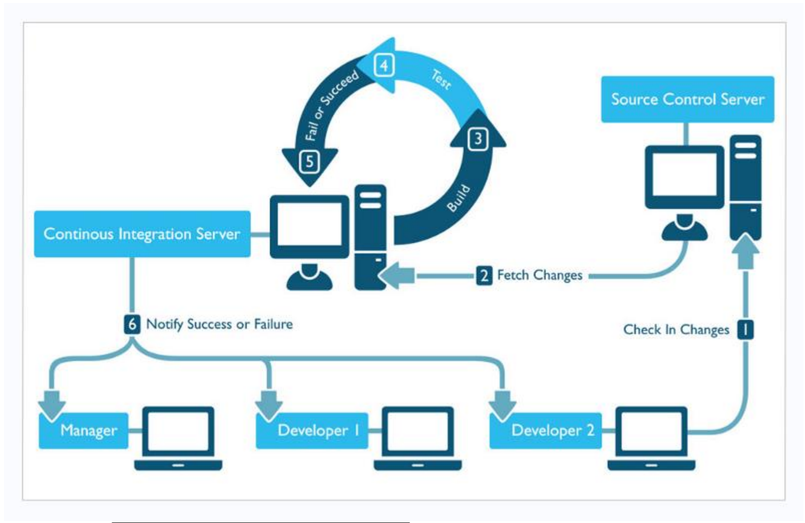
- Recall software refactoring from topics in software construction:
  - Transforming code to make it easier to read, maintain, and improve
- Refactoring is an important XP practice
- Done on-demand and not speculatively

Pair Programming

- All production code is written by two people looking at one machine (with one keyboard and one mouse)
- Study shows that productivity is equal to / better than two independent developers working
- Programmers play dual roles: programmer and strategizer (provider of out-of-context perspective)

Continuous Integration

- Recall from Week13:
  - Ongoing monitoring from integration to testing to deployment



$$\begin{tikzcd}
\text{Programming} \arrow[r, bend left] & \text{Local Tests} \arrow[l, bend left] \arrow[r, bend right] & \text{Integrate} \arrow[dl, bend left] \\
 & \text{System Tests} \arrow[ul] \arrow[d] & \\
 & {} &
\end{tikzcd}$$

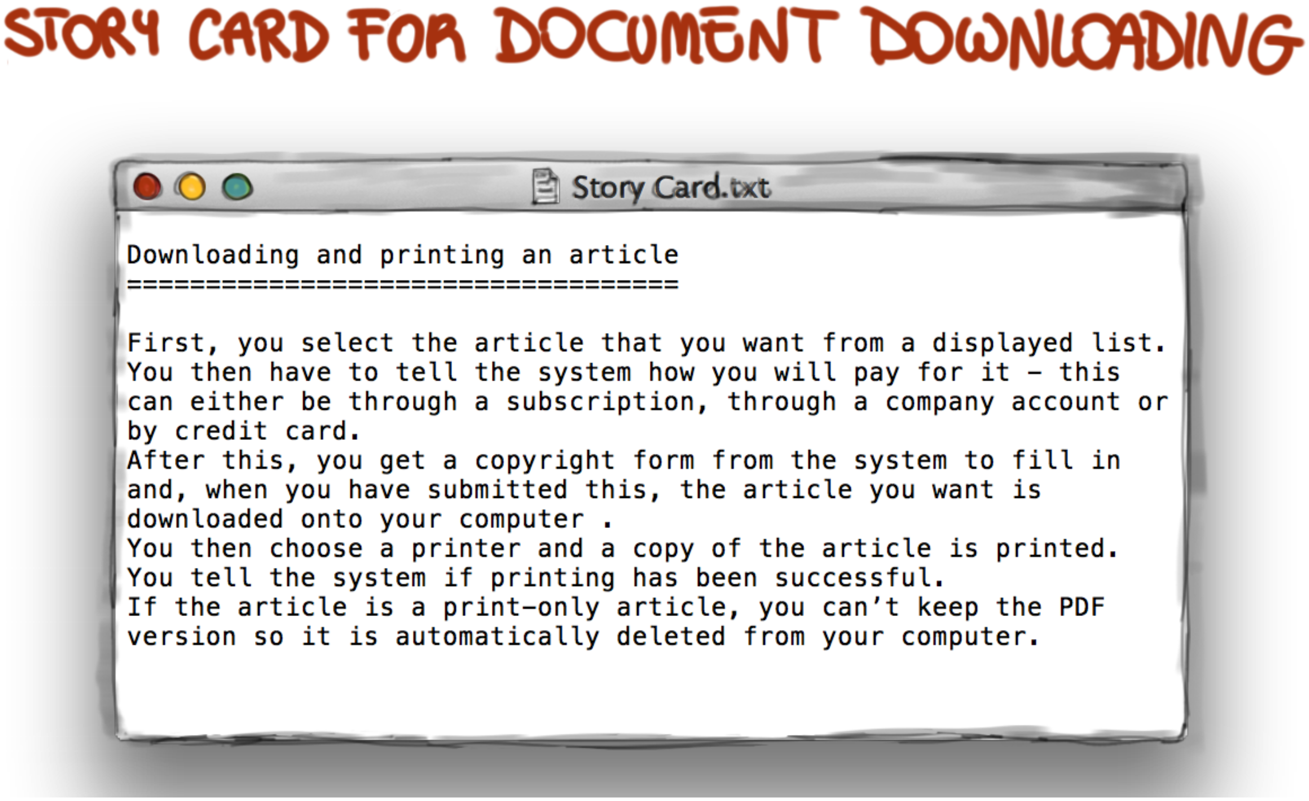
On Site Customer

- Customer is part of the team
  - Brings the requirements
  - Sits with the team

“If the system is not worth the time of one customer then it may not be worth building”

Requirements Engineering in XP

- Customer writes the requirements as story cards



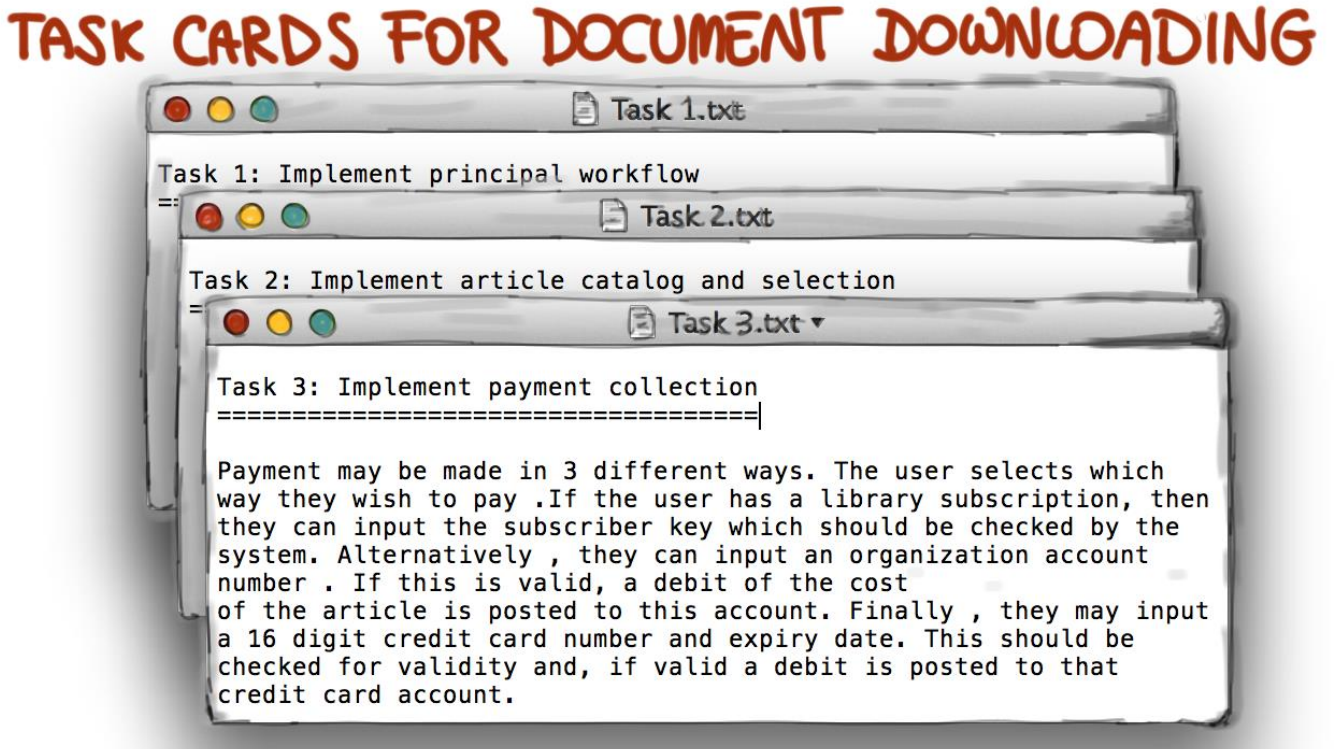
```
Downloading and printing an article
===================================

First, you select the article that you want from a displayed list.
You then have to tell the system how you will pay for it - this
can either be through a subscription, through a company account or
by credit card.
After this, you get a copyright form from the system to fill in
and, when you have submitted this, the article you want is
downloaded onto your computer .
You then choose a printer and a copy of the article is printed.
You tell the system if printing has been successful.
If the article is a print-only article, you can’t keep the PDF
version so it is automatically deleted from your computer.
```

Source: Alex Orso, CS6300

Requirements Engineering in XP

- The story cards are broken down into tasks and some tasks (story cards) are picked for next release



```
Task 1: Implement principal workflow
```

```
Task 2: Implement article catalog and selection
```

```
Task 3: Implement payment collection
====================================

Payment may be made in 3 different ways. The user selects which
way they wish to pay .If the user has a library subscription, then
they can input the subscriber key which should be checked by the
system. Alternatively , they can input an organization account
number . If this is valid, a debit of the cost
of the article is posted to this account. Finally , they may input
a 16 digit credit card number and expiry date. This should be
checked for validity and, if valid a debit is posted to that
credit card account.
```

Source: Alex Orso, CS6300