

CSE 403

Lecture 27

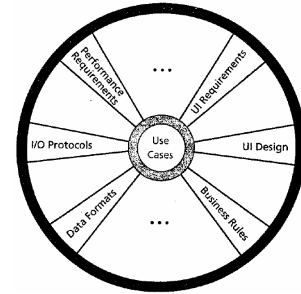
Course Wrap-up Discussion

slides created by Marty Stepp

<http://www.cs.washington.edu/403/>

Requirements

- How does the product you built differ from what you intended/expected to build?
 - What are some features you had to cut?
 - What are some implementation challenges you didn't anticipate?
 - Did you end up implementing all of your original use cases as planned? If not, what was cut/changed and why?
 - What phases, features, aspects took longer to complete than you thought? How does your initial estimation of your project's schedule compare to what really took place?



Teams and groups

- What are some of the challenges that came from working in a large team on a big project?
 - What were the roles of your team's members?
 - Did everyone work on individual tasks, or were there sub-groups?
 - How did the PM manage and collaborate with others?
 - What was the communication like between group members?
 - Was it as good as it should have been?
 - When, where, and how often did you meet in person to work?
 - How did you resolve conflicts?



Tools and technologies

- While building this project, you had to learn about new languages, tools, services, and technologies.

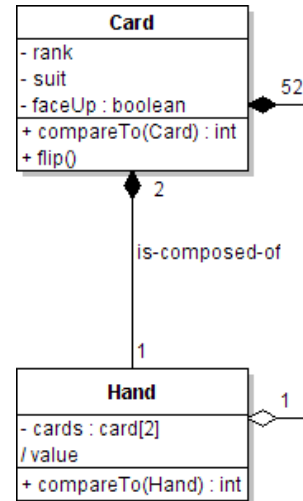
(Git/Github; Android; Heroku; testing tools; Jenkins)



- How difficult was this for you and for your group?
- What were some unforeseen challenges related to these technologies that hit your group during the project?
- If you had to do a project like this again, would you choose the same tools and technologies or switch, and why?
- Should we have done more to prepare you for using these tools?

Design

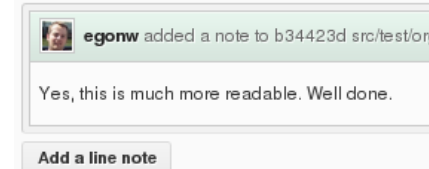
- This was (probably) your first time designing such a large-scale team project.
 - How difficult was it to come up with an initial design?
 - Does your final code look anything like your UML?
 - What design aspects were hard to anticipate, or came out very differently from your initial idea?
 - Was UML useful as a tool to talk about your design?
 - What about related techniques like CRC cards?
 - Do you feel like you could design a similar app well now?



GitHub, code reviews

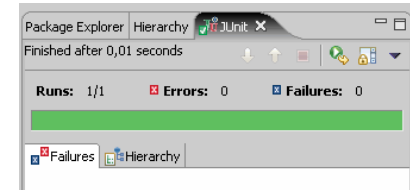
- We forced you to use a central GitHub repo and to do extensive code reviewing.
 - Was it helpful to use a central Github repo?
 - Did you feel like you knew Git/Github well?
 - Do you feel like you know it well now?
 - Did your group have any challenges related to the repo/Github?
 - What were the pros and cons of doing the code reviews?
 - How useful do you think code reviewing was on your project?
 - Did you learn any coding skills or good/bad practices from it?
 - Did it help your team to avoid any problems, bugs, bad code, etc.?

```
import org.openscience.cdk.CDKConstants
-import org.openscience.cdk.ChemObject;
import org.openscience.cdk.Molecule;
import org.openscience.cdk.exception.CD
import org.openscience.cdk.interfaces.I
@@ -352,11 +351,11 @@ public class Exten
    String filename = "data/mdl/chebise
    InputStream ins = this.getClass().g
    MDLV2000Reader reader = new MDLV200
-    searchmol = (Molecule) reader.read(
+    searchmol = reader.read(new MolecuL
```



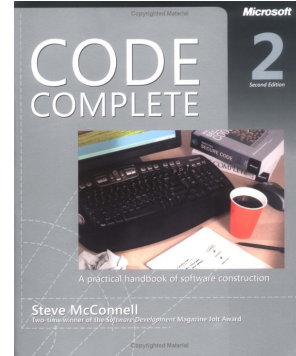
Testing

- We did a lot of unit testing, including coverage analysis, and also integration/system testing.
 - Did unit tests help you find any bugs or regressions?
 - How hard was it to get to the expected coverage percentage?
 - What kind(s) of testing were most interesting or useful?
 - integration, UI, usability, performance, reliability, security, ...
 - What kind of testing scheme would you use if a future large team project were entirely under your control?
 - Would you require unit testing? A minimum coverage?
 - What kind(s) of system testing, if any, would you use and why?



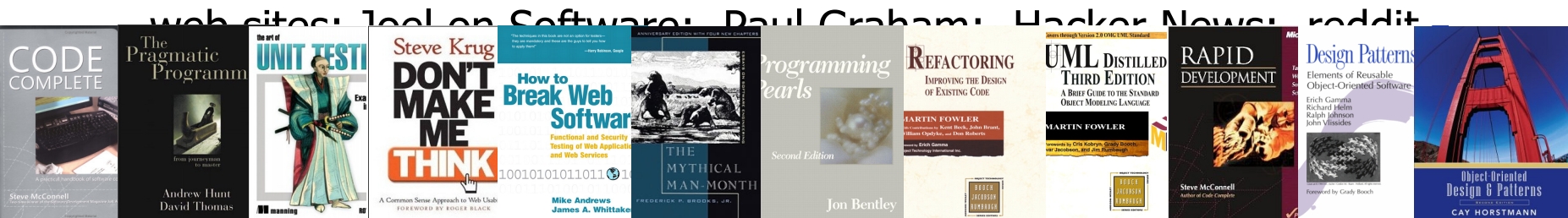
Readings

- Throughout the quarter we read articles, chapters, and sections about SW engineering topics.
 - What topics were interesting to read about?
 - What topics were tougher or less interesting?
 - Is there a topic we didn't read about but should have?
 - What do you think about the reading in the course overall?
 - Would the course be better with less/different expectations?



Further reading

- *Code Complete*, by Steve McConnell
- *The Pragmatic Programmer*, by Andrew Hunt / David Thomas
- *The Art of Unit Testing*, by Roy Osherove
- *Don't Make Me Think!*, by Steve Krug (usability)
- *How to Break [Web] Software*, by James Whittaker (testing)
- *The Mythical Man-Month*, by Fred Brooks
- *Programming Pearls*, by Jon Bentley
- *Refactoring*, by Martin Fowler
- *UML Distilled*, by Martin Fowler
- *Rapid Development*, by Steve McConnell
- *Design Patterns: Elements of Reusable ...*, by "Gang of Four"
- *Object-Oriented Design and Patterns*, by Cay Horstmann



Advice from past students

- Here is some advice given from students in past quarters:
 - "**Work together** (in the same place) as much as possible."
 - "Well-run and consistently scheduled **meetings** help a lot."
 - "We often underestimated tasks. If we had spent more time analyzing each task and **breaking it down** into smaller chunks, our estimated times would have been more accurate."
 - "Don't underestimate the difficulty of **learning** new languages, frameworks and tools."
 - "Make small, **frequent updates** and commits to your source repo. Not doing this leads to merges that can be a nightmare."
- What advice do *you* have for future CSE 403 students?