

Risk Analysis

CS577 Fall 2002

Outline

- ✓ Risk Management Definitions and Principles
- ✓ Risk Assessment and Control
 - Relations to CS577a Projects

“If You Don’t Actively Attack the Risks,



The Risks Will Actively Attack You.”

-Tom Gilb



Defining Risk

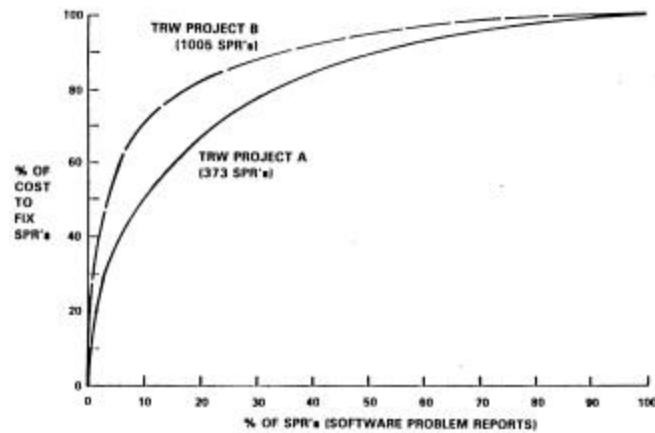
**“Risk”: Possibility of loss or injury
-Webster**

**Risk Exposure =
(Probability of unsatisfactory outcome) X
(Loss if unsatisfactory outcome)**

Importance of Risk Management

- ✓ Addresses Complex Software Systems
- ✓ Focuses Projects on Critical Risk Items
- ✓ Provides Techniques for Handling Risk Items
- ✓ Reduces Software Costs by Reducing Rework
 - Usually 40-50% of software costs

Rework Costs Concentrated in a Few High-Risk Items



If Risk Management is so important, why don't people do it?

Unwillingness to admit risks exist

- Leaves impression that you don't know exactly what you're doing
- Leaves impression that your bosses, customers don't know exactly what they're doing
- "Success-orientation"

Tendency to postpone the hard parts

- Maybe they'll go away
- Maybe they'll get easier, once we do the easy parts

Costs money and time up front

When do people do Risk Management?

After they've been burned in similar situation

- Pain-avoidance
- Convincing evidence of consequences

When everybody involved is convinced that risks exist, but that it's still worth going forward

- Everyone is a winner → Realistic expectations

When they've learned how to do it well

- Techniques not well-known, but can be learned

Components of "Satisfactory Outcome:" Stakeholder Win Condition Satisfaction

- ✓ Customer, Developer: Budget, Schedule
- ✓ User: Functionality, Performance, Reliability, Usability
- ✓ Maintainer: Modifiability, Portability
- ✓ Product Line Manager: Reusability

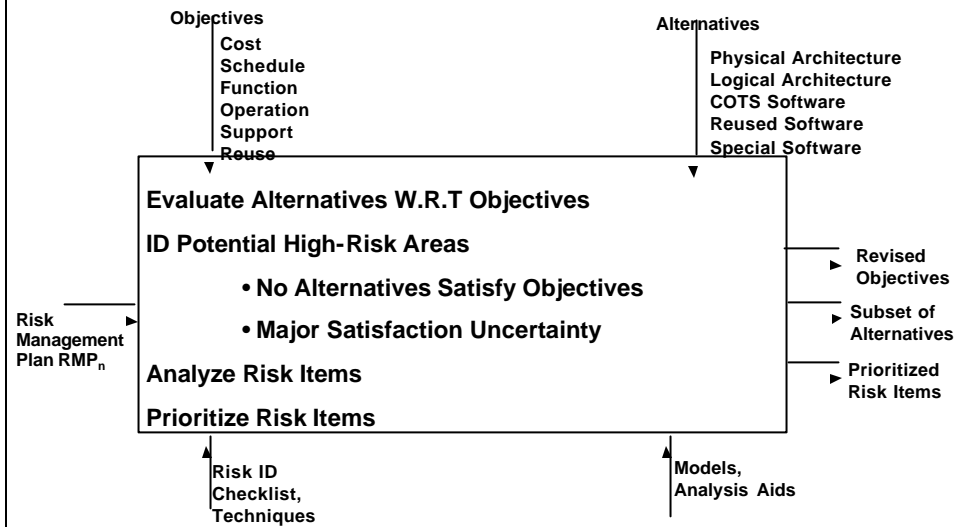
- ✓ Many Counterexamples:
 - Lee, "The Day the Phones Stopped," 1991
 - Gibbs, "Software's Chronic Crisis," 1994
 - Neumann, "Computer Related Risks," 1995

Outline

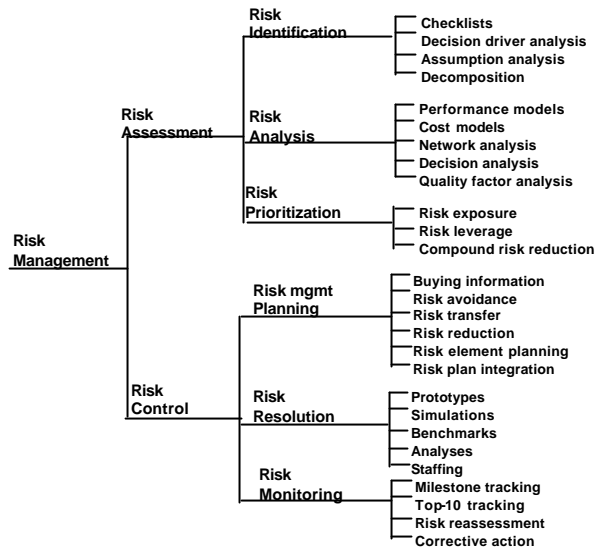
✓ Risk Management Definitions and Principles

➔ ✓ Risk Assessment and Control
– Relations to CS577a Projects

Risk Assessment Role in Each Life-Cycle Phase



Software Risk Management



Risk Identification Techniques

- ✓ Risk-item checklists
- ✓ Decision driver analysis
 - Comparison with experience
 - Win-lose, lose-lose situations
- ✓ Decomposition
 - Pareto 80 – 20 phenomena
 - Task dependencies
 - Murphy's law
 - Uncertainty areas

Top 10 Risk Items: 1989 and 1995

1989

1. Personal shortfalls
2. Schedules and budgets
3. Wrong software functions
4. Wrong user interface
5. Gold plating
6. Requirements changes
7. Externally-furnished components
8. Externally-performed tasks
9. Real-time performance
10. Straining computer science

1995

1. Personnel shortfalls
2. Schedules, budgets, process
3. COTS, external components
4. Requirements mismatch
5. User interface mismatch
6. Architecture, performance, quality
7. Requirements changes
8. Legacy software
9. Externally-performed tasks
10. Straining computer science

Example Risk-item Checklist: Staffing

- ✓ Will you project really get all the best people?
- ✓ Are there critical skills for which nobody is identified?
- ✓ Are there pressures to staff with available warm bodies?
- ✓ Are there pressures to overstaff in the early phases?
- ✓ Are the key project people compatible?
- ✓ Do they have realistic expectations about their project job?
- ✓ Do their strengths match their assignment?
- ✓ Are they committed full-time?
- ✓ Are their task prerequisites (training, clearances, etc.) Satisfied?

Candidate CS577 Risk Items

- ✓ Personnel: commitment; compatibility; ease of communication; skills (management, Web/Java, Perl, CGI, data compression, ...)
- ✓ Schedule: project scope; IOC content; critical-path items (COTS, platforms, reviews, ...)
- ✓ COTS: see next charts; multi-COTS
- ✓ Rqts, UI: mismatch to Library user needs
- ✓ Performance: #bits; #bits/sec; overhead sources

COTS and External Component Risks

- ✓ COTS risks: immaturity; inexperience; COTS incompatibility with application, platform, other COTS; controllability
- ✓ Non-commercial off-the shelf components: reuse libraries, government, universities, etc.
 - Qualification testing; benchmarking; inspections; reference checking; compatibility analysis

Advantages of COTS and Custom Software

COTS Integration

- ✓ Predictable license costs
- ✓ Broadly used, mature technology
- ✓ Available now
- ✓ Dedicated support organization
- ✓ Hardware/software independence
- ✓ Rich in functionality
- ✓ Frequent upgrades

Custom Development

- ✓ Complete freedom
- ✓ Smaller, often simpler
- ✓ Often better performance
- ✓ Control of development/enhancement
- ✓ Control of reliability tradeoffs

Disadvantages of COTS and Custom Software

COTS Integration

- ✓ Up front license fees
- ✓ Recurring maintenance fees
- ✓ Dependency on vendor
- ✓ Efficiency sacrifices
- ✓ Functionality constraints
- ✓ Integration not always trivial
- ✓ No control over upgrades/ maintenance
- ✓ Unnecessary features consume extra resources
- ✓ Reliability often unknown/ inadequate
- ✓ Scale difficult to change
- ✓ Incompatibilities among vendors
- ✓ Licensing and intellectual property issues

Custom Development

- ✓ Development expensive/unpredictable
- ✓ Availability date unpredictable
- ✓ Maintenance expensive
- ✓ Portability often expensive
- ✓ Drains expert resources

The Top Ten Software Risk Items

Risk Item	Risk Management Techniques
1. Personnel Shortfalls	Staffing with top talent; key personnel agreements; incentives; team-building; training; tailoring process to skill mix; peer reviews
2. Unrealistic schedules and budgets	Business case analysis; design to cost; incremental development; software reuse; requirements descoping; adding more budget and schedule
3. COTS; external components	Qualification testing; benchmarking; prototyping; reference checking; compatibility analysis; vendor analysis; evolution support analysis
4. Requirements mismatch; gold plating	Stakeholder win-win negotiation; business case analysis; mission analysis; ops-concept formulation; user surveys; prototyping; early users' manual; design/develop to cost
5. User interface mismatch	Prototyping; scenarios; user characterization (functionality, style, workload)

The Top Ten Software Risk Items (Concluded)

6. Architecture, performance, quality	Architecture tradeoff analysis and review boards; simulation; benchmarking; modeling; prototyping; instrumentation; tuning
7. Requirements changes	High change threshold; information hiding; incremental development (defer changes to later increments)
8. Legacy software	Design recovery; phaseout options analysis; wrappers/mediators; restructuring
9. Externally-performed tasks	Reference checking; pre-award audits; award-fee contracts; competitive design or prototyping; team-building
10. Straining Computer Science capabilities	Technical analysis; cost-benefit analysis; prototyping; reference checking

Risk Prioritization

- **Risk exposure**
- **Risk leverage**
 - **Betting analogy**
 - **Adjective calibration**
 - **Delphi/group techniques**
 - **Compound risk reduction**
- **Prioritization examples**

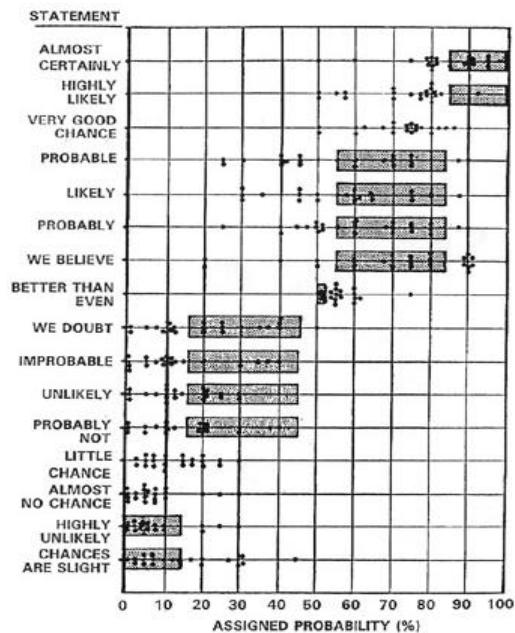
Risk Probability Assessment

- **Calculate probabilities, utilities**
 - **Hard to do in general**
- **Betting analogy**
 - **Define “satisfactory” level**
 - **Establish a personally meaningful amount of money, say, \$100**
 - **Determine how much money you would be willing to risk in betting on satisfactory outcome**

Risk Probability Assessment: Example

- Establish proposition
 - Using java will not cause us to slip our schedule
- Establish betting odds
 - No schedule slip: you win \$100
 - Schedule slip: you lose \$500
- Determine willingness to bet
 - Willing: low risk
 - Unwilling: high risk
 - Not sure:
 - risk due to uncertainty → buy information

WHAT UNCERTAINTY STATEMENTS MEAN TO DIFFERENT READERS



Watch Out For Compound Risks

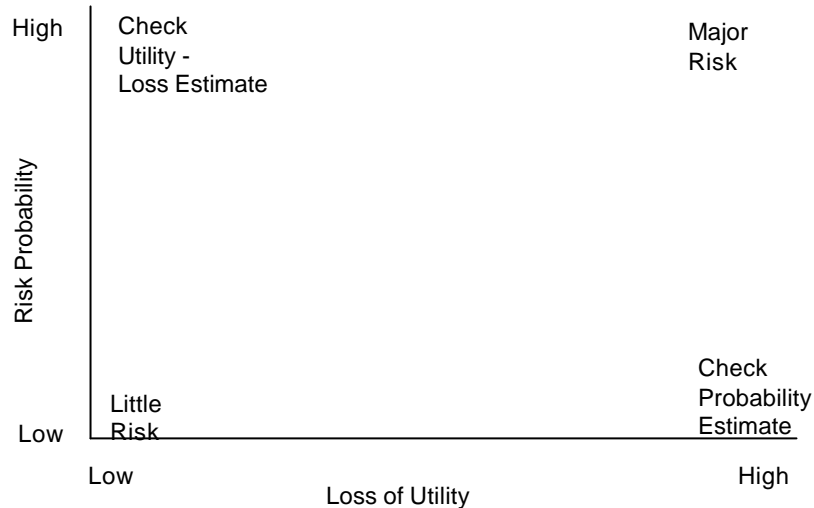
- Pushing technology on more than one front
- Pushing technology with key staff shortages
- Vague user requirements with ambitious schedule
- Untried hardware with ambitious schedule
- Unstable interfaces with untried subcontractor

➔ Reduce to non-compound risks if possible

- Otherwise, devote extra attention to compound- risk containment

Prioritizing Risks: Risk Exposure

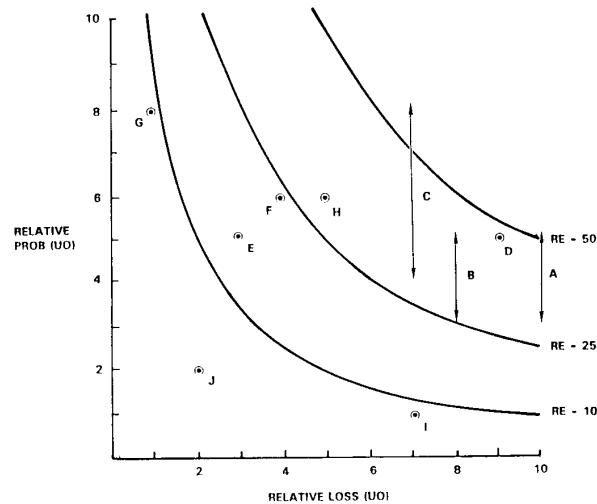
Risk Exposure - (Probability) (Loss of Utility)



Risk Exposure Factors (Satellite Experiment Software)

<u>Unsatisfactory Outcome (UO)</u>	<u>Prob (UO)</u>	<u>Loss (UO)</u>	<u>Risk Exposure</u>
A. S/ W error kills experiment	3 - 5	10	30 - 50
B. S/ W error loses key data	3 - 5	8	24 - 40
C. Fault tolerance features cause unacceptable performance	4 - 8	7	28 - 56
D. Monitoring software reports unsafe condition as safe	5	9	45
E. Monitoring software reports safe condition as unsafe	5	3	15
F. Hardware delay causes schedule overrun	6	4	24
G. Data reduction software errors cause extra work	8	1	8
H. Poor user interface causes inefficient operation	6	5	30
I. Processor memory insufficient	1	7	7
J. DBMS software loses derived data	2	2	4

Risk Exposure Factors and Contours: Satellite Experiment Software



Risk Reduction Leverage (RRL)

$$\text{RRL} = \frac{\text{RE BEFORE} - \text{RE AFTER}}{\text{RISK REDUCTION COST}}$$

· Spacecraft Example

	LONG DURATION TEST	FAILURE MODE TESTS
LOSS (UO)	\$20M	\$20M
PROB (UO) _B	0.2	0.2
RE _B	\$4M	\$4M
PROB (UO) _A	0.05	0.07
RE _A	\$1M	\$1.4M
COST	\$2M	\$0.26M
RRL	$\frac{4-1}{2} = 1.5$	$\frac{4-1.4}{0.26} = 10$