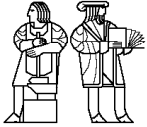
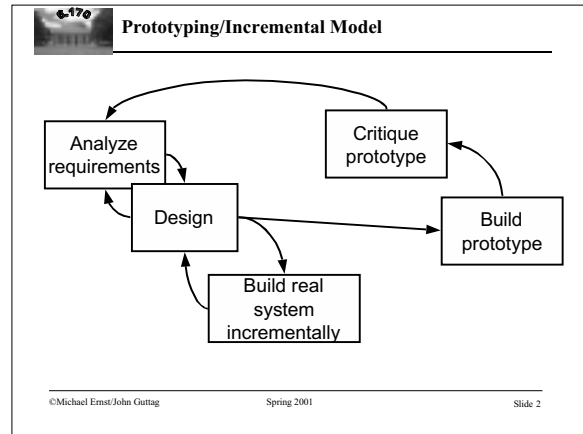


6.170 Lecture 20 Project Management and Control, part 2



John Guttag
MIT EECS



Prototype Phase

Not hacking
Carefully design prototypes
Discard prototypes rather than change-until-done

Part of requirements analysis
Learn the customer's needs more precisely
Build a mock-up, demo it, get feedback

Quick and dirty?
Dirty is easy

Lots of wasted work?
Plan to throw one away, you will anyway
Much cheaper if mistakes discovered early

©Michael Ernst/John Guttag Spring 2001 Slide 3

Some Uses of Prototypes

Sell project to management

Understand requirements
Build a mock up, get feedback from users
Learn the customer's needs

Understand building blocks
Hardware
Programming environment
Tool kits and libraries

Understand design
Find "gotchas" early on

©Michael Ernst/John Guttag Spring 2001 Slide 4

Understand Building Blocks

May have specifications that are
Ambiguous
Incomplete or inaccurate
Unclear about preconditions
Unclear about performance

Trying building blocks out early in appropriate context
Informs design
Modularizes debugging process

Example
Understanding timing considerations in Windows CE

©Michael Ernst/John Guttag Spring 2001 Slide 5


Effective Prototyping

A prototype is built to answer questions
Know what questions you wish to answer
Write them down at the start

Use list to decide
What functionality to implement
What tests to run
When discard prototype

Keep a lab notebook
Record decisions and rationale
Treat as log
Don't revise or throw out

©Michael Ernst/John Guttag Spring 2001 Slide 6


 **Prototyping Pitfalls**

Worthless prototype
Doesn't answer right (or any) questions

Failure to discard prototype
Longer you wait, the harder it gets
Must have drop dead date for prototyping phase

Second system effect
Prototype makes problem seem easier than it is
Much of the work goes to last 10% of getting it right
Features get added or schedule compressed

©Michael Ernst/John Guttag Spring 2001 Slide 7


 **Incremental Development Phase**

Short cycles, weeks not years
Design Redesign
Implement Reimplement
Validate Revalidate
Assess risk Reassess risk
 Consolidate & Optimize

Smallest steps representing visible progress
New behavior
Better performance
Reduced amount of code
Better platform for future development

Not same as prototyping phase
Not throw away code

©Michael Ernst/John Guttag Spring 2001 Slide 8

 **Advantages of Incremental Model**


Feedback
Easier to measure progress
Better documentation
Reality check

Leads to more modular designs
Piecewise validation easier
Changes easier

Better customer-vendor relationship
Less adversarial
Shared problem solving

Difficulty
Executives/customers must pay attention
A serious problem in real world

©Michael Ernst/John Guttag Spring 2001 Slide 9

 **Scheduling**


“More software projects have gone awry for lack of calendar time than for all other causes combined.”
-- Fred Brooks

“More students have ...” -- John Guttag

Three central questions of software business
When will it be done?
How much will it cost?
When will it be done?

Facts
1. Estimates almost always too optimistic
2. Estimates reflect what one wishes to be true
3. We confuse effort with progress
4. Progress is poorly monitored
5. Slippage is not aggressively treated

©Michael Ernst/John Guttag Spring 2001 Slide 10

 **Scheduling Is Crucial**


Usually gets far less attention than appropriate
Made to fit other constraints

Needed to make slippage visible
Like an annual business plan
Must be objectively checkable by outsiders

Unrealistically optimistic schedules a disaster
Decisions get made at wrong time
Decisions get made by wrong people
Decisions get made for wrong reasons

The great paradox
Everything will take twice as long as you think
Even if you know that it will take twice as long as you think


©Michael Ernst/John Guttag Spring 2001 Slide 11

 **In Large Projects**

Estimates don't change as activity approaches
No matter how wrong they end up being

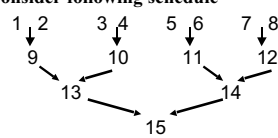
Once activity has started
Overestimates of cost come steadily down
Underestimates do not change until near scheduled end

©Michael Ernst/John Guttag Spring 2001 Slide 12

 **Optimism Root of Problem**

People assume that all will go well
Every task will take as long as it ought to take


Consider following schedule



Suppose that on average each task takes as long as planned
Odd tasks over-run by 20%; even under-run by 20%

How close to schedule does project finish?
Late by 20%

©Michael Ernst/John Guttag Spring 2001 Slide 13


 **Estimating With One's Heart**

Desires of client
Can dictate scheduled completion date
Cannot dictate the actual completion date

Don't let client push you into an unrealistic plan
Have courage to trust pessimistic estimates
Not an easy thing
"Madame No" sometimes gets fired

Evaluate your team honestly
Remember productivity differs greatly

©Michael Ernst/John Guttag Spring 2001 Slide 14

 **Effort Is Not the Same as Progress**


Cost is product of workers and time
Easy to track

Progress is more complicated
Hard to track

People don't like to admit lack of progress
Think they can catch up before anyone notices
Not usually possible

Design process and architecture to facilitate tracking

©Michael Ernst/John Guttag Spring 2001 Slide 15

 **How Does a Project Get to Be One Year Late?**

One day at a time

It's not the hurricanes that get you

It's the termites
Tom missed a meeting
Mary's keyboard broke
The compiler wasn't updated
...

Remember, "It ain't over 'till it's over."
If you find yourself ahead of schedule
Don't relax
Don't add features

©Michael Ernst/John Guttag Spring 2001 Slide 16