Lecture 3

Software Development (II)

1

Review

- Rapid Software Development Strategy
  - *Four dimensions of development: P, P, P, T*
- Class-mistake avoidance
- Development fundamentals
  - Management fundamentals
  - *Technical Fundamentals*
  - *Quality-Assurance Fundamentals*

2

Today's Topic

- Presentation of your business proposal
- Technical Fundamentals
- Quality-Assurance Fundamentals
- Life-Cycle Models

3

Presentation

4

Development Fundamentals

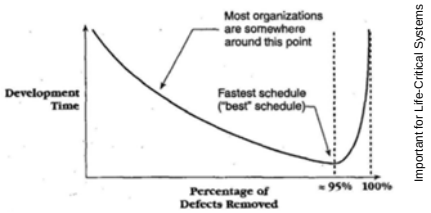
Quality-Assurance Fundamentals

How much does it cost not to find a defect
Error-Prone Modules
Testing
Reviews

5

Q-A Fundamentals

- When a software product has too many defects, developers spend more time fixing the software than they spend writing it.



Source: RD

95% of defect removal seems to achieve shortest schedule, least effort, and highest level of user satisfaction.

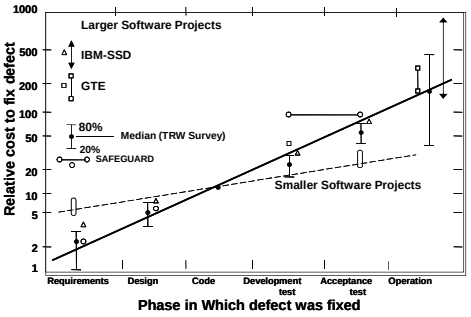
6

Q-A Fundamentals
How much does it cost not to find a defect

- Each hour spent on quality-assurance activities such as design reviews saves from 3 to 10 hours in downstream costs.
- A requirement defect that is left undetected until construction or maintenance will cost 50 – 200 times as much to fix as it would have cost to fix at requirements time.
- More generally, a defect that isn't detected upstream (during requirements or design) will cost 10 – 100 times as much to fix downstream (during testing) as it would have cost to fix at its origin.
- Studies have found that reworking defective requirements, design, and code typically consumes 40 to 50 percent of the total cost of software development (Jones 1986b; Boehm 1987a).

7

Increase in Software Cost-to-fix vs. Phase (1976)



8

Boehm (2006), "A View of 20th and 21st Century Software Engineering", ICSE 2006 Keynote Address

Q-A Fundamentals
Error-Prone Modules

- An error-prone module is a module that's responsible for a disproportionate number of defects. On its IMS project, IBM found that 57 percent of the errors were clumped in 7 percent of the modules (Jones 1991).
- Barry Boehm reports that about 20 percent of the modules in a program are typically responsible for about 80 percent of the errors (Boehm 1987b).
- Normal modules cost about $500 to $1,000 per function point to develop. Error-prone modules cost about $2,000 to $4,000 per function point (Jones 1994).
- Error-prone modules tend to be more complex than other modules in the system, less structured, and unusually large. They often were developed under excessive schedule pressure and were not fully tested.
- If a module's error rate hits about 10 defects per 1000 lines of code, review it to determine whether it should be redesigned or re-implemented. If it's poorly structured, excessively complex, or excessively long, redesign the module and re-implement it from the ground up. You'll save time and improve the quality of your product.

9

Q-A Fundamentals
Testing

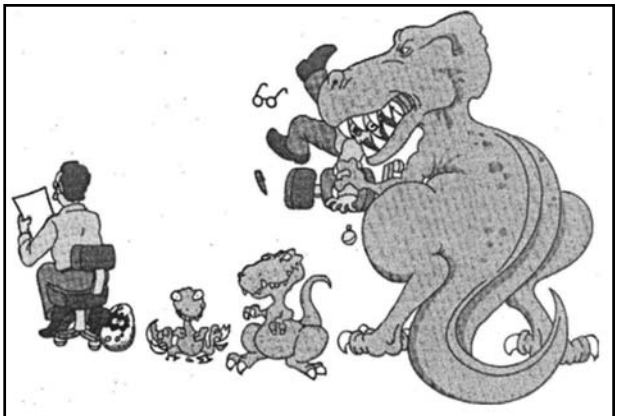
- Two types of testing:
  - Unit tests (by developers)
  - System tests (by independent testers)
- Effectiveness varies greatly
  - Unit testing can find 10% – 50% of the defects in a program.
  - System testing can find 20% – 60% of a program's defect.
  - Cumulative defect-detection rate is often < 60%.
  - The remaining errors are found either by other error-detection techniques such as reviews, or by end-users.
- Reviews
  - Walkthroughs

10

Q-A Fundamentals
Reviews

- Technical Reviews – to detect defects in requirements, design, code, test cases, etc.
  - Walkthroughs: can detect 30% – 70% of the errors in a program (two heads are better than one!)
  - Code reading: a more informal review process applied to code. Code-reading can detect twice as many defects per hour as testing (Card 1987)
  - Inspections: can detect 60% - 90% of the defects in a program; can be used in early development cycle; can produce net schedule savings of from 10% - 30%.
- Reviews are more effective than testing

11



12

Life-Cycle Models

13

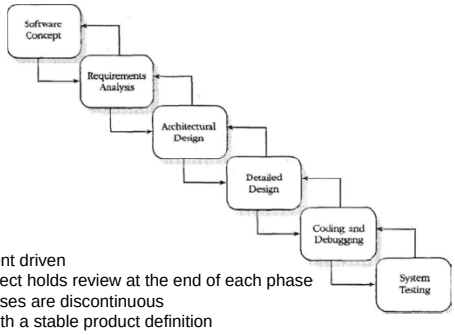
Lifecycle Planning

- Every software project goes through a lifecycle
  - Activities from beginning to the end
- There are several software lifecycle models
  - Pure waterfall
  - Code-and-fix
  - Spiral
  - *Modified Waterfalls*
  - Evolutionary Prototyping
  - Staged Delivery
  - Design-to-Schedule
  - Evolutionary Delivery
  - Design-to-Tools
  - Commercial Off-the-Shelf Software

14

15

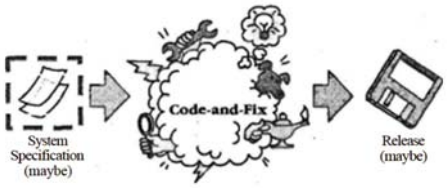
(1) Waterfall Model



Document driven
The project holds review at the end of each phase
The phases are discontinuous
Great with a stable product definition
Good for well-defined maintenance release of an existing product
Disadvantage: Rigid, needs to fully specify the requirements

16

(2) Code-and-Fix



Advantage:
1) no overhead, show signs of progress right away;
2) requires little experience
3) Good for tiny projects

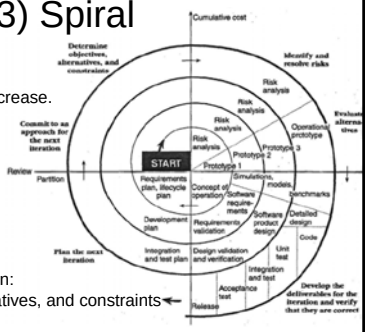
Disadvantage:

17

(3) Spiral

•Advantage:
- •As costs increase, risks decrease.
- •Risk oriented

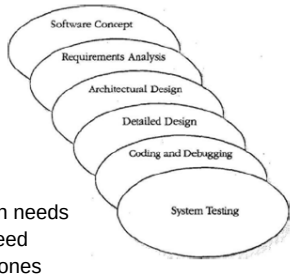
•Disadvantage:
- •Complexity



6 steps (options) in each iteration:
1) Determine objectives, alternatives, and constraints
2) Identify and resolve risks
3) Evaluate alternatives
4) Develop the deliverables for the iteration, and verify that they are correct
5) Plan the next iteration
6) Commit to an approach for the next iteration

18

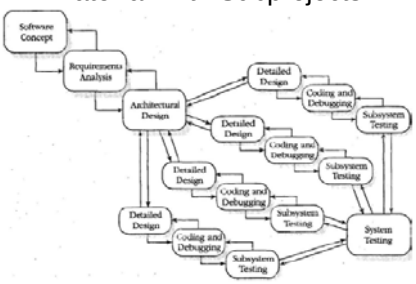
(4) Modified Waterfalls
Sashimi



- Waterfall with overlaps
- Reduced documentation needs
- Faster development speed
- More ambiguous milestones
- Harder to track progress

19

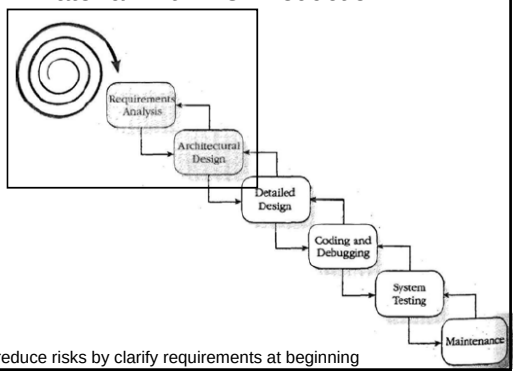
(4) Modified Waterfalls
Waterfall with Subprojects



- Break implementation into subprojects that run in parallel
- Risk: unforeseen interdependences

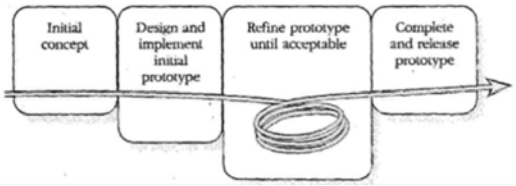
20

(4) Modified Waterfalls
Waterfall with Risk Reduction



- Try to reduce risks by clarify requirements at beginning

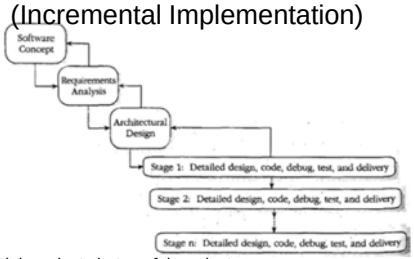
(5) Evolutionary Prototyping



- Begin develop the most visible aspects of the system
- Demonstrate the part to the customer and then continue to develop the prototype based on the feedback
- Repeat the previous step until the prototype is good enough
- Complete the unfinished part of the system
- Very useful when requirements are changing rapidly
- It is impossible to know how long it will take to complete the product

22

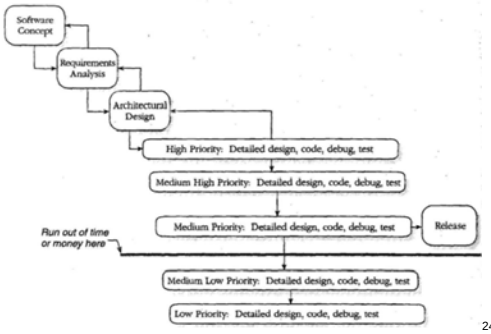
(6) Staged Delivery
(Incremental Implementation)



- Delivers partial products but useful products to the customer
- It will not work without careful planning at both the management and technical levels.
  - Avoid components developed at stage 2 depend on stuff at stage 4.

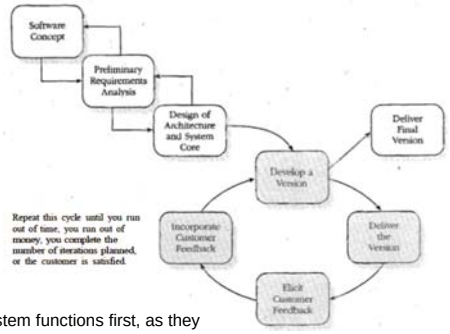
23

(7) Design-to-Schedule



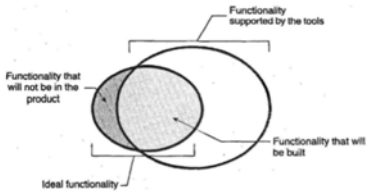
24

(8) Evolutionary Delivery



Deliver core system functions first, as they are unlikely changed by customers

(9) Design-to-Tools



- Tools
  - "code" and "class libraries"
  - Code generators
  - Rapid-development languages
- This model can be combined with other flexible lifecycle models.

26

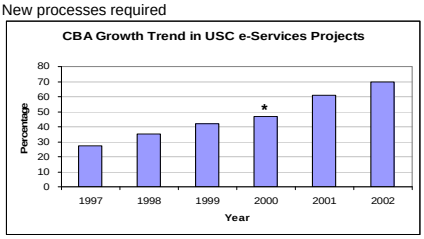
(10) Commercial Off-the-Shelf Software

- Buy commercial software
  - Rarely satisfy the needs
  - Available immediately

27

COTS: The Future Is Here

- Escalate COTS priorities for research, staffing, education
  - Software is not "all about programming" anymore
  - New processes required



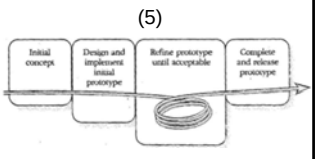
• CBA: COTS-Based Application
* Standish Group CHAOS 2000 (54%)

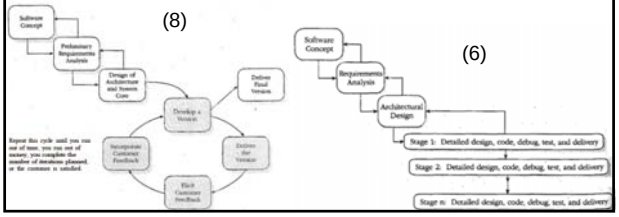
05/25/06 © USC-CSE 28

Boehm (2006), "A View of 20th and 21st Century Software Engineering", ICSE 2006 Keynote Address

Comparison between (5), (6) and (8)

(5) Evolutionary Prototyping
(6) Staged Delivery
(8) Evolutionary Delivery





Lifecycle model strengths and weaknesses

Lifecycle model capability	Pure Waterfall	Code-and-Fix	Spiral	Modified Waterfalls	Evolutionary Prototyping	Staged Delivery	Evolutionary Delivery	Design-to-Schedule	Design-to-Tools	Commercial Off-the-Shelf Software
Work with poorly understood requirements	Poor	Poor	Excellent	Fair to Excellent	Excellent	Poor	Fair to Excellent	Poor to Fair	Fair	Excellent
Works with poorly understood architecture	Poor	Poor	Excellent	Fair to Excellent	Poor to fair	Poor	Poor	Poor	Poor to Excellent	Poor to Excellent
Produces highly reliable system	Excellent	Poor	Excellent	Excellent	Fair	Excellent	Fair to Excellent	Fair	Poor to Excellent	Poor to Excellent
Produces system with large growth envelope	Excellent	Poor to fair	Excellent	Excellent	Excellent	Excellent	Excellent	Fair to excellent	Poor	N/A
Manage risks	Poor	Poor	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Poor to fair	N/A
Can be constrained to a predefined schedule	Fair	Poor	Fair	Fair	Poor	Fair	Fair	Excellent	Excellent	Excellent
Has low overhead	Poor	Excellent	Fair	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Excellent
Allows for midcourse corrections	Poor	Poor to excellent	Fair	Fair	Excellent	Poor	Fair to Excellent	Poor to Fair	Excellent	Poor
Provides customer with progress visibility	Poor	Fair	Excellent	Fair	Excellent	Fair	Excellent	Fair	Excellent	N/A
Provides management with progress visibility	Fair	Poor	Excellent	Fair to Excellent	Fair	Excellent	Excellent	Excellent	Excellent	N/A
Requires little manager or developer sophistication	Fair	Excellent	Poor	Poor to fair	Poor	Fair	Fair	Poor	Fair	Fair

30

Assignment #2

Due: 10/19/2011

Based on your business proposal from assignment #1, write down the system requirements (functions and features you need). The document to be handed in should consist of the following:

- Write a short description of purposes of your software products.
- List all functions of each product and give priorities for each function.
  - It should be noted that you should consider functions from different roles: regular customers, your own worker / administrator, VIP, etc. …
  - You should also give clear definition of each role, and there should be no ambiguity for each function that you listed. (e.g. How to be a VIP.)

31

Assignment #2

- For each function, identify its input and output.
- Identify core functions (minimal feature set) that is necessary to maintain your business operations.
  - Now you should have a total of four products in your planning:
    - A two-tier desktop application with all functions
    - A three-tier web application with all functions
    - A two-tier desktop application with core functions
    - A three-tier web application with core functions
- Analyze **function points** for all four products. (to be covered!)
- Based on the introduced life-cycle models, identify which life-cycle model should you use for your project. State your reasons concisely. (It should be noted that we are doing this a little bit too early, for reasons to be disclosed.)
- Estimate effort and schedule for each of these products. (be covered soon)

32