

INF 111 / CSE 121:  
Software Tools and Methods

Lecture Notes for Fall Quarter, 2007  
Michele Rousseau  
Set 7

(Some slides adapted from Susan E. Sim)

---

---

---

---

---

---

---

---

Announcements

- **Reminder: Quiz on Monday**
  - Lectures & Readings
- **Lecture vs. Lecture Slides**

Topic 7

2

---

---

---

---

---

---

---

---

Previous Lecture

- **Finished XP**

Topic 7

3

---

---

---


---

---

---

---

---



### Remember?

#### Iterations to Release Phase

- **Several Iterations before 1<sup>st</sup> Release**
- **# of Iterations determined in planning phase**
- **Each iteration takes 1-4 wks to implement**
- **Select stories wisely**
  - these enforce system architecture for the entire system
  - Customer chooses stories for each iteration
- **Functional tests created by Customer**
  - Run at the end of each iteration

**At the end of last iteration → Production**

Topic 7 4

---

---

---


---

---

---

---

---



### And Productionizing Phase?

- **End testing before release**
- **New changes may be found**
  - Decide whether to include in current release
  - Documented for later implementation  
→ Maintenance Phase
- **Iterations shortened**

Topic 7 5

---

---

---


---

---

---

---

---



### Today's Lecture

- **Testing**
- **No Silver Bullet**

Topic 7 6

---

---

---

---

---

---

---

---

# Testing

- o A basic Review

Topic 7 7

---

---

---

---

---

---

---

---

# Verification and Validation

```

    graph TD
      IR[Informal Requirements] --> FS[Formal Specification]
      FS --> SI[Software Implementation]
      V[Validation] --> IR
      V --> FS
      Ver[Verification] --> FS
      Ver --> SI
  
```

Verification: *is implementation consistent with requirements specification?*  
 Validation: *does the system meet the customer's/user's needs?*

Topic 7 8

---

---

---

---

---

---

---

---

# Software Quality: assessment by V&V

- o Software process must include verification and validation to measure product qualities
  - correctness, reliability, robustness
  - efficiency, usability, understandability
  - verifiability, maintainability
  - reusability, portability, interoperability,
  - real-time, safety, security, survivability, accuracy
- o Products can be improved by improving the process by which they are developed and assessed

Topic 7 9

---

---

---

---

---

---

---

---

## Testing Terminology

- Failure: **Incorrect or unexpected output, based on specifications**
  - Symptom of a fault
- Fault: **Invalid execution state**
  - Symptom of an error
  - May or may not produce a failure
- Error: **Defect or anomaly or “bug” in source code**
  - May or may not produce a fault

Topic 7

10

---

---

---

---

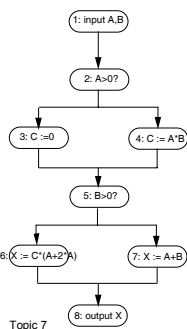
---

---

---

---

## Examples: Faults, Errors, and Failures



- Suppose node 6 should be  $X := C * (A + 2 * B)$ 
  - Failure/Fault-less error:
    - Suppose the inputs are  $(A=2, B=1)$ 
      - the executed path will be  $(1, 2, 4, 5, 7, 8)$  which will not reveal this fault because 6 is not executed
    - Suppose the inputs are  $(A=-2, B=-1)$ 
      - the executed path will be  $(1, 2, 3, 5, 6, 8)$  which will not reveal the fault because  $C = 0$
- Need to make sure proper test cases are selected
  - the definitions of C at nodes 3 and 4 both affect the use of C at node 6
    - executing path  $(1, 2, 4, 5, 6, 8)$  will reveal the failure, but only if  $B \neq 0$ 
      - (e.g. Inputs  $(A=1, B=-2)$ )

Topic 7

11

---

---

---

---

---

---

---

---

## Functional and Structural Testing

- Functional Testing
  - Test cases selected based on specification
  - Views program/component as black box
- Structural Testing
  - Test cases selected based on structure of code
  - Views program /component as white box (also called glass box testing)

Topic 7

12

---

---

---

---

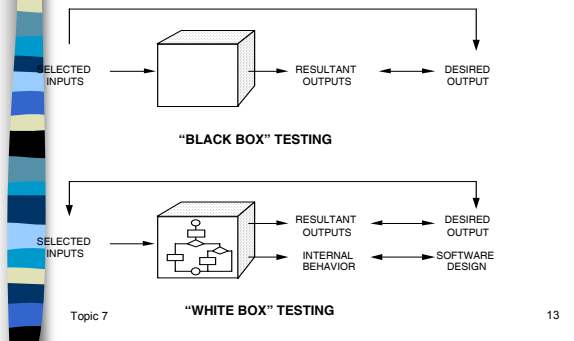
---

---

---

---

## Black Box vs. White Box Testing



---

---

---

---

---

---

---

---

## Different Levels of Testing

- **System Testing**
    - Defined at Requirements -> Run after integration testing
  - **Integration Testing**
    - Defined at Design -> Run after Unit Testing
  - **Unit Testing**
    - Defined at Implementation -> Run after Implementation of each unit
  - **Regression Testing (testing after Change)**
    - Defined throughout the process -> Run after modifications
- Topic 7 14

---

---

---

---

---

---

---

---

## Regression Testing

- **Permanent suite of test cases**
  - Saves effort creating test cases
  - Provides record of existing functionality
- **Add new test cases and delete obsolete ones when necessary**

*Ensure that changes made during maintenance do not destroy existing functionality*

Topic 7

15

---

---

---

---

---

---

---

---

## Unit Testing

- A unit test typically tests one class in the system
  - A unit test suite contains many test cases
  - Each test case typically tests one method in the system
- There can be many test cases for each method in the system
- Each test case either succeeds or fails, there is no gray area
- If a test case has an error, that is also a failure
- A test or test suite can be said to succeed to a certain percentage

Topic 7

16

---

---

---

---

---

---

---

---

## Automated Testing

- Idea: Have the computer do more of the work of running and tallying test cases
- How: Using tools, like JUnit
  - **Benefits**
    - Frequent testing
    - Regression testing
    - Adding test cases is easy
    - Concrete demonstration of effectiveness
- In XP, Test-Driven Development says to create tests first
  - **Automated Testing**

Topic 7

17

---

---

---

---

---

---

---

---

## J-Unit

- Framework for performing unit testing on Java programs
- Test cases are sub-classed from an interface
- Available as a stand-alone application and built into Eclipse
- Framework executes the test cases and records the Results
- Displays results in a GUI
- Keep the bar green to keep the code clean.”

Topic 7

18

---

---

---

---

---

---

---

---

## More J-Unit Help

- Eclipse Help
  - Help -> Help Contents -> Java Development User Guide -> Getting Started -> Basic Tutorial -> Writing and running
- JUnit tests
  - JUnit Home Page  
<http://www.junit.org>
- JUnit Primer
  - <http://www.clarkware.com/articles/JUnitPrimer.html>

Topic 7 19

---

---

---

---

---

---

---

---

## The Mythical Man-Month

- **Originally Published in 1975**
  - Fred Brooks
  - Based on Experiences From OS/360 in mid-60's
- **So why should we care?**
- **Some interesting Stats**
  - Amazon.com Sales Rank:
    - #3,201 in Books
    - #1 in Microprocessor Design
    - #3 in Systems Analysis & Design
    - #12 in Software Engineering

Topic 7 20

---

---

---

---


---

---

---

---

## Who is Fred Brooks?



- "Father of IBM OS/360"
- 1992 Computer Pioneer Award (IEEE)
- 1999 Turing award winner
- 2007 Harvard Centennial Medal
- Founded UNC-Chapel Hill CS dept

Topic 7 21

---

---

---


---

---

---

---

---



## No-Silver Bullet

“There is *no single development*, in either technology or management technique, which by itself *promises even one order-of-magnitude improvement within a decade* in productivity, in reliability, in simplicity”

Topic 7 22

---

---

---


---

---

---

---

---



## Essence & Accident

- **Essential Tasks**
  - Specifications, design & testing of conceptual constructs
- **Accidental (or incidental) Tasks**
  - Programming & Compiling

**The essential tasks are the hard part.**

Topic 7 23

---

---

---


---

---

---

---

---



## Why is building s/w difficult?

“I believe that hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation”

- It is the nature of s/w – inherent in the process
- Conceptual errors are the problem

Topic 7 24

---

---

---

---


---

---

---

---





## Four Inherent Difficulties

- Complexity
- Conformity
- Changeability
- Invisibility

Topic 7 25

---

---

---


---

---

---

---

---



## Complexity

- Very large # of states
- Scaling is up is not a repetition of the same elements in large sizes
- Elements interact in a non-linear fashion
- Complexity → Communication
- It is difficult to extend large programs without creating side effects

**Complexity makes management difficult**  
**Personnel turnover can be a disaster**

Topic 7 26

---

---

---


---

---

---

---

---



## Some of Brooks Suggestions

- **IF an OTS fits – buy it**
  - Why re-invent the wheel?
- **Requirements refinement and rapid prototyping**
  - Many iterations between client and designer
- **Grow – don't build – software**
  - Develop incrementally
- **Train great designers**

Topic 7 27

---

---

---

---

---

---

---

---

## Is XP the Silver Bullet?

Requires:

- o **Good Developers**
- o **...working well together**
- o **Sufficient Domain Knowledge**
  - Onsite Customer is knowledgeable
- o **Sufficient Technical Expertise**
  - Knowledge of tools and methods
- o **Good Communication Skills**
- o **Collocation**
  - How do you collocate 4000 programmers?

What if a method or tool is not a SB?

Topic 7 28

---

---

---

---

---

---

---

---

## Readings

- o **As mention in Wed Lecture**
  - Brooks: CH 17
  - Van Vliet: CH 4
- o **If you need more information on**
  - Software Process Models → Ch 3
  - Software Testing → Ch 13

Topic 7 29

---

---

---

---

---

---

---

---