

Objectives

- Unix Commands
- Software Development

Oct 15, 2008

Sprenkle - CS209

1

Review

- Why do we need Comparators?
- What is the benefit of using jar files?
- How do we create a jar file? Extract the contents of a jar file?
- What are the 3 preconnected streams?
 - How do we access them in Java?
- What is the regular expression that represents one of a set of characters?
- What does this command do?
 - `cp assign{8,9}.html`
- What is the Unix command to
 - Find a file?
 - Find out which files contain some text?

Oct 15, 2008

Sprenkle - CS209

2

Find Files that Contain Something: grep

- Shows all the lines in the file(s) that contain some expression
- Example: `grep System.out *.java`
 - What is the syntax for the command?
- Options:
 - `-v` if you want all the lines **without** the expression
 - `grep -v System.out *.java`
 - `-w` if you want whole words

Oct 15, 2008

Sprenkle - CS209

3

Find Files that Contain Something: grep

- Can use regular expressions and special symbols
 - `^`: begins with
 - `$`: ends with
 - `.`: wildcard
- Examples:
 - `grep c...h /usr/share/dict/words`
 - `grep -w c...h /usr/share/dict/words`

Oct 15, 2008

Sprenkle - CS209

4

More grep

- `-r`: recursively goes through directories (all files in those directories)
 - Example:
 - `grep -r System.out .`
- Special characters
 - `? \ . [] ^ $ ()`
 - Need to be escaped with `\` if used literally

Oct 15, 2008

Sprenkle - CS209

5

diff: Show differences between files

- Show lines of files that have differences
 - Displays line numbers of differences
- Example use
 - `diff file1 file2`

Oct 15, 2008

Sprenkle - CS209

6

Summary of New Commands

Command	Example Use	Meaning
find	find . -name "*.java"	Find all the files in the current directory that end in .java
grep	grep System.out *.java	Find all the lines in all the .java files in this directory that contain System.out
diff	diff file1 file1.bak	Show the lines of the files where there are differences between the files

Use `man` to learn more about the commands and its options

Oct 15, 2008

Sprenkle - CS209

7

Pipe Operator |

- Take output from previous command and "pipe it" as input to next command
- Advice on using pipe
 - Build up long commands, looking at output along the way
- Practice problems:
 - How can we determine the number of Java files we have in a directory?
 - How could we determine the number of `System.out.println`'s in a Java file?

Oct 15, 2008

Sprenkle - CS209

8

Using commands in commands: ``

- Syntax: ``command``
 - Backtick: on same key as ~
- Example: I want to check the permissions on all my shell scripts (which end in .sh)
 - Verify that they're executable by me and no one else
 - `ls -l `find . -name "*.sh``
- Note that these commands will take a little longer to execute because getting answer for "inner" command first

Oct 15, 2008

Sprenkle - CS209

9

head & tail

- Show the beginning or ending of the file (usually 10 lines)
 - `head filename`
 - `tail filename`
- `-n` for how many lines to show, where *n* is a number
 - `tail -100 filename`
- `-f` to watch a file
 - `tail -f filename`

Oct 15, 2008

Sprenkle - CS209

10

Homework: Due Friday

- Writing intermediate Unix commands
- Goal: See the power of these simple, flexible commands
- **Announcement:** we will be in room P405 on Friday

Oct 15, 2008

Sprenkle - CS209

11

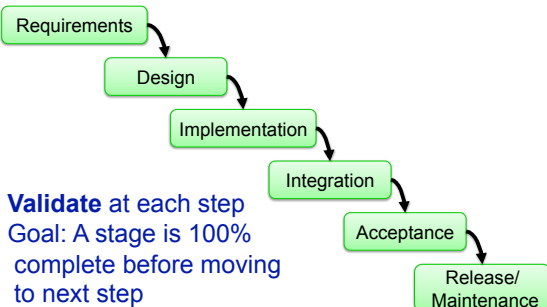
SOFTWARE LIFE CYCLE

Oct 15, 2008

Sprenkle - CS209

12

Traditional Software Engineering Process: Waterfall Model

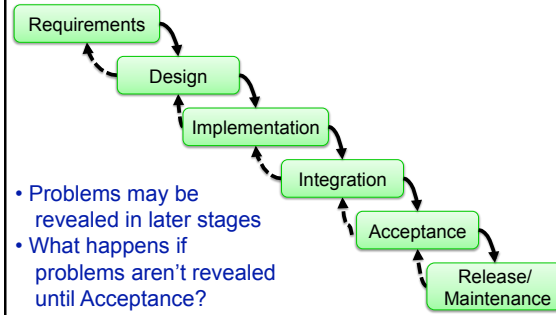


Oct 15, 2008

Sprenkle - CS209

13

Feedback in Waterfall Model

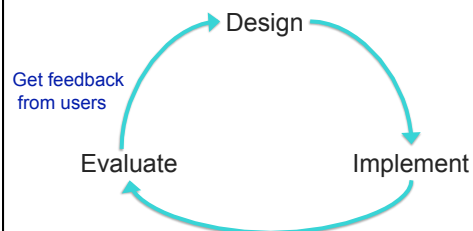


Oct 15, 2008

Sprenkle - CS209

14

Iterative Design

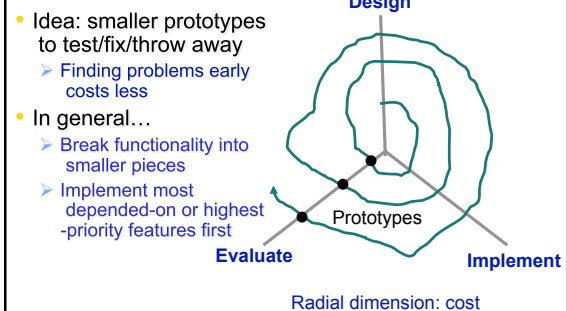


Oct 15, 2008

Sprenkle - CS209

15

Spiral Model



Oct 15, 2008

Sprenkle - CS209

16

Spiral Model Steps

- Design a {method, class, package}
- Implement the {method, class, package}
- Test the {method, class, package}
- Fix the {method, class, package}
- Deploy the {method, class, package}
- Get feedback
 - Probably will require modifications to design
- Repeat

Oct 15, 2008

Sprenkle - CS209

17

SOFTWARE TESTING PROCESS

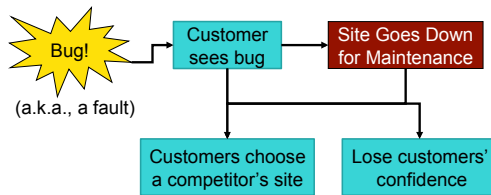
Oct 15, 2008

Sprenkle - CS209

18

Why Test Programs?

- Consider an online bookstore



Oct 15, 2008

Sprenkle - CS209

19

Microsoft Windows Vista Testing

- Beyond their internal testing ...
 - > 5 million people beta tested
 - > 60+ years of performance testing
 - > 1 Billion+ Office 2007 sessions
- Still, users found correctness, stability, robustness, and security bugs

Oct 15, 2008

Sprenkle - CS209

20

Type 1 Bugs: Compile-Time



- Syntax errors
 - > Missing semicolon, parentheses
- Compiler notifies of error
- Cheap, easy to fix

Oct 15, 2008

Sprenkle - CS209

21

Type 2 Bugs: Run-Time



- Usually logic errors
- Expensive to locate, fix

Oct 15, 2008

Sprenkle - CS209

22

Aside: Objections to "Bug" Terminology

- "Bug"
 - > Sounds like it's just an annoyance
 - Can simply swat away
 - > Minimizes potential problems
 - > Hides programmer's responsibility
- Alternative terms
 - > Defect
 - > Fault

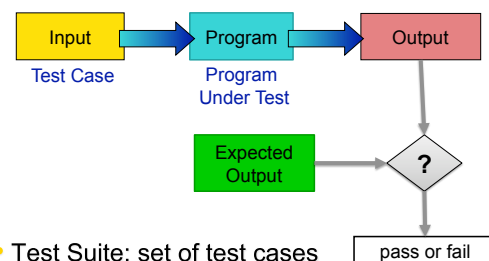


Oct 15, 2008

Sprenkle - CS209

23

Software Testing Process



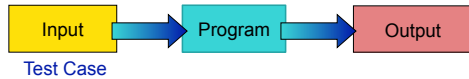
- Test Suite: set of test cases

Oct 15, 2008

Sprenkle - CS209

24

Software Testing Process



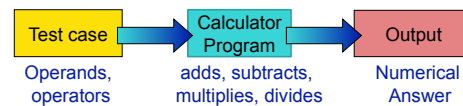
- Tester plays devil's advocate
 - **Hopes** to reveal problems in the program using "good" test cases
 - Better tester finds than a customer!
- How is **testing** different from **debugging**?

Oct 15, 2008

Sprenkle - CS209

25

How Would You Test a Calculator Program?



- What test cases/input?

Oct 15, 2008

Sprenkle - CS209

26

Example Test Cases for Calculator Program

- Basic Functionality
 - Addition
 - Subtraction
 - Multiplication
 - Division
 - Order of operations
- Invalid Input
 - Letters, not-operation characters (&,\$, ...)
- "Tricky" Cases
 - Divide by 0
 - Negative Numbers
 - Long sequences of operands, operators
 - VERY large, VERY small numbers

Oct 15, 2008

Sprenkle - CS209

27

Types of Testing

- Black-box testing
 - Test functionality (e.g., the calculator)
 - No knowledge of the code
 - Examples of testing: boundary values
- Non-functional testing
 - Performance testing
 - Usability testing (CS397: HCI)
 - Security testing
 - Internationalization, localization
- White-box testing
 - Have access to code
 - Goal: execute all code
- Acceptance testing
 - If customer accepts the product

Oct 15, 2008

Sprenkle - CS209

28

Levels of Testing

- Unit
 - Tests minimal software component
 - For us, Class-level testing
- Integration
 - Tests interfaces & interaction of classes
- System
 - Tests that completely integrated system meets requirements
- System Integration
 - Test system works with other systems, e.g., third-party systems

Oct 15, 2008

Sprenkle - CS209

29