# Class 3 CS540

Gregg Vesonder

Stevens Institute of Technology

# Roadmap- Class 3

- Clarifications from last class
- Journal entry
- Brooks - Mythical Man Month
- Estimation
- Risk Analysis again
- Reading this week: chapter 6 in BY, chapter 2 in Brooks
- Readings next week, chapter 2, pp 39-48 in BY, chapters 3-5 in Brooks
- Test in 2 weeks

# Class 2 Clarifications

- Full time students, the class and the logbook
- On testing
- Supplemental readings
- Glossed over glossaries
- The value of story boards
- Variables
    - Irregular, cannot be varied, e.g., experience of users,
    - Goal variables, project based not problem based, e.g., quality, time, costs
    - Control variables, what you can control -software tools

# Calendar -Key Dates

- October 3rd - first test
- Oct 11th - class on Tuesday not Monday!
- November 7th - second test
- November 21st - log books due
- December 12th - final exam

# Log Book

- Transitions
- Volunteer?
- The Questionnaire - next week

# Brooks Chapter 2

- Projects fail usually for lack of calendar time
  - Estimation is optimistic, last bug syndrome
  - Costs vary by staff months, progress does not
  - Sequential nature of some tasks
  - Communication is key difference, farm workers vs. developers
  - Each new worker must be trained by experienced staff
  - Communication rule of thumb n(n-1)/2
- Testing most mis-scheduled part
- Brooks heuristics:
  - 1/3 planning
  - 1/6 coding
  - 1/4 component and early system test
  - 1/4 system test, whole system

# Brooks, Chapter 2 (cont'd)

- "Gutless estimating … urging of patron"
- Solutions:
  - Industry wide need to publish data
  - "Take no small slips"
  - Trim the task
- Brook's Law: "Adding manpower to a late software project makes it later"
- Caveats:
  - #of months depends on sequential constraints
  - #of staff depends on # of independent subtasks (and ability to match talent to task)

# Metrics

- Project: metrics used specifically but not solely by management to control current projects and provide feedback for future projects
- Technical (Individual?): used by an engineers to improve their performance

# Technical/Individual Metrics

- Halstead
- McCabe
- Fan-in/Fan-out

# Halstead -"software science"

- Stresses syntactic units rather than LOC
- Model components:
  - Operators - actions: +, -, *, /, if-then-else,…
  - Operands - data: variables and constants
  - 4 basic entities (used in a bunch of equations)
    - $n_1$ - # of different operators
    - $n_2$ - # of different operands
    - $N_1$ - total occurrences of operators
    - $N_2$ - total occurrences of operands
  - Length of Program for Halstead: $N = N_1 + N_2$

# Halstead uses

- Simple to calculate, no in-depth examination of structure
- Measure of overall quality of programs - simplicity/bloat criteria
- In conjunction with others, helpful in maintenance and initial programming
- Substantial literature
- At surface level requires completed code so not good in estimation but with certain assumptions N can be calculated early on
- Does not account for complexity of interfaces

# McCabe's Cyclomatic Complexity

- Based on a directed graph showing control flow of program thereby showing the number of independent paths in the program
- Cyclomatic Complexity, CV= e - n + p + 1 where:
  - e = # of edges
  - n = # of nodes
  - p = # of connected components (1 for main program 1 for each procedure)
- 10 should be upper limit of complexity for a component according to McCabe

# McCabe Uses

- Great for testing because it uncovers all linearly independent paths
- Does not add more complexity to nested loops and in general does not consider context
- Unlike Halstead does take into account control flow complexity, but not data therefore, often used together
- Useful for individual developer feedback and during maintenance

# Fan In/Fan Out

- Measures interaction - basically, # of modules that call a given module and number of modules called by a given module.
- High degrees of fan-in/fan-out are undesirable
- Typical equations: [LOC|Cyclomatic complexity]*(#fan-in*#fan-out)$^2$
- Takes into account data driven programs but underestimates (of course) complexity for programs/modules with little interaction

# Testing Your Powers of Estimation- Bentley's Quiz

This little quiz is designed to help you evaluate your proficency as a number guesser. For each question, fill in upper and lower bounds that, *in your opinion*, give you a ninety percent chance of including the true value; try not to make your ranges too narrow or too wide. I estimate that it should take you between five and ten minutes. Please make a good faith effort.

[_____, _____] January 1, 2000, population of the United States in millions.

[_____, _____] The year of Napoleon's birth.

[_____, _____] Length of the Mississippi-Missouri river in miles.

[_____, _____] Maximum takeoff weight in pounds of a Boeing 747 airliner.

[_____, _____] Seconds for a radio signal to travel from the earth to the moon.

[_____, _____] Latitude of London.

[_____, _____] Minutes for a space shuttle to orbit the earth.

[_____, _____] Length in feet between the towers of the Golden Gate Bridge.

[_____, _____] Number of signers of the Declaration of Independence.

[_____, _____] Number of bones in the adult human body.

# Requirements Process

1. ICED-T analysis: Intuitive, Consistent, Efficient, Durable, Thoughtful
2. Simplified Quality Functional Deployment
3. Compute effort (Function points one way, next class)
4. Estimate staff and development time
5. Revise requirements to meet above (if exceeds cost, time)
6. Redo 1-4
7. Replan with GANTT chart (optional)
8. Review MOV to see if it is worth it

# Project Metric Estimates

- Used for cost prediction and project monitoring
- Predicting size of system should get easier as project continues - stable specs and fewer tasks remaining
- Two steps:
  - Estimate size
  - Estimate effort and cost from sizing (coding necessary to fulfill requirements). Effort includes resources + calendar time + staff budget constraints
- Cost Prediction usually based on either expected effort or elapsed time

# Cost Estimation

- Cost estimation falls into these categories (Kitchenham, 1994):
  - Expert
  - Analogy
  - Decomposition
  - PERT Models - Delphi Method
  - Mathematical Models - COCOMO, Rayleigh Curve, Function Points
  - Parkinson's Law- work expands to fill time available
  - Pricing to win- cost and effort is whatever customer has to spend
  - Top-Down/Bottom-UP

# Software Estimation

- Initially KLOC and then divide by LOC/month figures (3KLOC/yr, average programmer productivity rate -- Futrell, et.al.)
- Some data was collected by IBM - regression analyses
- Basic flavor is estimate adjusted by unique characteristics of the project which is assessed by 10 to n factors that weight the final estimate

# Software Estimation -2

- In addition to LOC, function points, feature points, number of bubbles on Data Flow Diagram, objects, classes, number of pages of user documentation, …

# Easy?

- As we saw in our exercise -- estimation is not easy even when we have a lot of information:
  - 15% of all software projects fail to meet their goals, overruns of 100-200% are common.
- "When performance does not meet the estimate, there are two possible causes: poor performance or poor estimates. In the software world, we have ample evidence that our estimates stink, but virtually no evidence that people in general don't work hard enough or intelligently enough." -- Tom DeMarco

# Capers Jones Table

| Language | Ratio-Source:Executable |
|---|---|
| Assembler | 1:1 |
| Macro-Assembler | 1:1.5 |
| C | 1:2.5 |
| ALGOL | 1:3 |
| COBOL | 1:3 |
| FORTRAN | 1:3 |
| Pascal | 1:3.5 |
| RPG | 1:4 |
| PL1 | 1:4 |
| MODULA-2 | 1:4.5 |
| Ada | 1:5 |
| PROLOG | 1:5 |
| LISP | 1:5 |
| FORTH | 1:5 |
| BASIC | 1:5 |
| LOGO | 1:6 |
| 4th-GLs | 1:8 |
| APL | 1:9 |
| OBJECTIVE-C | 1:12 |
| SMALLTALK | 1:15 |
| Query-Languages | 1:20 |
| Spreadsheets | 1:50 |

# SLOC!

- Source Lines Of Code:
  - Make sure single statement, not two separated by semicolon
    - Syntax issue
  - All delivered, executable statements (OA&M)
  - Count data definitions only once
  - No Comments
  - Count all instances of calls, subroutines, …
  - No industry standards
  - This can be toyed with if you base compensation on it!

# COCOMO

- COnstructive COst MOdel
- Based on Boehm's analysis of a database of 63 projects - models based on regression analysis of these systems
- Linked to classic waterfall model
- Effort is number of Source Lines of Code (SLOC) expressed in thousands of delivered source instructions (KDSI) - excludes comments and unmodified utility software
- Model has 3 versions and considers 3 types of systems:
  - Organic - simple business systems
  - Embedded - avionics
  - Semi-detached - management inventory systems

# COCOMO System Types

| | SIZE | INNOVATION | DEADLINE | CONSTRAINTS |
|---|---|---|---|---|
| Organic | Small | Little | Not tight | Stable |
| Semi-Detached | Medium | Medium | Medium | Medium |
| Embedded | Large | Greater | Tight | Complex hdw/customer interfaces |

# COCOMO Formula

Effort in staff months $=b*KDLOC^{c}$

| | b | c |
|---|---|---|
| organic | 2.4 | 1.05 |
| semi-detached | 3.0 | 1.12 |
| embedded | 3.6 | 1.20 |

# A Retrospective on the Regression Models

- They came to similar conclusions:
  - Time:
    - Watson-Felix $T = 2.5E^{0.35}$
    - COCOMO(organic) $T = 2.5E^{0.38}$
    - Putnam $T = 2.4E^{0.33}$
  - Effort:
    - Halstead $E = 0.7\ KLOC^{1.50}$
    - Boehm $E = 2.4\ KLOC^{1.05}$
    - Watson-Felix $E = 5.2\ KLOC^{0.91}$

# Intermediate *COCOMO*

- Adds 15 attributes of the product that has to be rated on a six point scale from Very Low to Extra High
- There are 4 categories of attributes: product, computer, personnel and project.
- The ratings are reflected in P of the equation

$$\text{Effort in staff months} = (b * KDLOC^{C}) * P$$

# Intermediate COCOMO attributes

- PRODUCT:
  - RELY - required reliability
  - DATA- data bytes per DSI (smaller db)
  - CPLX - code complexity (VH= real time)
- COMPUTER:
  - TIME - execution time, % used
  - STOR - storage requirements, % used
  - VIRT - changes made to hdw and OS
  - TURN- Dev turnaround time, batch vs interactive
- PERSONNEL
  - ACAP - analyst capability, skills
  - PCAP - programmer capability
  - AEXP- applications experience
  - LEXP - language experience
  - VEXP- virtual machine experience
- PROJECT
  - MODP - Modern Development Practices
  - TOOL - use of sfw tools
  - SCED - amount of schedule compression

# Intermediate COCOMO Attributes

http://www.cs.unc.edu/~stotts/COMP145/cocomo6.gif

## Project Characteristics Table

Cost adjustments for computing the EAF (Effort Adjustment Factor)

| | v.low | low | nominal | high | v.high | ex.high |
|---|---|---|---|---|---|---|
| **product attributes** | | | | | | |
| required software reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | |
| database size | | 0.94 | 1.00 | 1.08 | 1.16 | |
| product complexity | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| **computer attributes** | | | | | | |
| execution time constraints | | | 1.00 | 1.11 | 1.30 | 1.66 |
| main storage constraints | | | 1.00 | 1.06 | 1.21 | 1.56 |
| virtual machine volitility | 0.87 | 1.00 | 1.15 | 1.30 | | |
| computer turnaround time | | 0.87 | 1.00 | 1.07 | 1.15 | |
| **personnel attributes** | | | | | | |
| analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | |
| applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | |
| programmer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | |
| virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 | | |
| programming language experience | 1.14 | 1.07 | 1.00 | 0.95 | | |
| **project attributes** | | | | | | |
| use of modern programming practices | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 | |
| use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | |
| required development schedule | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 | |

# COCOMO II

- Mapped to life cycle, three phases:
  - Application Composition Model -based on counting Object points where objects are screens, reports and 3GL (C, FORTRAN,BASIC) modules weighted by difficulty. Object points are easier to determine at an earlier time.
  - Early Design Model uses unadjusted function points which are converted to SLOC based on language. Rather than Function Point cost structures, uses its own cost drivers which are related to third model, the Post Architecture Model. Cost drivers are: product reliability and complexity, required reuse, platform difficulty, personnel experience, personnel capability, facilities and schedule. Rated on 7 point scale.

# COCOMO II-2

- Post Architecture Model is most detailed model. Differs from original COCOMO in set of cost drivers, and range of values to parameters. New cost drivers are:
  - Documentation needs
  - Personnel continuity
  - Required reusability
  - Multi-site development
  - (-) computer turnaround time
  - (-) use of modern programming practices

# The COCOMO Family

- COCOMO II - estimates cost, effort and schedule of a perspective project
- COCOTS - estimates cost, effort and schedule associated with using commercial off-the-shelf components (COTS) in a software development project. (research project)
- COQUALMO - explores relationship between cost, schedule and quality
- CORADMO - estimates cost of developing software using rapid application development techniques.
- COPROMO - predicts the most cost effective allocation of investment resources in new technologies intended to improve productivity
- COPSEMO - estimates cost of developing software as distributed over development activity stage
- COSYSMO - estimates the system engineering tasks in software intensive projects
- http://sunset.usc.edu/research/cocomosuite/suite_main.html

# Function Points

- A break from KLOC - important because estimates can be based on design data
- Based on counting data structures
- Great for business apps or any that are data dependent, not as good for apps that have a heavy algorithmic component, e.g., compilers
- Five factors in the Function point Analysis Model:
  - I # of input types that are user inputs changing data structures.
  - O # of output types
  - E # of inquiry types, input controlling execution. E.g., menu selection
  - L # of logical internal files, internal data used by system such as index files
  - F # of interfaces data output or shared with another app

# Function Point Calculations

- Unadjusted Function Points
  - UFP= 4I + 5O + 4E + 10L + 7F
- The constants can be altered if we know more about the data
  - Each input type has a number of data element types (attributes) and refers to other file types. As these increase, the complexity level increases, resulting in changes to the constants

# Complexity Table

| TYPE: | SIMPLE | AVERAGE | COMPLEX |
|---|---|---|---|
| INPUT (I) | 3 | 4 | 6 |
| OUTPUT(O) | 4 | 5 | 7 |
| INQUIRY(E) | 3 | 4 | 6 |
| LOG INT (L) | 7 | 10 | 15 |
| INTERFACES (F) | 5 | 7 | 10 |

# Adjusted Function Points

- Application characteristics are considered
- Considers 14 characteristics (environmental factors)on a 6 point scale (0-5)
- Total Degree of Influence (DI) is sum of scores.
- DI is converted to a technical complexity factor (TCF)
  - TCF = 0.65 + 0.01DI
- Adjusted Function Point is computed by
  - FP = UFP X TCF
- For a given language there is a direct mapping from Function Points to LOC
- COUNTING FUNCTION POINTS IS DIFFICULT!
- Feature points - extension of function points to deal with expanded set of applications, such as embedded and real time systems.
- And Object points- at a higher level than function points, one object point to each unique class or object

# Application Characteristics for FPA

- Data Communications
- Distributed Functions
- Performance
- Heavily used configuration
- Transaction rate
- Online Data Entry
- End-user efficiency
- Online update
- Complex processing
- Reusability
- Installation ease
- Operational ease
- Multiple sites
- Facilitate change

# Initial Conversion

http://www.qsm.com/FPGearing.html

| Language | Median SLOC/function point |
|---|---|
| C | 104 |
| C++ | 53 |
| HTML | 42 |
| JAVA | 59 |
| Perl | 60** |
| J2EE | 50 |
| Visual Basic | 42 |

# Function Points

- Pros:
  - Consistent, language independent measure of size
  - Easier to understand by client
  - Insight by simple modeling
  - More difficult to fudge
  - Facilitate management decisions -- feature creep
- Cons:
  - Labor intensive
  - Skilled counters, extensive training required
  - Inexperience results in inconsistent results
  - Weighted to file manipulation and transactions
  - Historical data improves it
  - Systematic error introduced by single person, multiple raters advised

# Wideband Delphi Method

- A group of experts can give a better estimate
- The Wideband Delphi Method:
  - Coordinator provides each expert with spec
  - Experts discuss estimates in initial group meeting
  - Each expert gives estimate in interval format: most likely value and an upper and lower bound
  - Coordinator prepares summary report indicating group and individual estimates
  - Group iterates until consensus

# Delphi Calculations

Group estimate is average of the weighted individual estimates.

Individual Estimate =(lower bound + 4*most likely + upper bound)/6

Variance = (upper bound - lower bound)/6

# Requirements Process

1. ICED-T analysis: Intuitive, Consistent, Efficient, Durable, Thoughtful
2. Simplified Quality Functional Deployment
3. Compute effort (Function points one way, next class)
4. Estimate staff and development time
5. Revise requirements to meet above (if exceeds cost, time)
6. Redo 1-4
7. Replan with GANTT chart (optional)
8. Review MOV to see if it is worth it

# Heuristics to do Better Estimates

- Decompose Work Breakdown Structure to lowest possible level
- Review assumptions with all stakeholders
- Do your homework - past organizational experience
- Retain contact with Developers
- Update estimates and track new projections (and warn)
- Use multiple methods
- Reuse makes it easier (and more difficult)

# Heuristics to Cope with Estimates

- (from S. McConnell, "How to defend and unpopular schedule, IEEE Software, 13(3), 1996.)
- More developers, if it is early enough
- Higher output developers (order of magnitude difference)
- More admin support
- Increase degree of developer support (faster machines, …)
- Eliminate red tape (50% issue)
- Devote full time end user to project
- Increase level of exec sponsorship to break new ground (new tools, techniques, training)
- Set a schedule goal date but agree to reassess after detailed design
- Use broad estimation ranges rather than single point estimates

# Heuristics to Reengineer Requirements

- Move some of the desired functionality into version 2
- Deliver product in stages 0.6, ..
- Cut features
- Polish some features less
- Relax the detailed requirements for some features.

# Software Risk Management

- (much of this adapted from http://www.eas.asu.edu/~riskmgmt/intro.html and the SEI)
- Risk is defined as exposure to harm or loss, not only probability but effect as well.
- NOT RISK AVOIDANCE
- The SEI (and others) phases of risk analysis are:
  - Identify
  - Analyze
  - Plan
  - Track
  - Control

COMMUNICATION

# Identify Risk

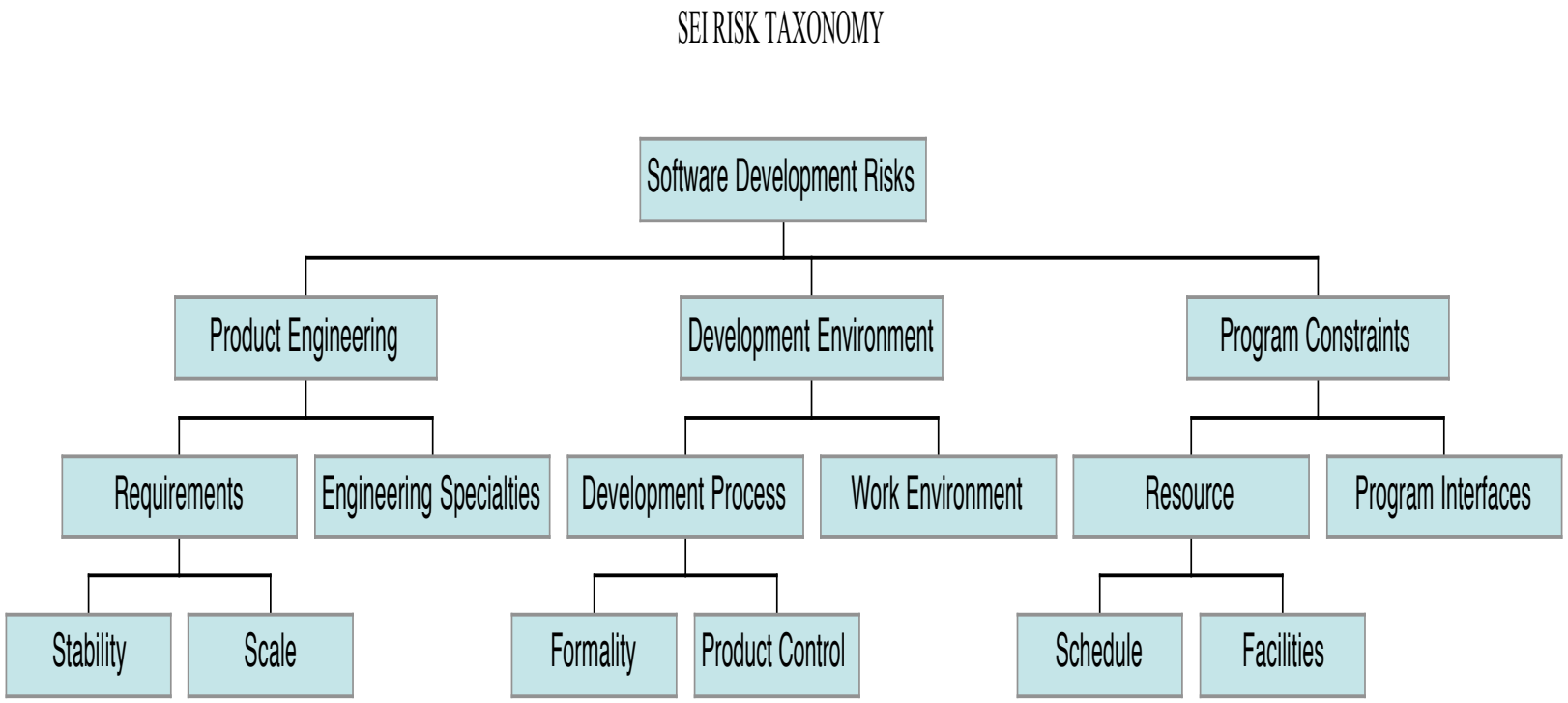
- Risks can be known, unknown and unknowable or known knowns, known unknowns (apply to this project), unknown unknowns :-)!
- SEI method of risk identification (and management) based on following assumptions:
  - Risks are often known by tech staff but poorly communicated
  - A repeatable method is necessary for risk management
  - Must cover all areas
  - Attitude must be non-judgmental and supportive so that controversial views can be heard
  - Success or failure of the project can not be based solely on risk assessment

# SEI Development Risk Taxonomy

- 3 major categories:
  - Product engineering - what is being developed
  - Development Environment - how it is being developed
  - Program constraints - contractual, organization and operational factors

# Taxonomy Expanded

(adapted from CMU/SEI-93-TR-6)



A. Product Engineering
   1. Requirements
      a. Stability
      b. Completeness
      c. Clarity
      d. Validity
      e. Feasibility
      f. Precedent
      g. Scale
   2. Design
      a. Functionality
      b. Difficulty
      c. Interfaces
      d. Performance
      e. Testability
      f. Hardware Constraints
      g. Non-Developmental Software
   3. Code and Unit Test
      a. Feasibility
      b. Testing
      c. Coding/Implement-ation
   4. Integration and Test
      a. Environment
      b. Product
      c. System
   5. Engineering Specialties
      a. Maintainability
      b. Reliability
      c. Safety
      d. Security
      e. Human Factors
      f. Specifications

B. Development Environment
   1. Development Process
      a. Formality
      b. Suitability
      c. Process Control
      d. Familiarity
      e. Product Control
   2. Development System
      a. Capacity
      b. Suitability
      c. Usability
      d. Familiarity
      e. Reliability
      f. System Support
      g. Deliverability
   3. Management Process
      a. Planning
      b. Project Organization
      c. Management Experience
      d. Program Interfaces
   4. Management Methods
      a. Monitoring
      b. Personnel Management
      c. Quality Assurance
      d. Configuration Management
   5. Work Environment
      a. Quality Attitude
      b. Cooperation
      c. Communication
      d. Morale

C. Program Constraints
   1. Resources
      a. Schedule
      b. Staff
      c. Budget
      d. Facilities
   2. Contract
      a. Type of Contract
      b. Restrictions
      c. Dependencies
   3. Program Interfaces
      a. Customer
      b. Associate Contractors
      c. Subcontractors
      d. Prime Contractor
      e. Corporate Management
      f. Vendors
      g. Politics

**Figure A-1 Taxonomy of Software Development Risks**

# Analyze Risk

- Probability of risk, USAF Handbook categories are very low, low, medium, high and very high
- Impact of risk, USAF Handbook categories are negligible, marginal, critical and catastrophic
- Risks are rarely independent
- A matrix is used to determine overall risk for different categories (e.g., effort, performance, schedule, cost, support)

# *Sample* Impact/Probability Matrix
## (used to calculate overall risk)

| Impact/Probability | V. High | High | Medium | Low | V. Low |
|---|---|---|---|---|---|
| Catastrophic | H | H | M | M | L |
| Critical | H | H | M | L | 0 |
| Marginal | M | M | L | 0 | 0 |
| Negligible | M | L | L | 0 | 0 |

# Plan for the Risks

- What can you do:
  - Mitigate impact by developing a contingency plan should risk occur and identify the trigger to initiate the contingency plan
  - Avoid the risk by changing something
  - Accept the risks and the consequences if it occurs
  - Study the risk further so that you can decide on one of the above
- In addition:
  - Specify why risk is important
  - What info is need to track status of risk
  - Who is responsible for Risk Management and what is the cost

# Risk Tracking and Control

- Track like everything else in the project monitoring status of risks and actions taken to address them. Appropriate risk metrics should be in place.
- Control, the risk process should be in place in the beginning, deviations to the plan should be corrected, triggering events should be handled and the process should be assessed for effectiveness.

# Thought Problems

- What mechanism would you establish to provide better estimates of time and effort for software projects in your group, division and company
- You've just been promoted and now head a new team (with all new hires, none that you know) that has been assigned to do a high risk project - what is your plan of attack for risk management?

# So Far

- Software Process Models
- Software Project Planning
- Requirements
- Estimation
- Risk Analysis
- Next Time:
  - Multics case study
  - Architecture Reviews
  - Questionnaire Design

# Lecture Resources

- Futrell, R.T., Shafer, D.F. and Shafer, L.I. Quality software project management. Prentice Hall, 2002
- Others embedded in text
- Quiz Answers:
  - January 1, 2000, population of the United States is 272.5 million.
  - Napoleon was born in 1769.
  - The Mississippi-Missouri river is 3,710 miles long.
  - Maximum takeoff weight of a B747-400 airliner is 875,000 pounds.
  - A radio signal travels from the earth to the moon in 1.29 seconds.
  - Latitude of London is about 51.5 degrees.
  - A space shuttle orbits the earth in about 91 minutes.
  - 4200 feet between the towers of the Golden Gate Bridge.
  - 56 signers of the Declaration of Independence.
  - 206 bones in the adult human body.