



# **Software Risk Management**

**CS 577a, 510**

**Barry Boehm, USC**



# Outline

- **Risk Management Definitions and Principles**
- **Risk Assessment and Control**
  - Relations to CS577a Projects
- **Recommended Implementation Approach**
- **Conclusions**



# “If You Don’t Actively Attack the Risks,





# The Risks Will Actively Attack You.”

-Tom Gilb



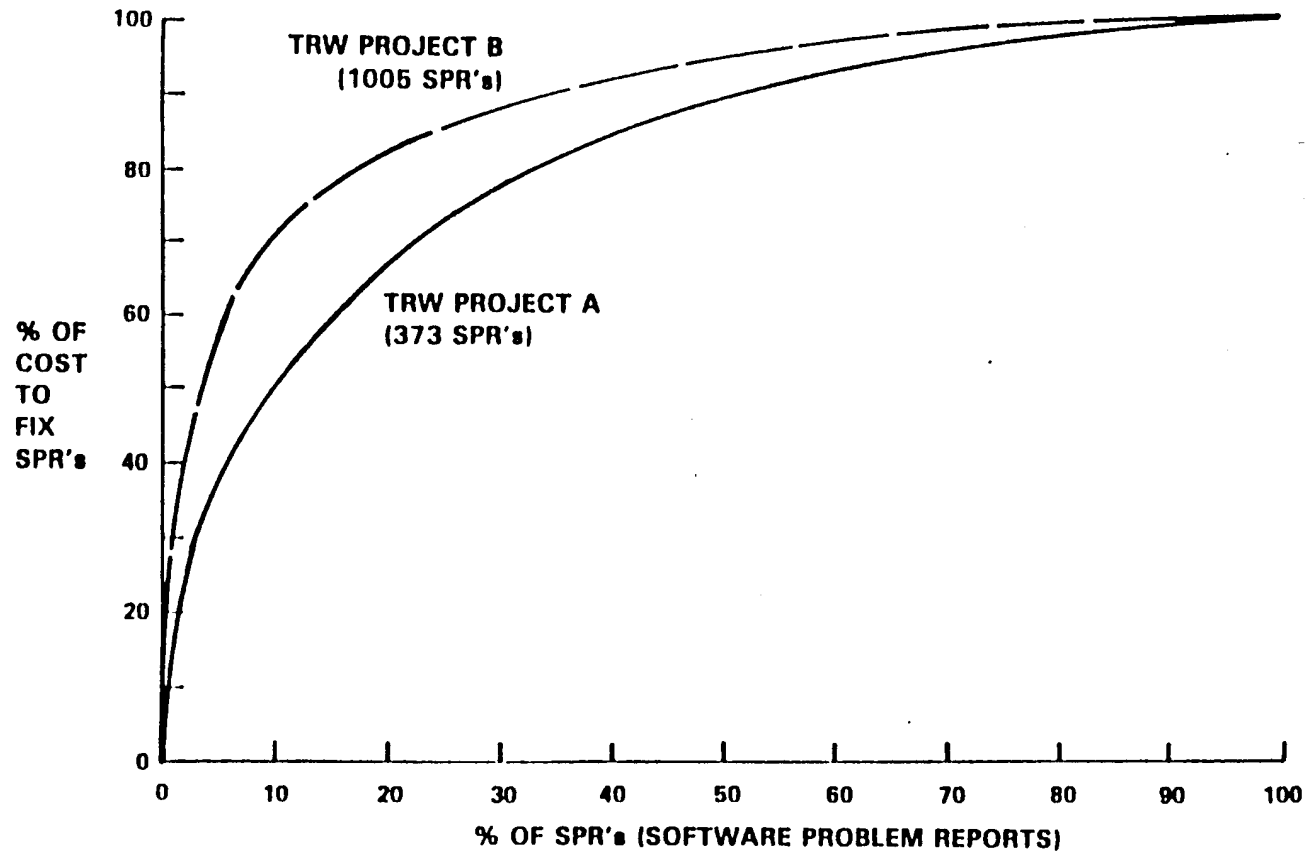


# Importance of Risk Management

- **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



# Defining Risk

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

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



## **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

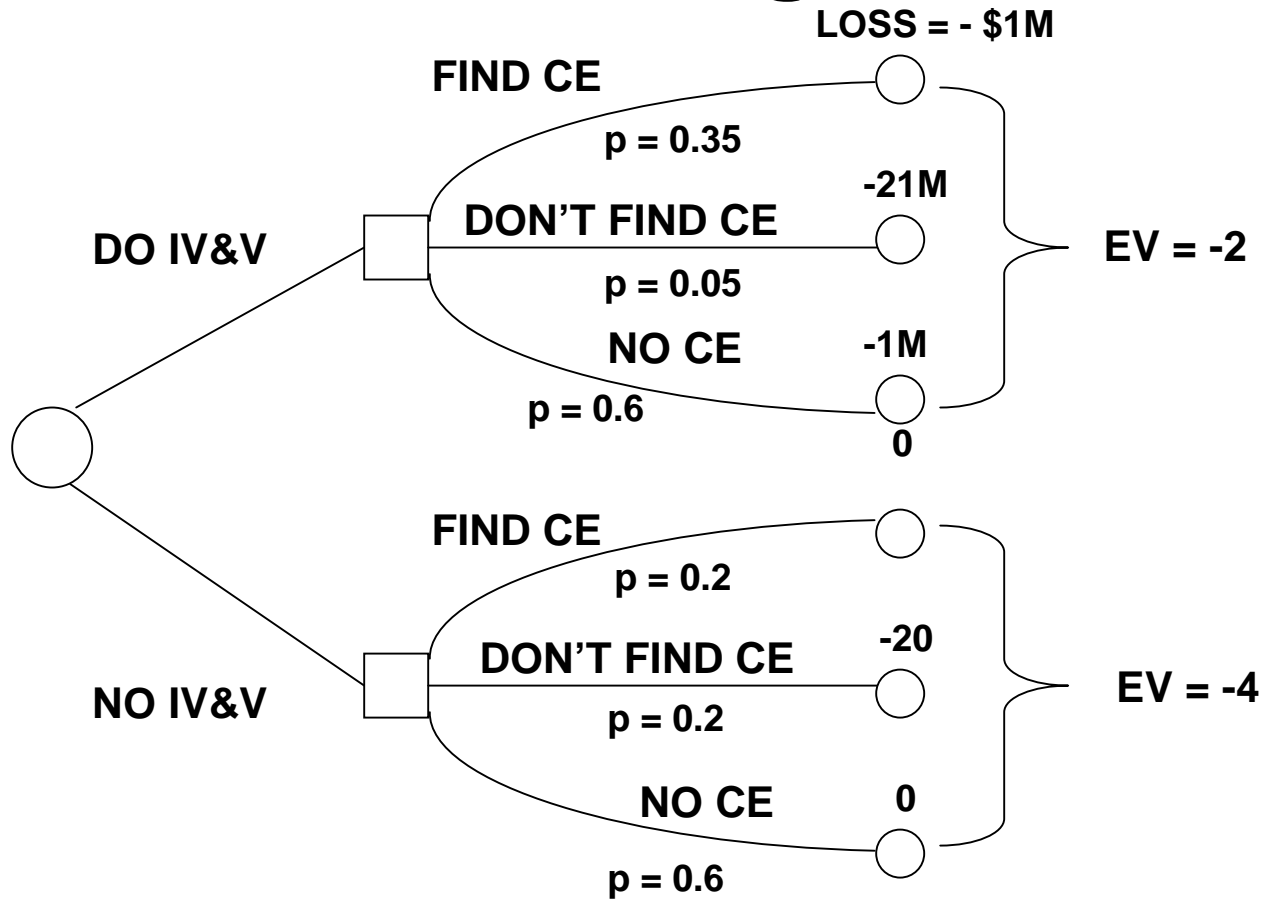


# A Typical Software Risk Situation

- **Software Controls Spaceborne Experiment**
- **PROB (Software has critical error) = 0.4**
- **Loss from Critical Error Occurring = \$20M**
- **If Do Independent V&V:**
  - **PROB (Critical Error Remains) = 0.05**
  - **Added Cost to Software Project = \$1M**
- **If No Independent V&V:**
  - **PROB (Critical Error Remains) = 0.2**
  - **Added Cost to Software Project = \$0**



# The Fundamental Risk Analysis Paradigm



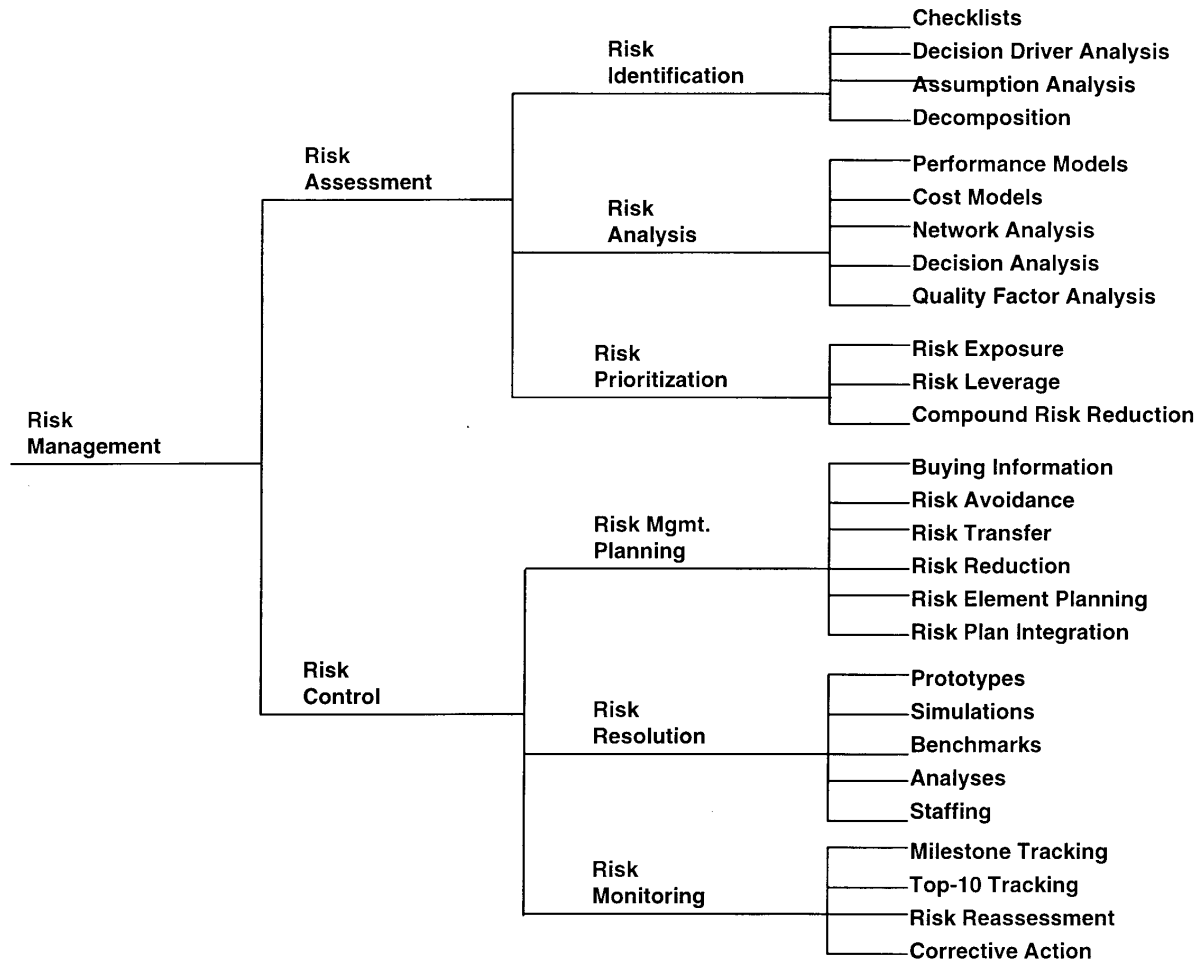


# Outline

- Risk Management Definitions and Principles
- ➔• Risk Assessment and Control
  - Relations to CS577a Projects
- Recommended Implementation Approach
- Conclusions



# Software Risk Management





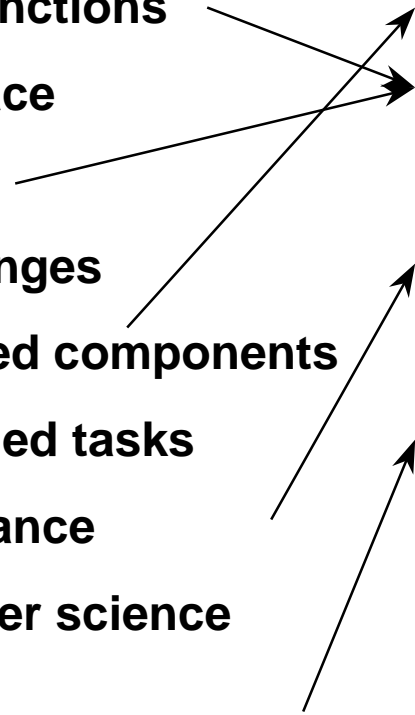
# Top 10 Risk Items: 1989 and 1995

**1989**

**1995**

1. Personnel 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

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





# 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

<b>Risk Item</b>	<b>Risk Management Techniques</b>
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)

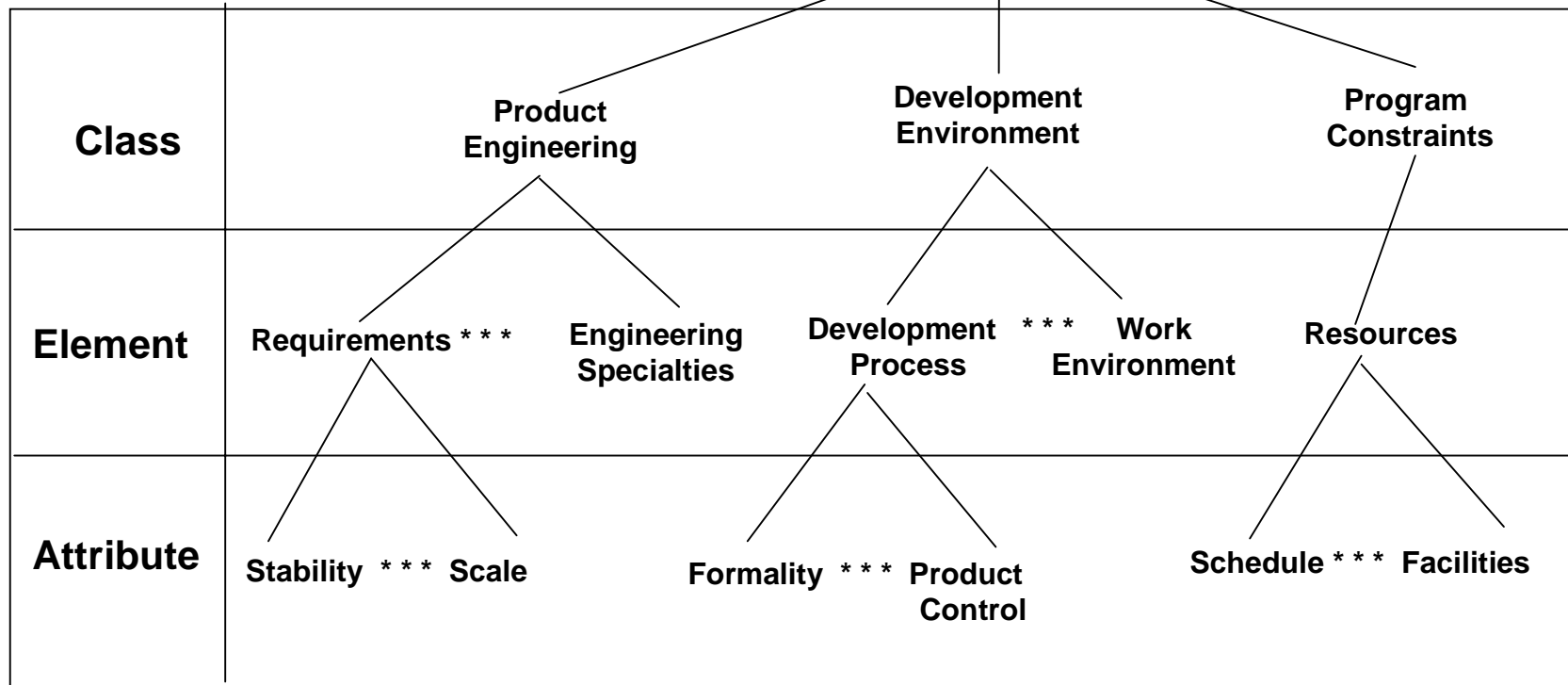
- |  |   |
|--|---|
| <b>6. Architecture, performance, quality</b>       | <b>Architecture tradeoff analysis and review boards; simulation; benchmarking; modeling; prototyping; instrumentation; tuning</b> |
| <b>7. Requirements changes</b>                     | <b>High change threshold; information hiding;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



# SEI Risk Taxonomy for Risk Identification

## -CMU/SEI-93-TR-06

### Software Development Risk





# SEI Risk Taxonomy: Sample Questions

Class → **A. Product Engineering**

Element → **2. Design**

Attribute → **c. Performance**

Starter Question → [22] Has a performance analysis been done?  
(Yes) (22.a) What is your confidence in the performance analysis?

Follow-up Questions → (Yes) (22.b) Do you have a model to track performance through design and implementation?

Starter Question → [23] Are there any problems with performance?

Cues →

- throughput
- scheduling asynchronous real-time events
- real-time response
- recovery time lines
- database response, contention, or access



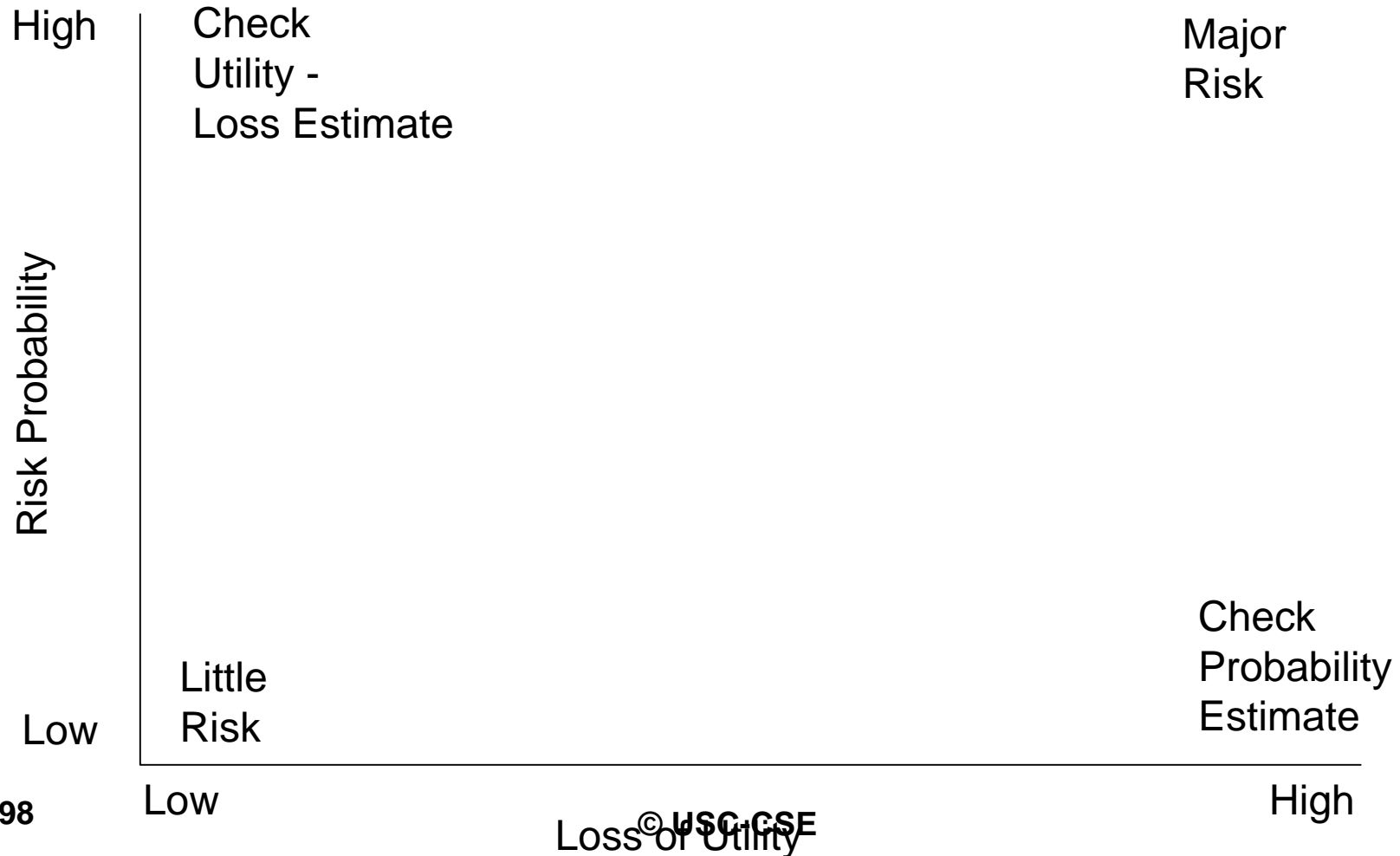
# SEI Risk Taxonomy: Lessons Learned

- **Taxonomy-based questionnaire works**
- **Group of 5 to interview is maximum**
- **Important to have peer group in interviews**
- **2-1/2 hour interview is sufficient**
- **Interview protocol is important**
  - **allow risk discussions to proceed naturally**
  - **use non-judgmental questions and probes**



# Prioritizing Risks: Risk Exposure

Risk Exposure - (Probability) (Loss of Utility)



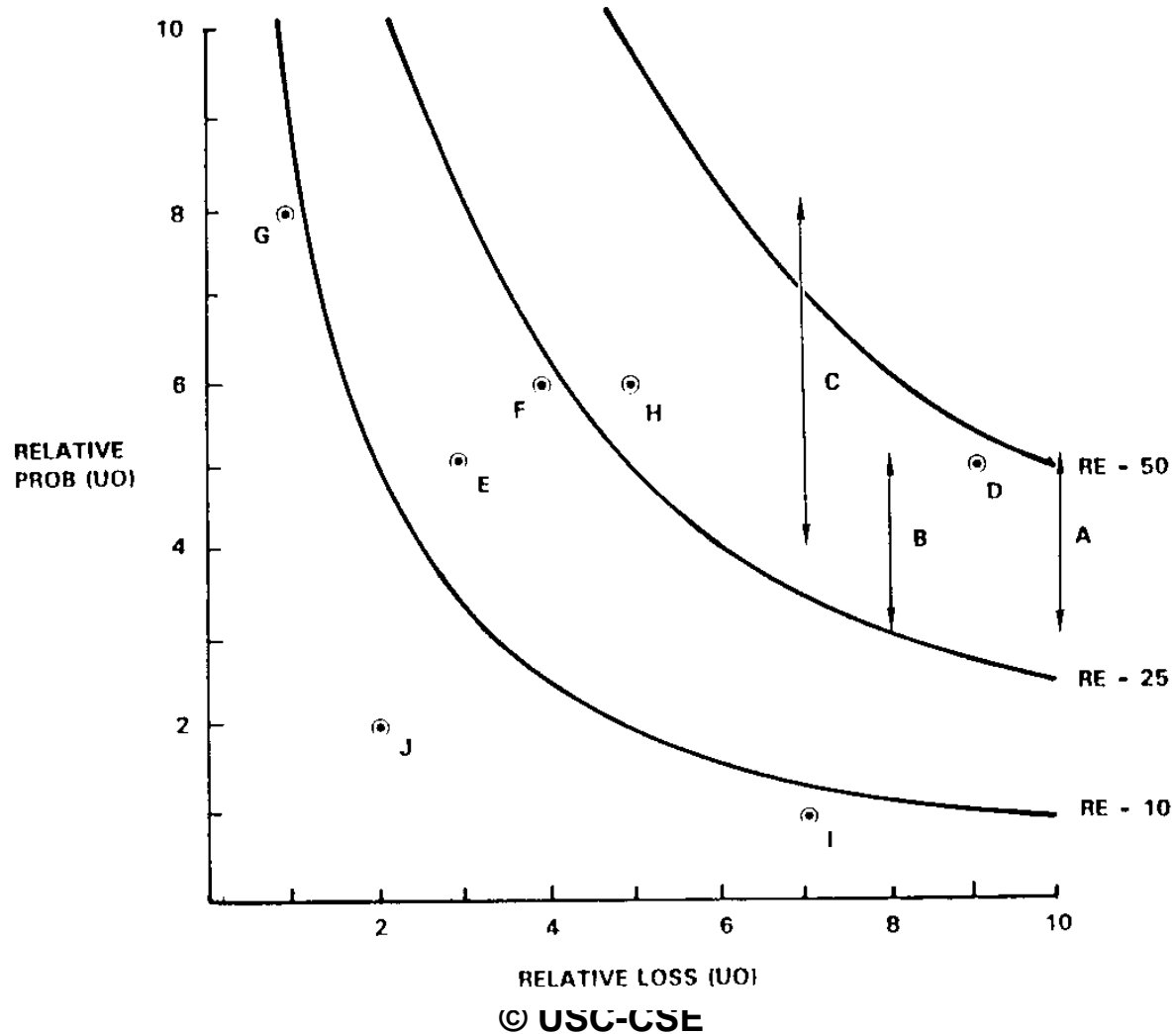


# Risk Exposure Factors (Satellite Experiment Software)

<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)

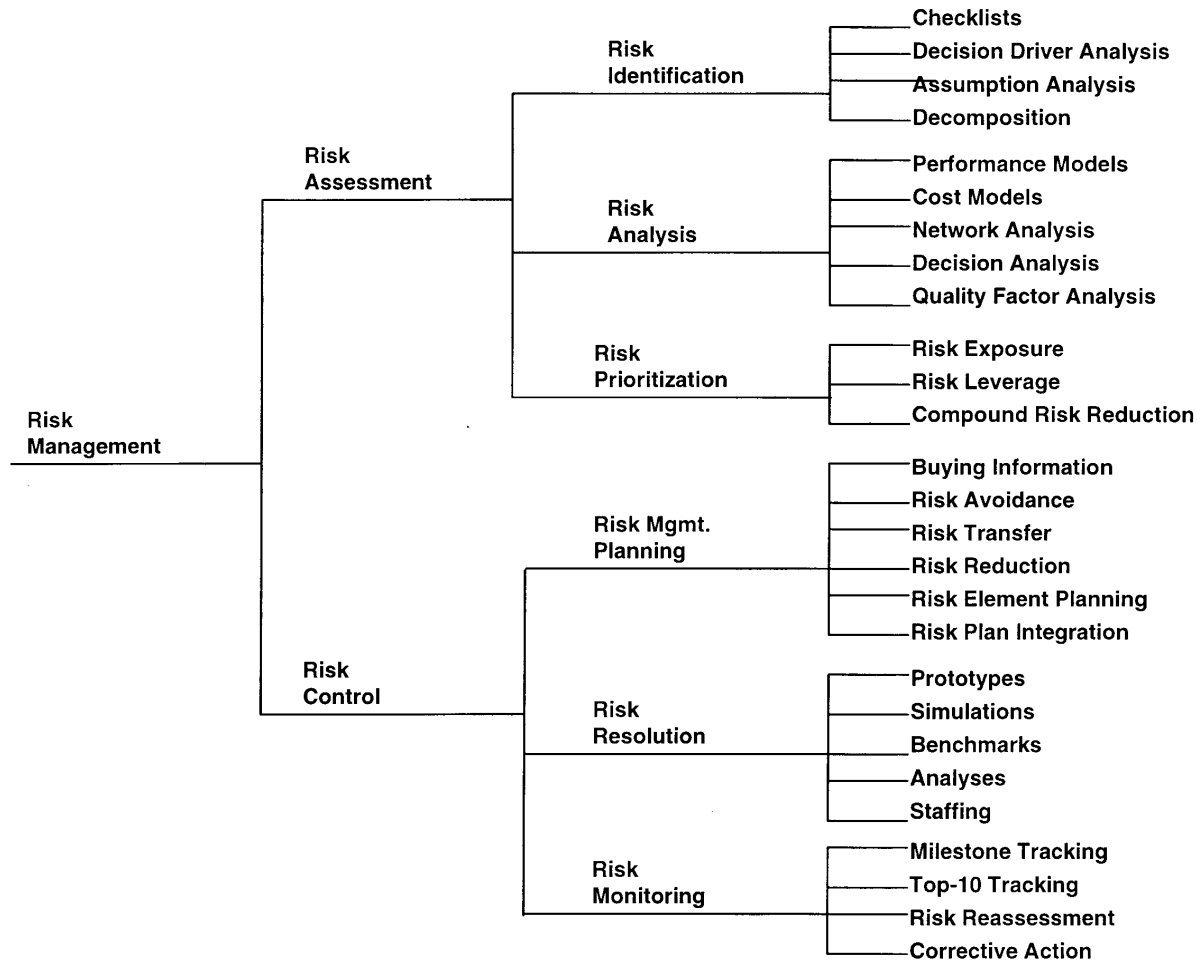
$$\text{RRL} = \frac{\text{RE}_{\text{BEFORE}} - \text{RE}_{\text{AFTER}}}{\text{RISK REDUCTION COST}}$$

## · Spacecraft Example

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



# Software Risk Management





# Risk Management Plans

**For Each Risk Item, Answer the Following Questions:**

**1. Why?**

**Risk Item Importance, Relation to Project Objectives**

**2. What, When?**

**Risk Resolution Deliverables, Milestones, Activity Nets**

**3. Who, Where?**

**Responsibilities, Organization**

**4. How?**

**Approach (Prototypes, Surveys, Models, ...)**

**5. How Much?**

**Resources (Budget, Schedule, Key Personnel)**



# Risk Management Plan: Fault Tolerance Prototyping

## 1. Objectives (The “Why”)

- Determine, reduce level of risk of the software fault tolerance features causing unacceptable performance
- Create a description of and a development plan for a set of low-risk fault tolerance features

## 2. Deliverables and Milestones (The “What” and “When”)

- **By week 3**
  1. Evaluation of fault tolerance option
  2. Assessment of reusable components
  3. Draft workload characterization
  4. Evaluation plan for prototype exercise
  5. Description of prototype
- **By week 7**
  6. Operational prototype with key fault tolerance features
  7. Workload simulation
  8. Instrumentation and data reduction capabilities
  9. Draft Description, plan for fault tolerance features
- **By week 10**
  10. Evaluation and iteration of prototype
  11. Revised description, plan for fault tolerance features



## Risk Management Plan: Fault Tolerance Prototyping (concluded)

- **Responsibilities (The “Who” and “Where”)**
  - **System Engineer: G. Smith**
    - Tasks 1, 3, 4, 9, 11, support of tasks 5, 10
  - **Lead Programmer: C. Lee**
    - Tasks 5, 6, 7, 10 support of tasks 1, 3
  - **Programmer: J. Wilson**
    - Tasks 2, 8, support of tasks 5, 6, 7, 10
- **Approach (The “How”)**
  - Design-to-Schedule prototyping effort
  - Driven by hypotheses about fault tolerance-performance effects
  - Use real-time OS, add prototype fault tolerance features
  - Evaluate performance with respect to representative workload
  - Refine Prototype based on results observed
- **Resources (The “How Much”)**
  - \$60K - Full-time system engineer, lead programmer, programmer (10 weeks)\*(3 staff)\*(\$2K/staff-week)
  - \$0K - 3 Dedicated workstations (from project pool)
  - \$0K - 2 Target processors (from project pool)
  - \$0K - 1 Test co-processor (from project pool)
  - \$10K - Contingencies
  - \$70K - Total



# Risk Monitoring

## Milestone Tracking

- Monitoring of risk Management Plan Milestones

## Top-10 Risk Item Tracking

- Identify Top-10 risk items
  - Highlight these in monthly project reviews
  - Focus on new entries, slow-progress items
- Focus review on manager-priority items

## Risk Reassessment

## Corrective Action



## Project Top 10 Risk Item List: Satellite Experiment Software

Risk Item	Mo. Ranking			Risk Resolution Progress
	This	Last	#Mo.	
Replacing Sensor-Control Software Developer	1	4	2	Top Replacement Candidate Unavailable
Target Hardware Delivery Delays	2	5	2	Procurement Procedural Delays
Sensor Data Formats Undefined	3	3	3	Action Items to Software, Sensor Teams; Due Next Month
Staffing of Design V&V Team	4	2	3	Key Reviewers Committed; Need Fault-Tolerance Reviewer
Software Fault-Tolerance May Compromise Performance	5	1	3	Fault Tolerance Prototype Successful
Accommodate Changes in Data Bus Design	6	-	1	Meeting Scheduled With Data Bus Designers
Testbed Interface Definitions	7	8	3	Some Delays in Action Items; Review Meeting Scheduled
User Interface Uncertainties	8	6	3	User Interface Prototype Successful
TBDs In Experiment Operational Concept	-	7	3	TBDs Resolved
Uncertainties In Reusable Monitoring Software	-	9	3	Required Design Changes Small, Successfully Made



# Example Top-N Risk Item List

<b>Risk</b>	<b>Risk Aversion Options</b>	<b>Risk Monitoring</b>
1. Changes of requirements from previous semester.	Option 1: Propose a solution for the system (describing the requirements in details) to the users and having them commit to the requirements.  Option 2: Adopt an incremental approach to the development by building a prototype first.	Option 1: Once committed, the requirements must be closely monitored. Changes to requirements must be thoroughly assessed and if excessive, they should be defer till later. Option 2: This has an impact on the schedule and hence close monitoring on progress and effort are required.
2. Tight Schedule	Study the requirements carefully so as not to overcommit. Descope good-to-have features if possible. Concentrate on core capabilities.	Close monitoring of all activities is necessary to ensure that schedule are met.
3. Size of project	If requirements are too excessive, descope good-to-have features and capabilities out of the project. Identify the core capabilities to be built.	
4. Finding a search engine	Conduct a software evaluation of search engine. Have team members actively source for free search engines and evaluate them. Determine the best for the project.	Have team members submit evaluation report and conduct demos so that an informed decision can be made.
5. Required technical expertise lacking	Identify the critical and most difficult technical areas of the project and have team members look into them as soon as possible.	Monitor the progress of these critical problems closely. If need be, seek external help.



# **Recommended Implementation Approach Incremental Evolution**

- 1. Establish top-10 risk item tracking process**
  - Simple; inexpensive; gets early results
- 2a. Experimentally apply selected risk mgmt. Techniques**
  - SEI Risk Taxonomy; risk assessment tools
- 2b. Develop, apply risk mgmt. Plan on initial project**
- 3a. Extend risk assessment and control to all projects**
- 3b. Experimentally apply advanced risk mgmt.**
- 4. Evolve the mapping of best risk management techniques to your project situations**



# Conclusions

- **Risk Management is a good match for complex software projects**
  - Avoiding software surprises, show-stoppers
  - Reducing costs
- **Proven risk management techniques are available**
- **Risk management is not a cookbook approach**