


**6.170 Lecture 19**  
**Project Management**  
**and Control, part 1**



John Guttag  
MIT EECS

**6.170 Plan for Rest of Lectures**

---

**Focus on two things**  
Final project  
Things that will help  
Demonstration - Wednesday  
Software in the real world

**Some brief project-related advice**  
Documentation, team organization, goals

**Project management, system integration and testing**

**No guest lecture tomorrow**  
Speaker cancelled over weekend

---

©Michael Ernst/John Guttag Spring 2001 Slide 2

**6.170 Documentation**

---

**Goal: reduce opportunity for misunderstandings**

**Do not rely on oral communication**  
Record decisions in writing

**Not a waste to document things that may change**  
Provide target for criticism  
Good way to find mistakes early

**When a teammate finds a flaw in your design**  
Triumph for both of you  
He/she found flaw  
You explained things well

---

©Michael Ernst/John Guttag Spring 2001 Slide 3

**6.170 Documentation, cont.**

---

**Use version control system**  
For documentation as well as code

**Keep change log**  
Date  
Person  
Reason  
Summary

**Code is most important documentation**  
Property of project  
Not property of programmer  
Why shared coding standard is so important  
Must be enforced by management

---

©Michael Ernst/John Guttag Spring 2001 Slide 4

**6.170 Today's Project is Tomorrow's Legacy**

---

A legacy is not necessarily bad

**Last Will and Testament,**  
**John D. Rockefeller**

**Systems tend to grow over time**  
Modifying a large old program is hard  
Often possible to shrink it

**Replacement by new small program often useful**  
Essential systems best replaced piecemeal

---

©Michael Ernst/John Guttag Spring 2001 Slide 5

**6.170 Team Organization**

---

**Most importantly, you need one**

**Two distinct considerations**  
How decisions get made (management structure)  
How information flows (communication structure)  
Have a plan for this

**Two poles**  
Centralized  
Needed for large projects  
Decentralized  
What I recommend for your project

---

©Michael Ernst/John Guttag Spring 2001 Slide 6

**6.170** **Decentralized Organization**

**Key decisions made jointly**  
Requirements  
High level design  
Schedule  
Who will work on what  
Response to slippage

**Lower level design and code exchanged for examination**  
Everyone responsible for everything  
Code reviews tremendously helpful  
Try it, you'll like it

**Do need a leader at all times**  
Chair meetings, provide agenda, make decisions  
Leadership changes based on relevant expertise

---

©Michael Ernst/John Guttag Spring 2001 Slide 7

**6.170** **Goals**

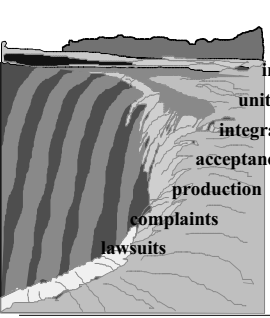
**Shared goals essential**

**Question of priorities in real world**  
Schedule  
Functionality  
Maintainability

---

©Michael Ernst/John Guttag Spring 2001 Slide 8

**6.170** **The Waterfall Model**



requirements analysis  
design  
implementation  
unit testing  
integration testing  
acceptance testing  
production  
complaints  
lawsuits

---

©Michael Ernst/John Guttag Spring 2001 Slide 9

**6.170** **Waterfall Model**

**Works well when**  
The requirements are high quality and stable  
The developers have previously built similar systems  
The project is not very complex

**When used in other situations**  
Lots of rework  
High late stage costs  
Because everyone gets it wrong the first time

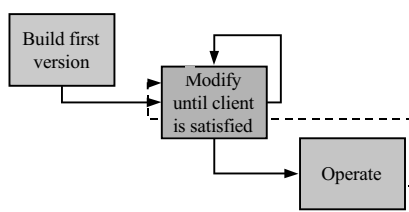
**Rarely right for most companies**

**Might be right for late stage development**  
E.g., customization

---

©Michael Ernst/John Guttag Spring 2001 Slide 10

**6.170** **Hacking Model**



```
graph TD; A[Build first version] --> B[Modify until client is satisfied]; B --> B; B --> C[Operate]; C -.-> B;
```

---

©Michael Ernst/John Guttag Spring 2001 Slide 11

**6.170** **Hacking Model Has Problems**

**No specification**  
Figure out specification as you code

**Useful for**  
Short programs

**If applied to something too complex**  
Lots of rework  
High costs at a late stage

---

©Michael Ernst/John Guttag Spring 2001 Slide 12

**6.170 Characteristics of a Better Model**

---

**Reduce cost of late-stage problems**  
Adequacy of specification  
Performance limitations

**Rapidly build necessary competencies**  
Use new knowledge to improve design

**Focus on reducing risk**  
Make (and discover) mistakes early

**Look at one better model**  
Not the only reasonable model  
Different situations call for different models

---

©Michael Ernst/John Guttag Spring 2001 Slide 13

**6.170 Be Mindful of Cost of Correcting a Defect**

---

**Measured by IBM**  
Design: \$10  
Programming: \$100  
QA: \$1000  
Post-deployment: \$25,000

Stage	Cost
Design	\$10
Programming	\$100
QA	\$1,000
Post-deployment	\$25,000

---

©Michael Ernst/John Guttag Spring 2001 Slide 14