Class 8 CS540

Gregg Vesonder

Stevens Institute of Technology

Roadmap- Class 8

- Clarifications from last class
- Log Book volunteer
- Brooks Chapters
- Elementary statistics
- Problem Solving
- Meeting skills
- Negotiation
- Management management
- Professor Bernstein's paper
- Reading: Brooks Chapters 11 & 12, BY pp248-268 Bernstein paper - on web
- Reading next class BY 4, pp346-353, B11-13

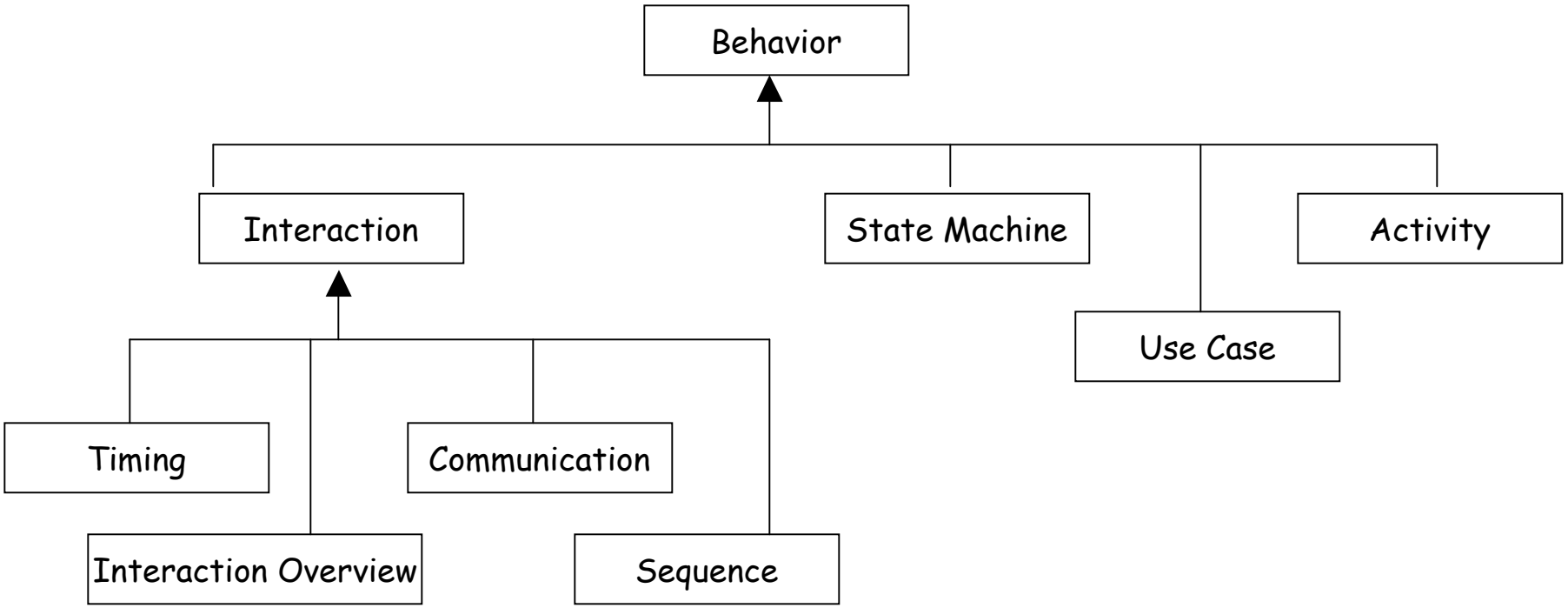
Calendar -Key Dates

- November 7th - second test
- November 21st - log books due
- December 12th - final exam

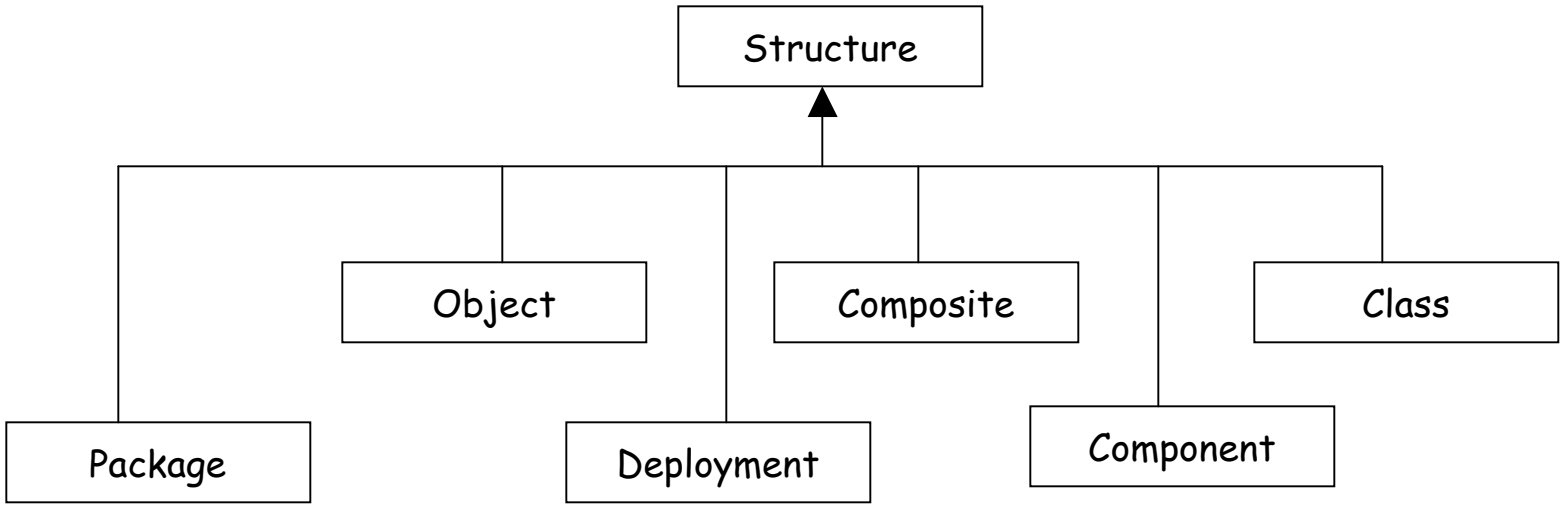
Clarifications

- Architecture - don't panic
- The UML
- Scandinavian School
- Gang of Four -- Design Patterns, Design Patterns: Elements of reusable object oriented software, Gamma, Helm, Johnson, Vlissides
- Survey: update blog and website, 6 slides on a page, slides earlier

The UML (2.0)


Behavior
Interaction
State Machine
Activity
Use Case
Timing
Communication
Interaction Overview
Sequence

The UML -2


Structure
Object
Composite
Class
Package
Deployment
Component

Scandinavian School

- User Centered Design
- Value the worker, automate, yet maintain the craft
- Really worker centered design
- Anti-dystopian view of technology

Logbook

- Your Entry
- Positive and Negative Transfer

Plan to throw one away

- Chemical engineering the pilot plant
- Software pilot, not necessarily good for the users
  - Especially with new system concept or technology
- Cosgrove - software developer delivers satisfaction of a need rather than tangible product
  - Actual need and user's perception changes as programs, built, tested and used
  - Invisibility of software exposes developers to change

Design, Build, Organize for Change

- Software: modularization, complete definition and documentation of intermodule interfaces, numbered versions with schedule and freeze date
- Common failing of development groups is too little management control
  - Today there's either extreme
- Structuring an organization for change is difficult
  - Everyone technically flexible, multiple assignments that technically broaden individuals - BREADTH & CURRENCY
  - Dual ladder vs. MTS concept
  - Manager's need technical refresher courses, technical people management training
  - Project objectives, progress and problems should be shared with all senior staff
  - Surgical team?

Brooks on Maintenance

- Maintenance cost is 40% or greater of development cost
- "Backwards J" of bugs, users exercise new aspects of system
  - Scallop effect
- Fixing a bug has a .2-.5 probability of introducing another - regression testing
- Lehman & Belady: Software systems "wear out" with maintenance entropy sets in, more modules, more modules affected with each release

Brooks- Sharp Tools

- Dated chapter - but!
- Developers have their unique tools - more like unique combinations today
- Common development and maintenance environment much more efficient - best practices
- Still a role for toolmakers
- Still target vs vehicle machines (simulations):
  - Gaming
  - Palm pilot
  - Cable boxes
  - Cell phones

Sharp Tools - cont'd

- Scheduling and sharing scarce resources
  - Test beds of PCs
  - Networks
  - Bandwidth
- Playpen - Integration - System test
- Reading plan for a large set of documents - a roadmap, worse today where CDs provide a torrent of information
- High level languages - set (psychological meaning)
- "I am convinced that interactive systems will never displace batch systems for many applications" - p.136
  - But reconsider!

Simple Statistics

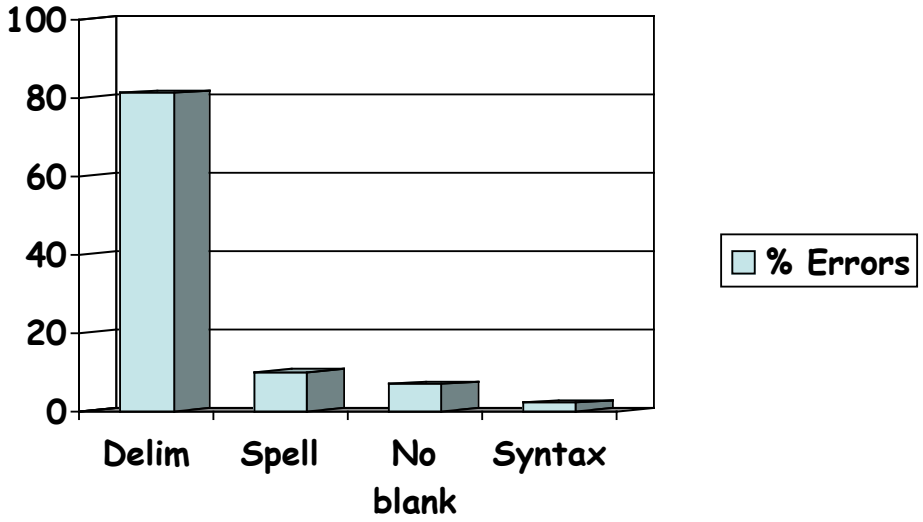
- Mean, median & mode
- Frequency distributions - pareto
- Standard deviation
- Linear regression
- Correlation
- Box plots
- Excel

Measures of Central Tendency

- Measures of central tendency are a summary of a whole distribution of events or measurements.
  - Mean (aka average) just add and divide by number of scores: 8, 5, 5, 4 ,3, 2, 2, 1 = 31/8 = 3.75
  - Median is the middle value when scores are sorted in order
    - What happens if you have an odd number of scores, even number of scores?
    - Answer to above is 3.5
  - Mode number that occurs most often - what is the mode for the series above

Frequency Distributions

- Pareto charts, illustrates 80-20 rule. Vilfredo Pareto discovered 80% of the land in Italy was owned by 20% of the people - expanded to other relationships



Measures of Dispersion

- Range - difference of largest and smallest, 8, 5, 5, 4 ,3, 2, 2, 1: range = 8-1 = 7
- Mean deviation - calculate mean get absolute difference of each number from mean (3.75) and divide by number of scores: 4.25+1.25+1.25 +.25 + .75+1.75+1.75+2.75/8 = 1.75
- Standard deviation is square root of the variance, variance = $\Sigma(x-\mu)^2/n-1$ = , std dev = 2.252
  - If I added more variability in the data 12, 5, 4, 3, 2,2,1,1, even though mean is same, standard deviation is 3.6154

Correlation

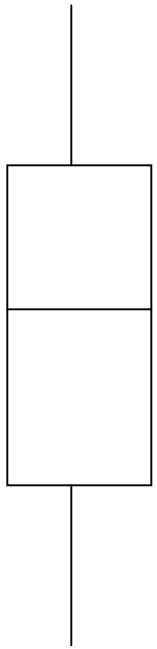
- Correlation coefficient, r = (((Σ(xy))/n)–((x/n)(y/n)))/((std dev x)(std dev y))), phew!
- Data set 1: (1, 8) (2, 15) (3, 20) (4,25) r=.996
- Data set 2: (10, 5) (20, 3) (30, 2) (40, 1) r=-.982
- "inferring causality from simple correlations is an extremely dangerous pastime!"- Underwood

Linear Regression

- Fitting a line to data, get to y=mx+b .. Beyond the scope when you get into regression, but it does form a predictive model.
- First data set y = 5.6x + 2.9
- 8.5, 14.1, 19.7, 25.3

Simple Box Plots

- A method for combining descriptive statistics



median

Running Experiments

- A control and several experimental groups
  - Within and Between person variables - ordering effects
  - Keep things constant .. Experience, intelligence, unless that is one of the factors
  - Ethics
  - Natural field studies -- productivity on different versions of the same system
  - Examples: user interfaces, different development processes, environments, …
- More during CHI time
- On to problem solving -- -what we do all the time

Problem Solving

- Some Perspectives:
  - Polya, How to Solve It:
    - Understand the problem
    - Devise a plan
    - Carry out the plan
    - Look Back
  - Hayes 6 Steps:
    - Find the problem
    - Represent the problem
    - Plan the solution
    - Carry out the plan
    - Evaluate the solution
    - Consolidate gains

Problem Solving -2

- Bransford and Stein's IDEAL
  - Identify the problem
  - Define and represent the problem
  - Explore possible strategies
  - Act on the strategies
  - Look back and evaluate the effects of your activities

Problem Solving Strategies

- Working backward from the goal
- Problem Decomposition
- Means-Ends Analysis and Backtracking
- Forward Chaining
- Analogy
- Most Problem solving theorists/researchers use a state space representation with operators

Types of Problems

- Well defined and ill defined
- Well defined:
  - Information about initial state and goal state
  - Information about permissible moves
  - Information about operator restrictions
- Ill defined:
  - Lack of clarity about what knowledge is needed and when the goal is achieved
  - Solver has to help define the problem

On Insight

- Insight - Gestalt aha experience
  - Ape experiments - 2 sticks and a banana
- Problem restructuring - 2 strings problem
  - Pendulum, breaking of set, "mechanical" way of approaching situation (new environment)
- Functional Fixedness -assume from experience that objects only have a limited number of uses
- Restructuring
  - Elaboration
  - Constraint reduction - broader view of goal
  - Re-encoding, recategorization

Brainstorming

- Have a crisp problem statement - preparation!
- Assign a scribe and post ideas
- Small and cooperative groups
  - Use delphi if necessary
- Moderator
  - No judgment
  - Every idea has its day
  - Build on other's ideas - helps listening
  - Encourage unique ideas, odd person hypothesis

On Meetings

- (derived from 3M Anatomy of a great meeting)
- Why Meet?
  - Move group actions forward by:
    - Presenting information to others
    - Collaborating -review, evaluate, discuss, problem solve and decide
  - Social Reasons:
    - Need to belong
    - Need to achieve, have an effect (success)
    - Desire to communicate, share common reality
  - Social and task needs must both be met

Meetings - Focus on Task

- Be clear about meeting's objective
- Create a solid agenda with realistic time budget for each item - should be tied to objective and items should be assigned priorities
- Prepare in advance, especially if you are running or are a key participant in the meeting

Meetings - Focus on People

- Who is invited?
  - Information providers
  - Decision makers
  - Necessary collaborators (buy-in)
  - Avoid lurkers
- Ground rules:
  - Start and end on time
  - One conversation at a time (caucus time)
  - Honor all points of view
  - Don't interrupt

Meetings and People-2

- Clarify the decision process to all members
  - Autocratic-leader makes decision
  - Democratic-each participant votes, majority rules
  - Consensus - everyone agrees to move forward
- Roles:
  - Facilitator
  - Recorder - careful here
  - Leader
  - Participant

Before and After Meeting

- Before:
  - Advance agenda
  - Participants
  - Time and place (directions, reach numbers)
  - Preparation of materials
  - Pre meeting materials
  - A/V equipment available
  - Special needs?
- After, crisp meeting notes and followup:
  - Decisions
  - Action items
  - Open issues
  - Parking lot

Heuristic Meeting Methods

- The dry board marker as Talking Stick
  - Usual suspects
  - Designate a leader
  - Solicit others
- Prepare, prepare, prepare
- Roles, goals and strategy for a meeting
- Gather consensus before the meeting

Negotiation

- Compromise
- Collaboration - win-win
- Competition - win-lose
- Accommodation - lose-win
- Withdrawal/Avoidance lose-lose

WinWin

- Situations where everyone wins when negotiations succeed, everyone loses if negotiations fail - disputes in which everyone can win
- Focus on 2 party disputes
- Voluntary choice, fair-division procedures permit disputants to make changes, sometimes with the help of a moderator

WinWin Procedures

- Strict and balanced alteration
  - Taking turns
  - Going 1st issue
- Divide and choose
  - I cut, you choose
  - Division is the toughest
- Adjusted winner
  - Allocate points to each item until they total 100, each party then given items that they placed more points than opponent, tally and then try to equalize

WinWin Fairness Properties

- Proportionality - each party thinks they are getting at least half of the total value
- Envy-Freeness - no party is willing to give up a portion it receives for someone else's portion
- Equitability - relative happiness at end, must be equivalent, in effect both exceeded 50% by the same amount (point allocation procedure is an indicator)
- Efficiency - it is efficient if no other allocation for a party is better for some party without being worse for another - has to be linked with other properties, since all/none is efficient

So what about SE?

- Requirements are a negotiation among parties about "goods."
- EasyWinWin builds on this and uses collaborative technology
- Stakeholders use win-win negotiate techniques to collect, elaborate and assign priorities to requirements
- Resolve issues with mutually satisfying and fair agreements

EasyWinWin

- Review and expand negotiation topics based on a domain taxonomy of software requirements
- Brainstorm stakeholder interests - electronically list their desired
- Converge on win conditions - crisp requirements based on ideas in brainstorming
- Capture glossary of terms
- Assign priorities to win conditions by business importance and ease of realization

Win Conditions

	Business Importance High	**Business Importance Low**
Feasibility Easy	Low hanging fruit	Maybe later
Feasibility Difficult	Important with hurdles	Forget them

EasyWinWin - 2

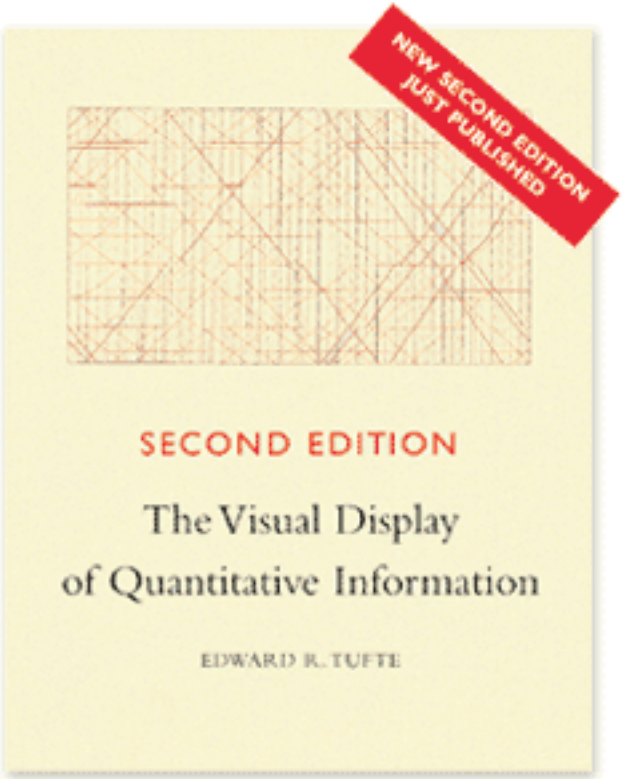
- Reveal issues and constraints - discuss items with low consensus at the last step
- Identify issues and options on the current list (hopefully discussion clarified)
- Negotiate agreements - win conditions w/o issues are declared agreements - negotiate the rest

- Groupware helps this process

Managing Management

- No surprises
- Preparation
- Draft emails in editor, let sit, send later if at all
  - Emails are unpredictable and emoticons do not make up for emphasis and body language
- Clear and simple with supporting view graphs
  - Tell them what you will tell them, tell them, tell them what you told them
  - Socialization
  - Red, yellow & green - assume copies will not have color
- Informal meeting - prepare a list topics and facts
  - Sender receiver issues
  - Observers/Interpreters are helpful
  - Roles - be explicit
- Log impressions of meetings that night
- Tufte!

Tufte


NEW SECOND EDITION
JUST PUBLISHED
SECOND EDITION
The Visual Display
of Quantitative Information
EDWARD R. TUFTE

Bernstein - Fault tolerance

- Acknowledges that software has errors
- Goal: Mean Time To Failure, large and Mean Time To Repair, small -- making software available to users in the face of software errors
- In the 90's fault tolerance gave way to high availability, high reliability RAID based systems
  - Dominant vendors convinced folks this was enough
- "Software system development is frequently focused on performance and functional technical requirements and does not address the need for reliability or trustworthiness"
- Robustness deals with expected problems, fault tolerance with unexpected problems
- Trustworthy software is stable, does not crash with minor flaws and will shut down in an orderly fashion with major flaws

Early Approaches

- Initially mimicked hardware fault tolerance techniques
- N version concept in software mimics N-way redundant hardware in software
  - Each module is built with N different implementations, then each module submits its output to a "decider" that determines the correct answer.
  - Works if programs do not share similar failure modes
- Recovery Blocks - transactions are monitored and execution can be "rolled back"
  - Various things can be done to rolled back transaction including dropping it.
- But hardware fault tolerance is predicated on manufacturing defects

Software and Manufacturing

- Software faults are usually products of design shortcomings not manufacturing faults
  - Software manufacturing is configuration, …, i.e., reproduction
- Parnas' Undesired Events paper
- As Sha states:
  - Complexity begets faults
  - Some faults are easy to find and fix, others are Heisenbugs - a bug whose presence is affected by the act of observing it (intermittent symptoms, debuggers change situation)
  - Cannot spring for unlimited testing

Fault Tolerance and Reliability

- Fault Tolerance is the ability of a software system to avoid executing a fault that causes a system to fail
- $R(t) = e^{-\lambda t}$ λ is proportional to software complexity, *C*, and inversely proportional to effort, E.
- $R(t) = e^{-kCt/E}$ and assuming duration t =1 and scaling constant k = 1, then $R(E,C) = e^{-C/E}$, the higher the complexity, the more effort.

Reliability

- Can be improved by:
  - Investing in tools
  - Simplifying
  - Do more inspections and testing during development
    - Code reading, critique
    - Code reviews, check against requirements
    - Code inspection

Software Stability Key to Simplicity

- Instability due to:
  - Buffer usage that increases and diminishes system performance
  - Computations that can't be completed before new data arrives
  - Round off errors that propagate
  - Flawed algorithms
  - Memory leaks
  - Register overflows
  - Thrashing of files and networks
  - Broadcast storms

Constraints on Design

- Control free interfaces (low coupling)
- Software error recovery - exception handling throughout
  - Fault tolerance provides services consistent with spec after detecting fault - unknown problems
  - Exception handling contains and eliminates a known problem
- Recovery blocks
- Limit language features used
- Limit module size and initialize memory -- 300 to 500 instructions, smaller too many interfaces, larger too big for one person
- REUSE MODULES w/o CHANGE!

Moving to Simplicity - Refactoring

- Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure
  - Improving the design of the code after it has been rewritten
  - Make it work, make it work right, make it work better
  - Ultimate goal is to subtract code when adding a feature

Increasing Reliability through Rejuvenation

- Rejuvenation - at periodic intervals, application is terminated and restarted at a known, clean state
- These are rejuvenation periods where everything is reinitialized
- A bug avoidance technique
- Usually measured in days and weeks - prevents memory leaks from getting the best of the system
- (systematic rebooting)

Effort Factors

- Hire good people, keep them
- Don't confuse vocational training with education
- Maximize effectiveness of staff, large changes (>100 instructions) and small changes (<5) are error prone
- Heuristics:
  - Design defensively, leave in debug mode
  - Stress test
  - Explain all anomalies uncovered, else provisional release
  - Hold arch reviews
- Classical engineering technique - stress a system until it breaks and guarantee it for much less than break point

Last Words

- Preventing a fault from becoming a failure - remember testing vocabulary
- It is fault tolerant iff:
  - Program can compute acceptable result even if it suffers from incorrect logic
  - Produces acceptable result even with corrupted data
- Robustness deals with expected problems, fault tolerance with unexpected problems
- Critical software paths: correct, incorrect but acceptable behavior, hazardous

Thought Problems

- There are 3 orcs and 3 hobbits on the left bank of a river, and they all have to get over to the right bank. The only way across is by using a boat that can carry up to two creatures. The problem is, if there are ever more orcs than hobbits on one side, the orcs will eat the hobbits. Hobbits will not eat orcs under any circumstances. Of course at least one creature must be in the boat as it goes from side to side.
- Is an effort recognizing fault tolerance necessary in your organization? If so, what would you select and how would you implement it?

So Far

- Software Process Models, Software Project Planning (woosh!), Requirements, Estimation, Risk Analysis, Multics case study, Architecture Reviews, Questionnaire Design
- Software Quality Assurance
- Configuration Management and Testing
- Architecture and Design
- This Time: Software Engineering skills: Problem Solving, meeting, stat,
- Next Time: OO

In Context

- Spanned a variety of techniques that support software engineering, a breather in intensity
- Discussed fault tolerance - acknowledging design shortcomings
- On to OO next week

References

- The Win-Win Solution by S.J. Brams and A.D. Taylor
- L. Bernstein "Software fault tolerance forestalls crashes: To err is human; to forgive is fault tolerant." Advances in Computers Volume 58, Highly Dependable Software, M. Zelkowitz(Ed), Academic Press, 2003.
- http://sunset.usc.edu.research/WINWIN/EasyWinWin