GREAT SOFTWARE DEVELOPMENT - II

Content from Chapter 1 of "Head First Software Development", Pilone et al.

Miami University Software Technology & Analysis Group (MUSTANG)
Computer Science & Software Engineering
Miami University, Oxford, Ohio, USA

AGENDA

- Review
- Project
- Quiz #1

Software Runs Everything

- Software connects us
- Software protects us
- Software cures us
- Software entertains us
- Software is everywhere
- Software is in everything
- *Software is important*

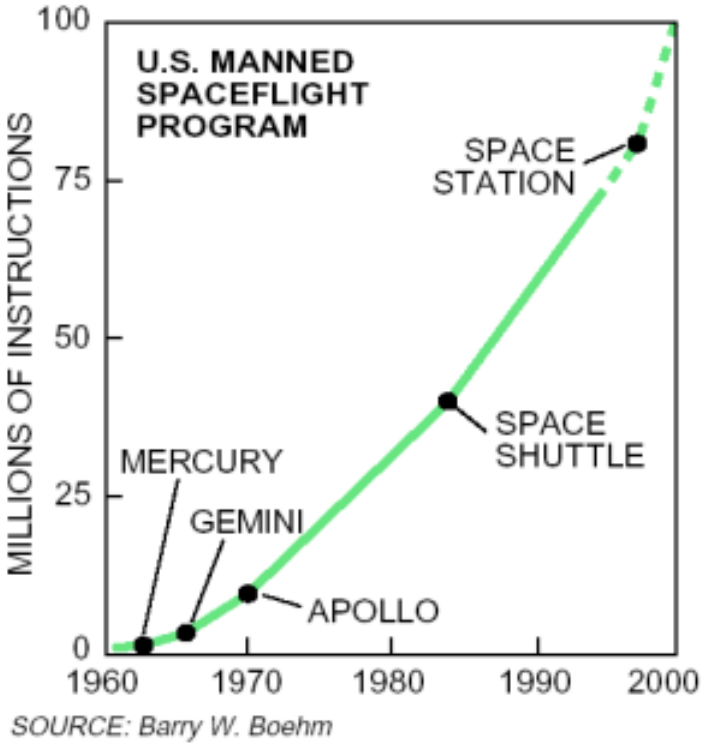
The global software market had total revenues of $292.9 billion in 2011.

-- MarketLine

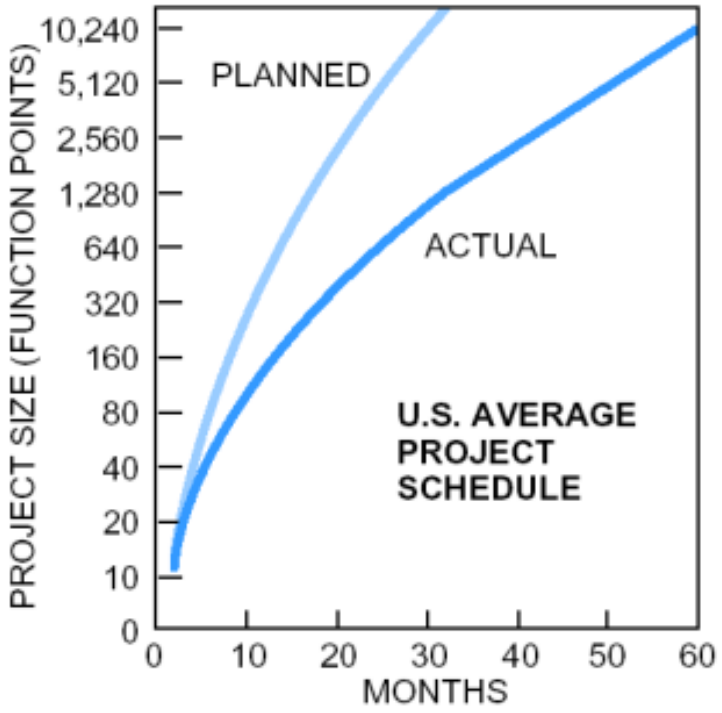
But Building Great/Good Software is Hard

- 2/3 of projects late
- 1/2 run over budget
- 1/4 of all projects cancelled

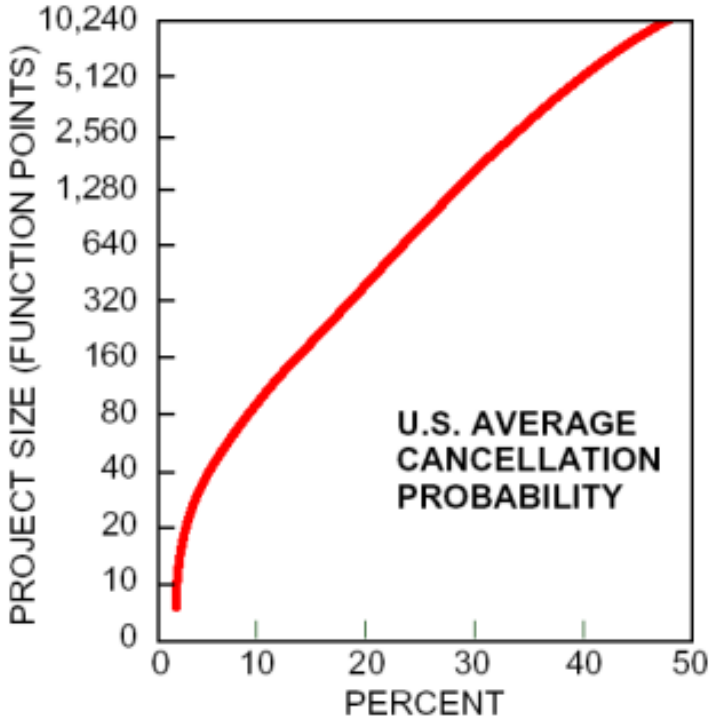
WHY? SCALE



SOURCE: Barry W. Boehm



SOURCE: Software Productivity Research



SOURCE: Software Productivity Research

Users want more and more features

What is "Software Engineering"

- What is "Software"?
- What is "Engineering"?
- What does it mean to be an "Engineer"?

WHAT IS "SOFTWARE"?

- Okay. We know this one!

- A product that results from a computer program.
- It isn't the code, itself, but what is realized once the code is actually running

WHAT IS "ENGINEERING"?

- We've got this, intuitively.

- The use of well-accepted and well-understood methodology to apply science and technology to the design and analysis of individual products or systems.
- It is worth noting that methodology includes tools, techniques, and processes, etc.

WHO IS AN "ENGINEER"?

- We've also got this!

- An engineer is someone who uses science and technology to design and/or develop products, but who is uniquely characterized among other developers by the use of a well-accepted and well-understood methodological, process-oriented approach.

WELL-UNDERSTOOD? WELL-ACCEPTED?

- It is important that the methodology be well-understood and well-accepted because it enables communication and inspires confidence
  - Lessons are learned over time. Best practices leverage this.
  - All other things being equal, projects are more successful when the methodology is well-understood and accepted by everyone.
  - Being on the same page improves communication, coordination, etc.

Is there One Right Way?

- Want to remodel kitchen
- One contractors process:
  - I understand what you want: Give me $30,000 and I'll build it.
- Another contractor's process:
  - Let me create a drawing, show it to you, and make sure we're on the same page.
  - Then, let me add the fine details we didn't discuss, and get your feedback and approval.
  - Then, let me put together a project plan, scheduling the trades (electricians, plumbers, HVAC mechanics, etc.), and make sure it makes sense to you.
  - Then, let me put the project out for bids, pick the best bid in each category, and review it with you, to make sure you like the details of how the projects will proceed.
  - Let me show you the final schedule, including city inspections – and your inspections – to make sure that you are happy every step of the way, and if not, we can fix it before it gets too expensive to undo.
  - Let's also talk about the final walk-through and acceptance checklist
  - Does this make sense? Can I get a down payment?

The Right Way?

- Depends upon many factors:
  - Clarity and stability of requirements
  - Time to delivery and cost of delay
  - Cost of failure
  - Complexity/Interactions within product

In this class you will learn the:

- Why,
- What, and
- How

Of great software development

QUALITY CONTROL: A SHORT HISTORY

Quality control in early manufacturing was **Product-Centric** ("what")

- Regularly test **product** outputs
- Make adjustments to factory as needed
- *But what to fix?*

In mid 20th century, shift to **Process-Centric** quality control ("how")

- Still test **product** outputs
- Also measure **process** elements
  - Plans, people, tools
- Use **cause-and-effect model** to adjust factory as needed

- **SE has inherited this legacy**
- ***SE methods are process-centric***

WHAT'S A "SOFTWARE PROCESS" ANYWAY?

- It's the "how" that produces the "what" – quality software
  - Desired, on time, under budget
- A prescribed sequence of steps. Steps include:
  - Planning, Execution, Measurement, Product, and process itself
    - Examples: bugs, progress, time, conflicts, re-work
- **A software process is a self-aware algorithm.** Observes and adapts according to measurements
- Agile processes are adaptive to the "customer"
  - Features, schedule, budget, priorities, markets, change
  - Must measure these as well as internal elements (correctness)

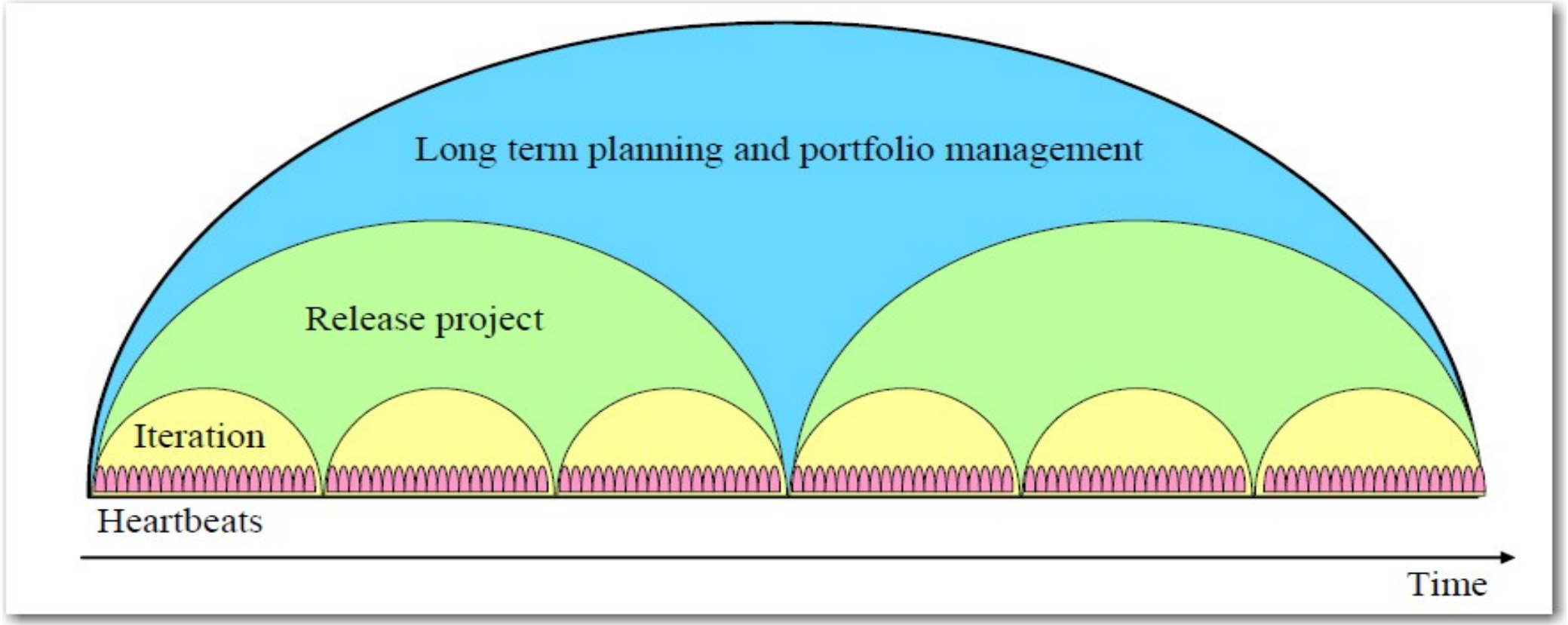
VENERABLE MODEL: WATERFALL

- The “Waterfall”
  - Requirements Analysis
  - Design
  - Implementation
  - Verification
  - Operation/maintenance
- Bad
  - Doesn’t allow for change or discovery of errors along the way
  - Fatal flaw in many situations

ITERATIVE DEVELOPMENT

- Development is organized into a series of short fixed-length mini-projects called **iterations**.
- The outcome of each iteration is a tested, integrated and executable system.
- An iteration represents a complete development cycle:
  - it includes its own treatment of *requirements*, *analysis*, *design*, *implementation* and *testing* activities.

ANOTHER VIEW OF THE ITERATIVE APPROACH



From: Rautiainen, K. & Vähäniitty, J. 2010, "Chapter 1: Using Time Pacing to Manage Software Development" in *Towards Agile Product and Portfolio Management, eds. V.Heikkilä, K. Rautiainen & J. Vähäniitty, Espoo: Aalto University, pp. 2-30.*

Advantages of an Iterative and Agile Process

- Reduce risks
  - Risks are identified early, progress is easier to see.
- Get a robust architecture
  - Architecture can be assessed and improve early.
- Handle evolving requirements
  - Users provide feedback to operational systems.
  - Responding to feedback is an incremental change.
- Allow for changes
  - System can adapt to problems
- Attain early learning
  - Everyone obtains an understanding of the different workflows early on.

WHAT ARE THE ATTRIBUTES OF GOOD SOFTWARE

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
  - Maintainability ➔ Software must evolve to meet changing needs;
  - Dependability ➔ Software must be trustworthy;
  - Efficiency ➔ Software should not make wasteful use of system resources;
  - Acceptability ➔ Software must accepted by the users for which it was designed.
    - This means it must be understandable, usable and compatible with other systems.

Software Costs

- Software costs are the dominant system cost
- Maintenance costs often greater than development costs
- Software Engineering addresses cost-effective development

What are the costs of Software Engineering

- Roughly 60% of costs are development costs, 40% are testing costs.
  - For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

SOFTWARE CRISIS

- Characteristics
  - Low quality software
  - Unpredictability (cost, effort)
- Solutions
  - Automation
  - Software engineering

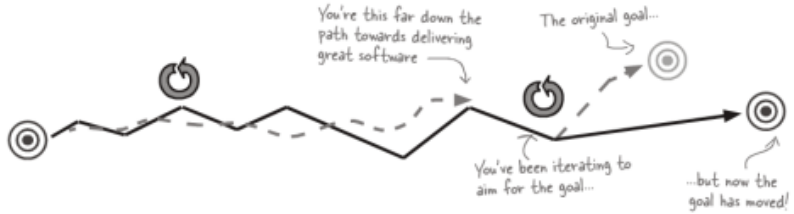
PROJECT

- Project begins in March
- 1 large planning phase, 3 Iterations based on SE processes learned in class
- Will have to decide on your own what technologies you will be using based on your team's abilities or learning desires
  - e.g., Java Web Technologies
- 3 Demonstrations
- Largest portion of final grade (50%)

QUIZ #1: QUESTION #4 – PRODUCTIVITY

- Why doesn't releasing to the customer all the time kill productivity?

RECAP



- Iteration is example of general idea of **making a large project act like a small one**
  - *Remember*: real problem in development is **scale**

NEXT CLASS

- Introduction to Software Requirements!