

Risk Management Starts on Day One

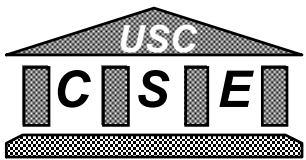
Barry Boehm, USC

So. Calif. Risk Management Symposium

September 13, 2002

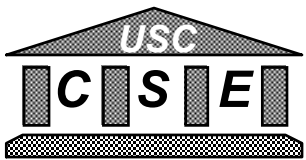
Boehm@sunset.usc.edu

<http://sunset.usc.edu>



Outline

- **Early and Late Risk Resolution**
 - Quotes, Notes, and Data
 - Temptations to Avoid
- **Key Risk Management Practices**
 - Risk Assessment
 - Risk Control



Early Risk Resolution Quotes

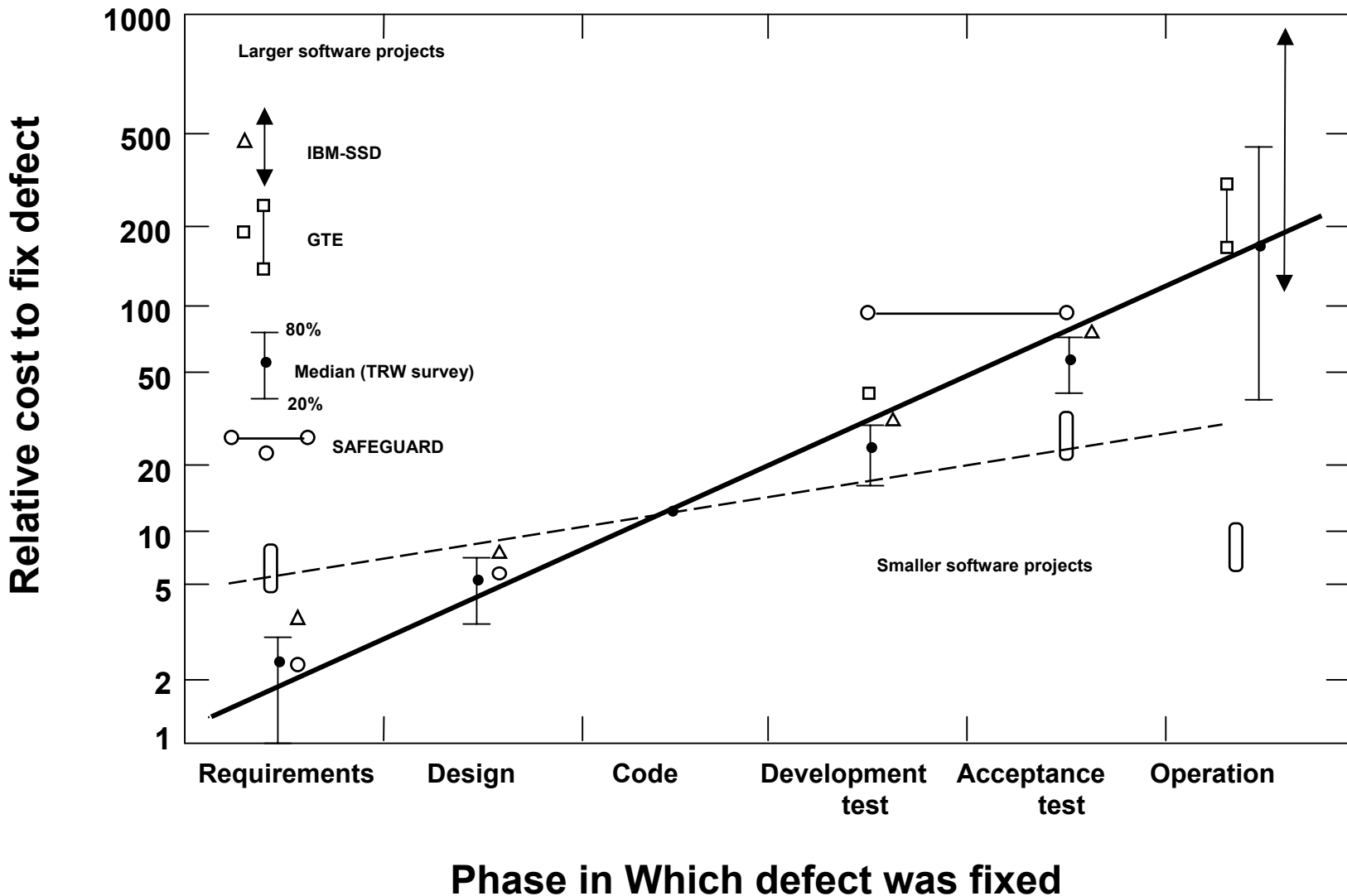
“In architecting a new software program, all the serious mistakes are made on the first day.”

Robert Spinrad, VP-Xerox, 1988

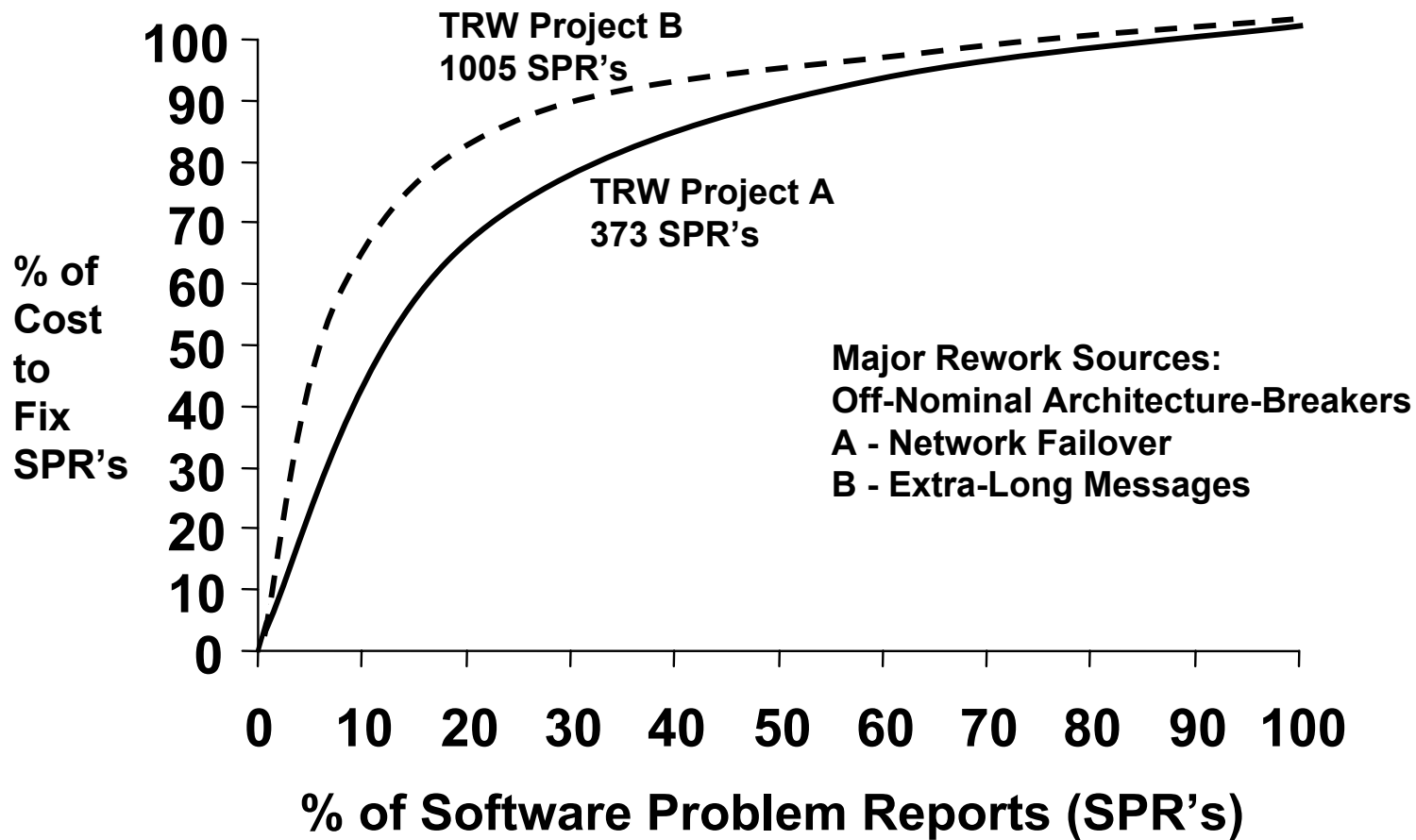
“If you don’t actively attack the risks, the risks will actively attack you.”

Tom Gilb, 1988

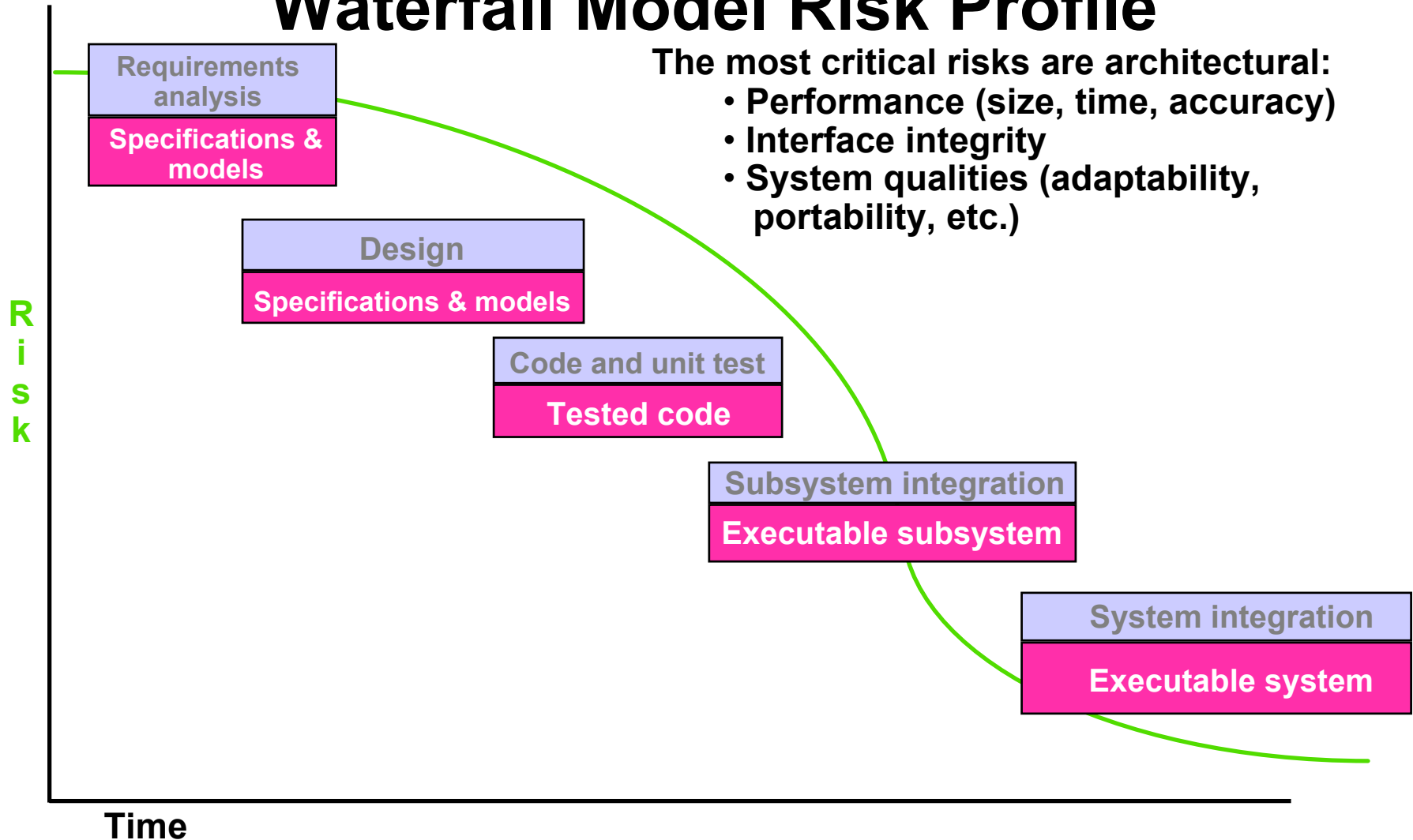
Risk of Delaying Risk Management: Software



Steeper Cost-to-fix for High-Risk Elements



Waterfall Model Risk Profile



The most critical risks are architectural:

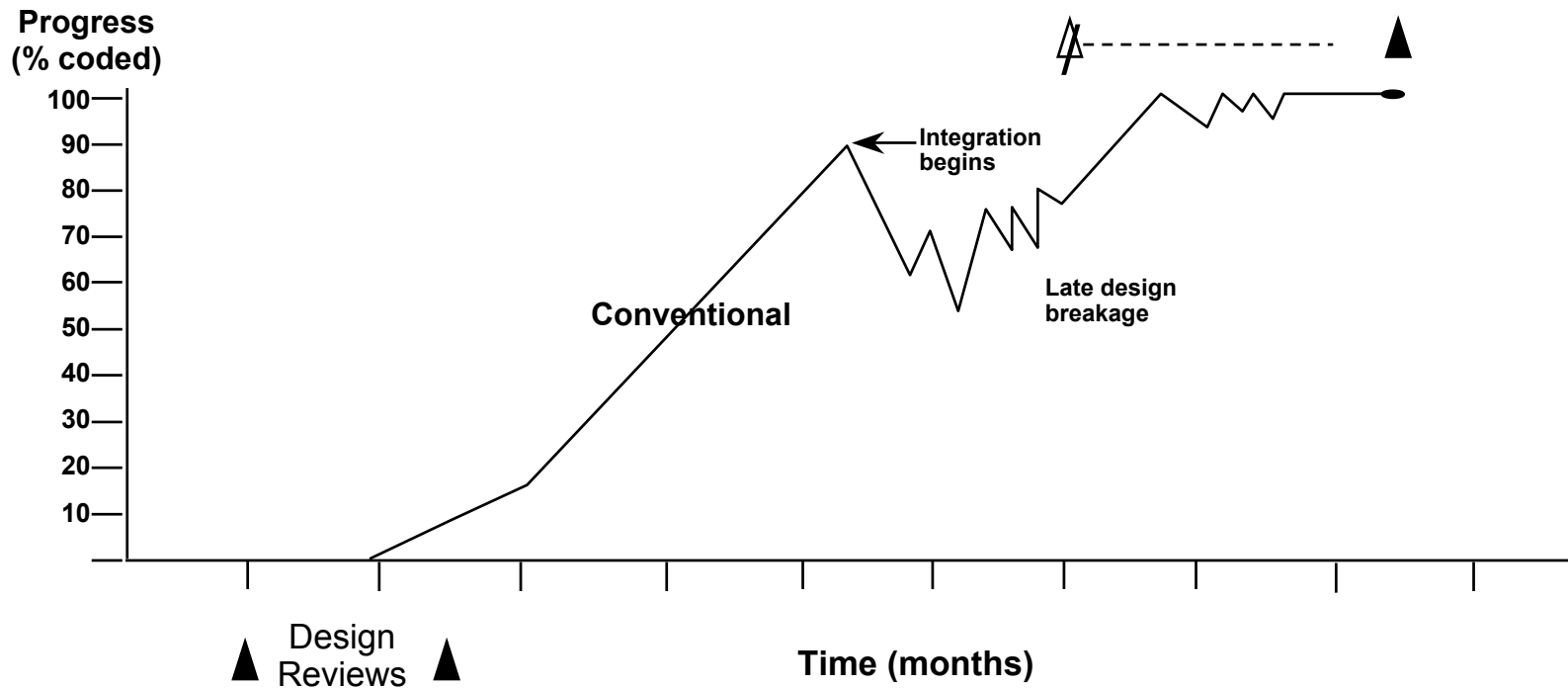
- Performance (size, time, accuracy)
- Interface integrity
- System qualities (adaptability, portability, etc.)

Conventional Software Process

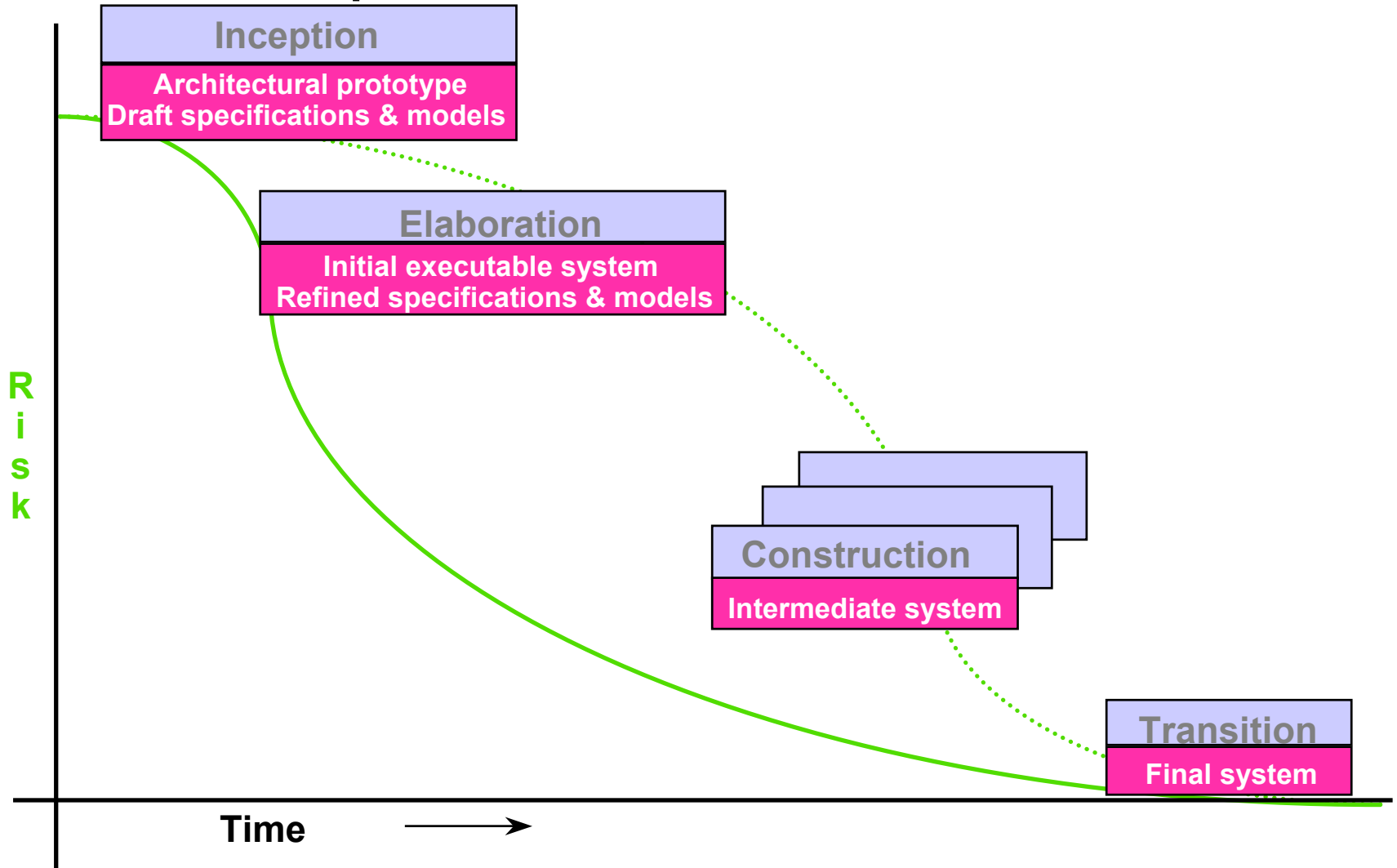
Problem: Late Tangible Design Assessment

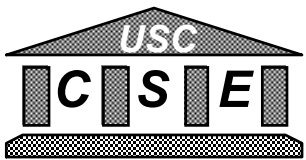
Standard sequence of events:

- Early and successful design review milestones
- Late and severe integration trauma
- Schedule slippage



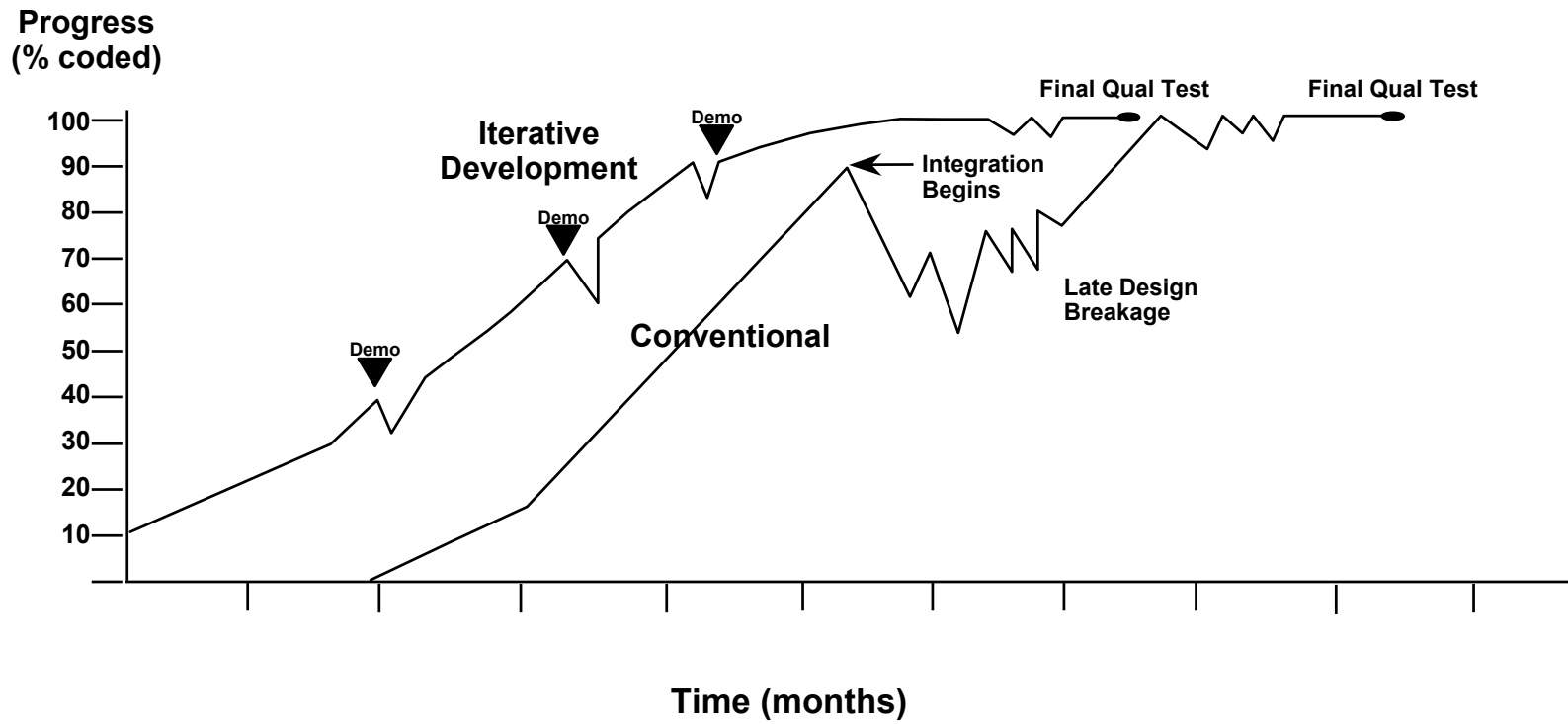
Spiral/MBASE/Rational Risk Profile





Project Results

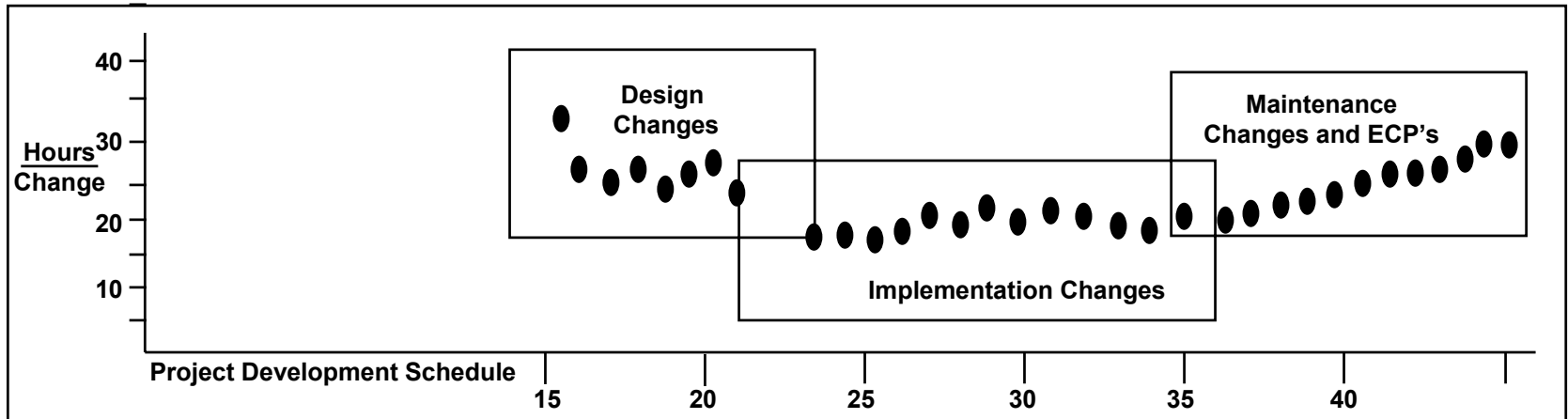
Continuous Integration



Reducing Software Cost-to-Fix: CCPDS-R

- Royce, 1998

- **Architecture first**
 - Integration during the design phase
 - Demonstration-based evaluation
- **Risk Management**
- **Configuration baseline change metrics:**



Day One Temptations to Avoid

- **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”
 - “Shoot the messenger” syndrome
- **Tendency to postpone the hard parts**
 - Maybe they'll go away
 - Maybe they'll get easier, once we do the easy parts
- **Unwillingness to invest money and time up front**

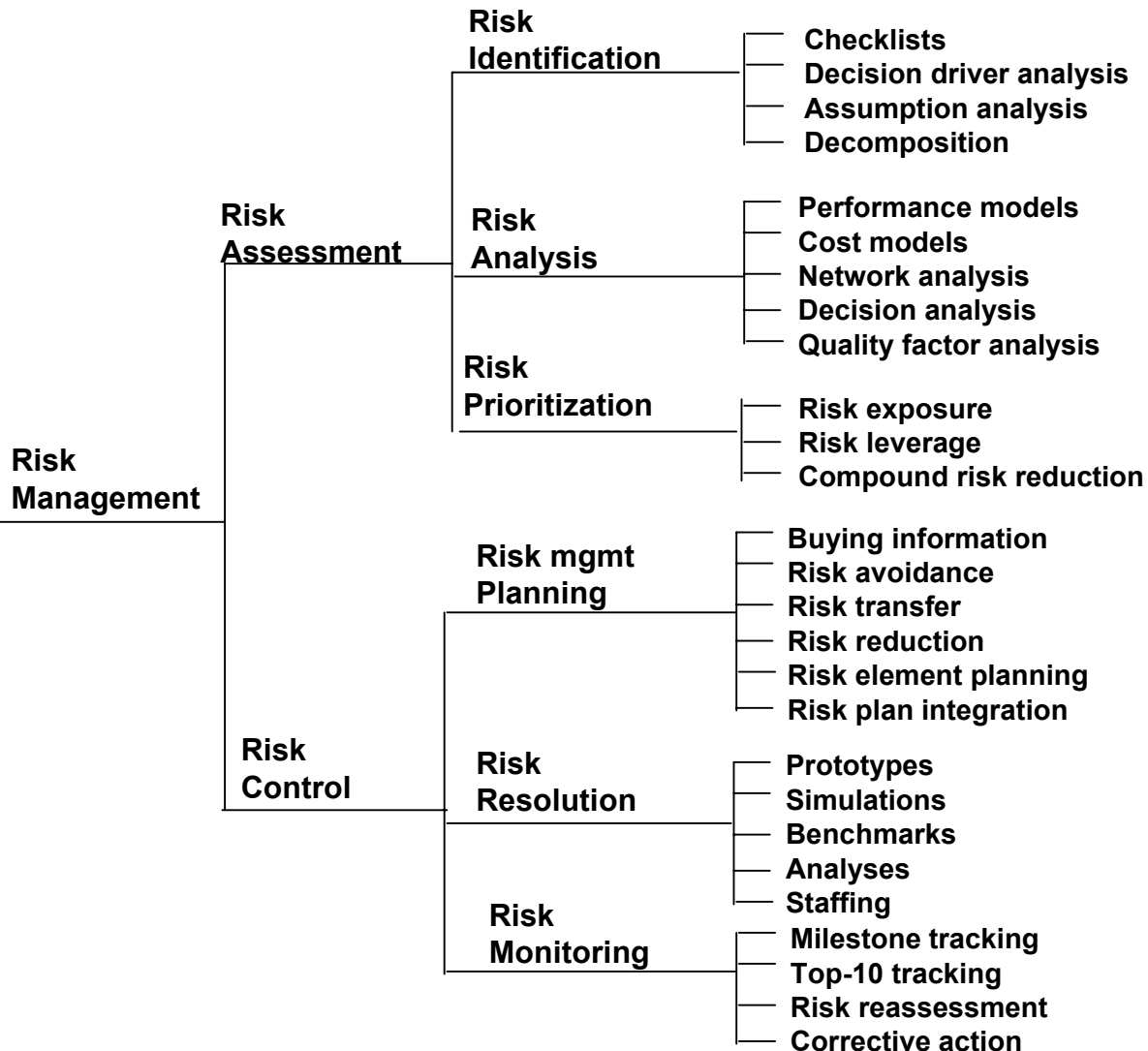
Outline

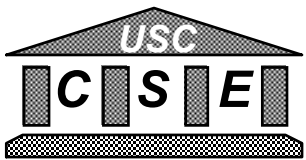
- **Early and Late Risk Resolution**
 - Quotes, Notes, and Data
 - Temptations to Avoid

Key Risk Management Practices

- Risk Assessment
- Risk Control

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
- **Model Clashes**

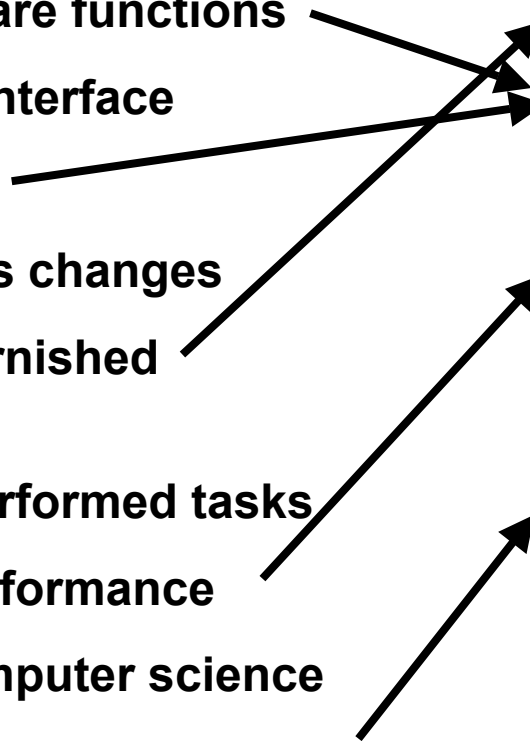
Top 10 Risk Items: 1989 and 1995

1989

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

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



Using Risk to Determine “How Much Is Enough” - testing, planning, specifying, prototyping...

- **Risk Exposure RE = Prob (Loss) * Size (Loss)**

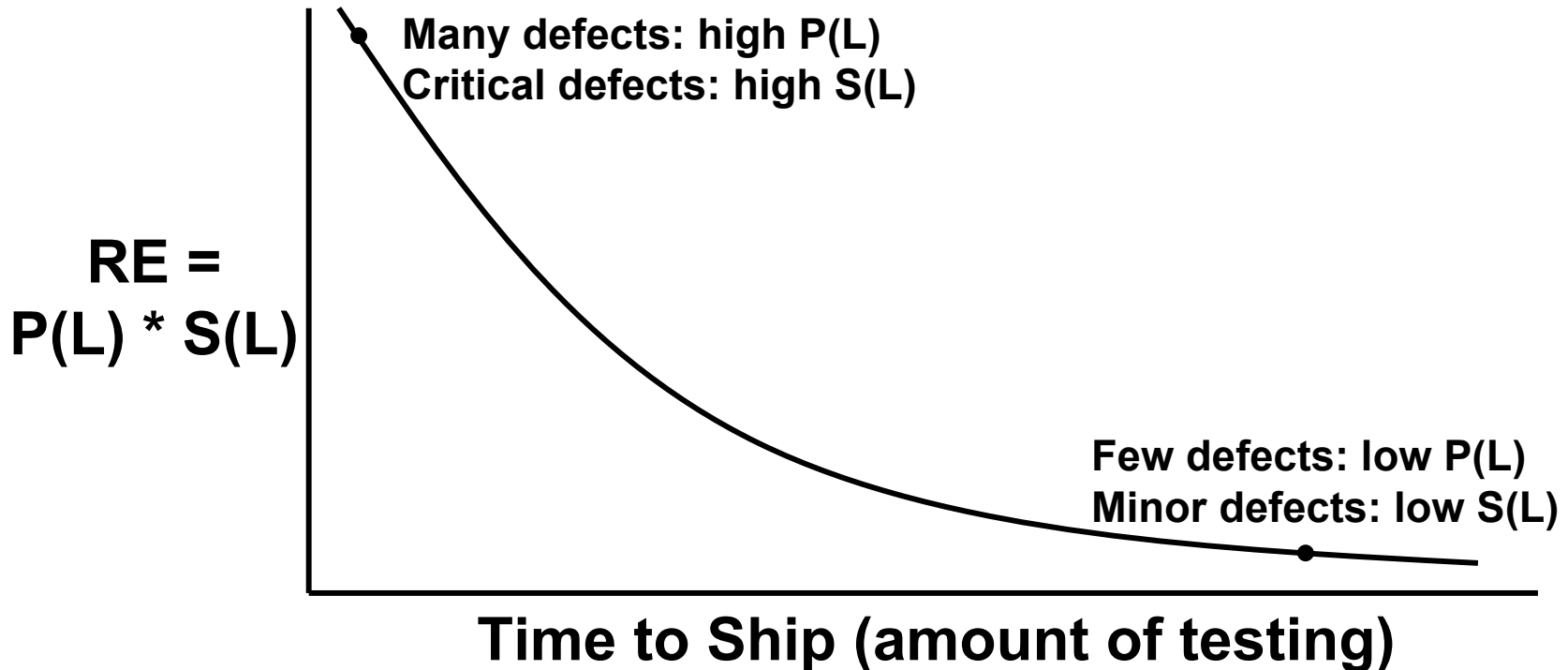
– “Loss” – financial; reputation; future prospects, ...

- **For multiple sources of loss:**

$$RE = \sum_{\text{sources}} [\text{Prob (Loss)} * \text{Size (Loss)}]_{\text{source}}$$

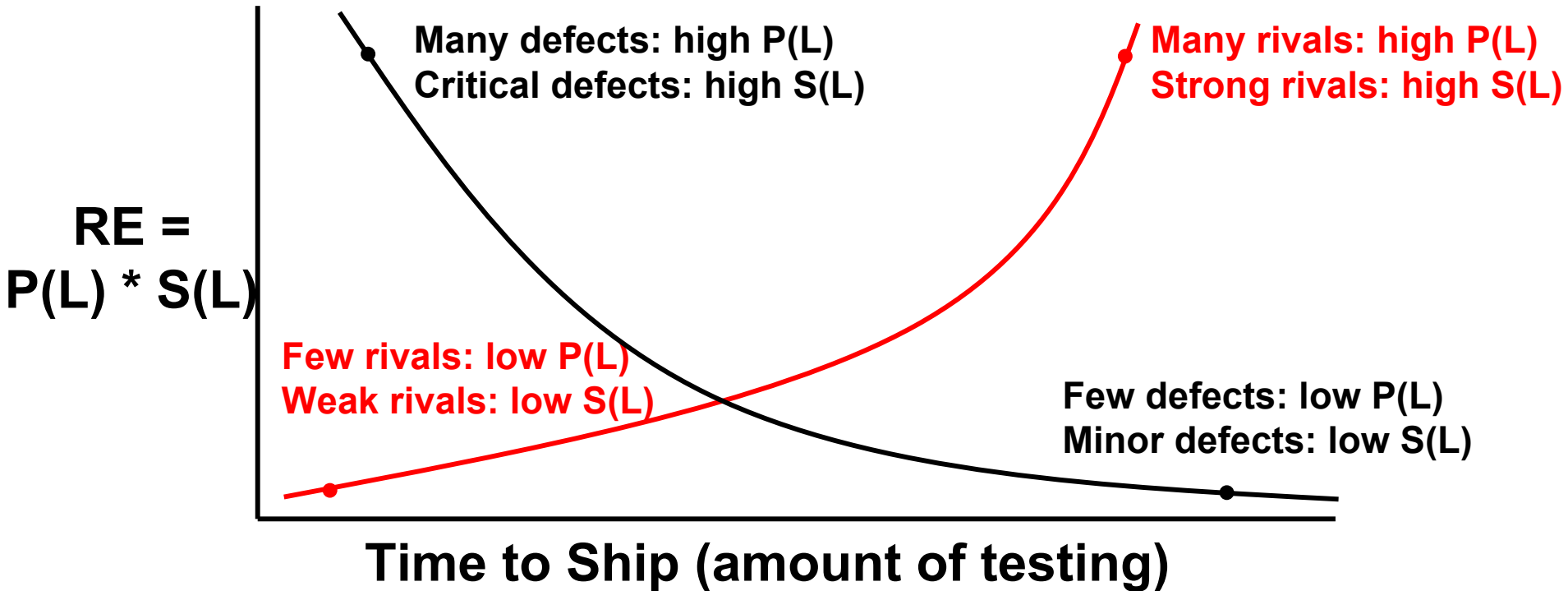
Example RE Profile: Time to Ship

- Loss due to unacceptable dependability



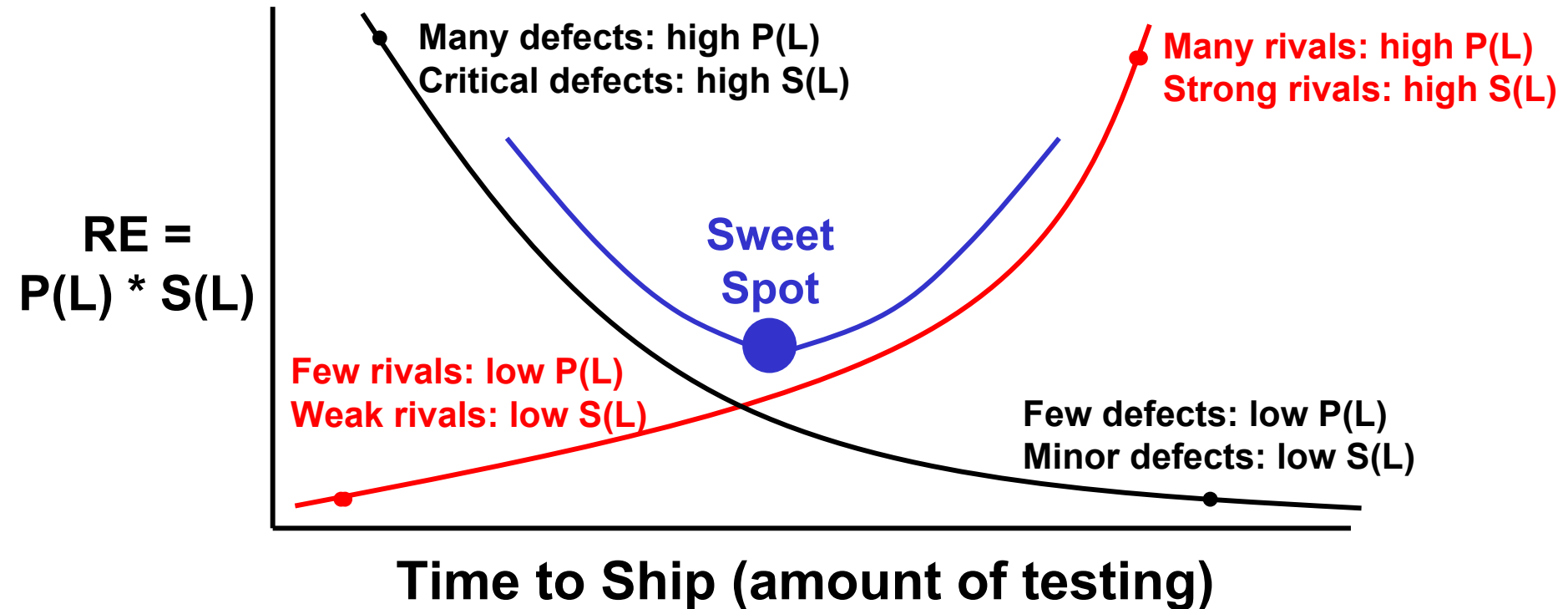
Example RE Profile: Time to Ship

- Loss due to unacceptable dependability
- **Loss due to market share erosion**

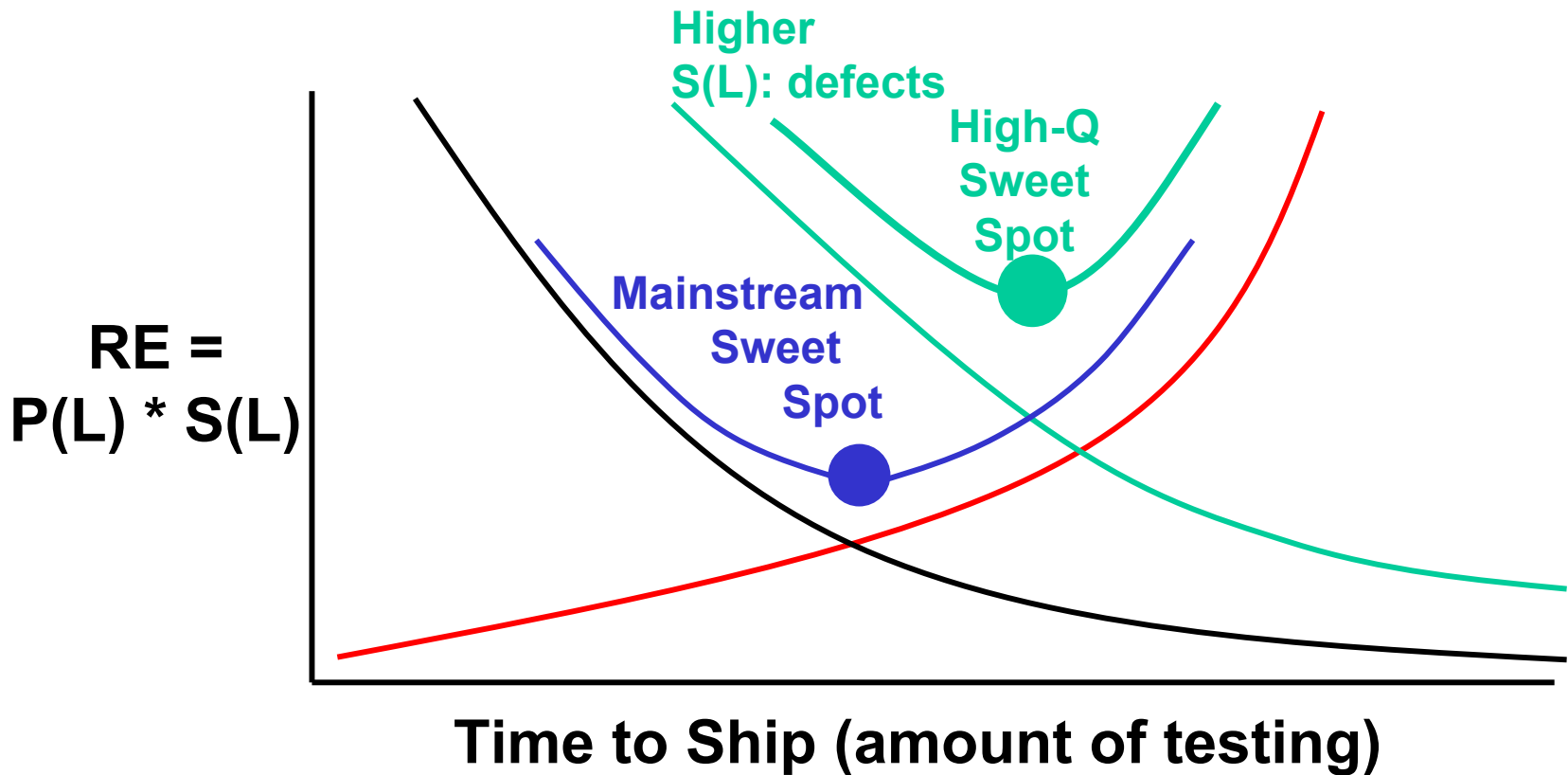


Example RE Profile: Time to Ship

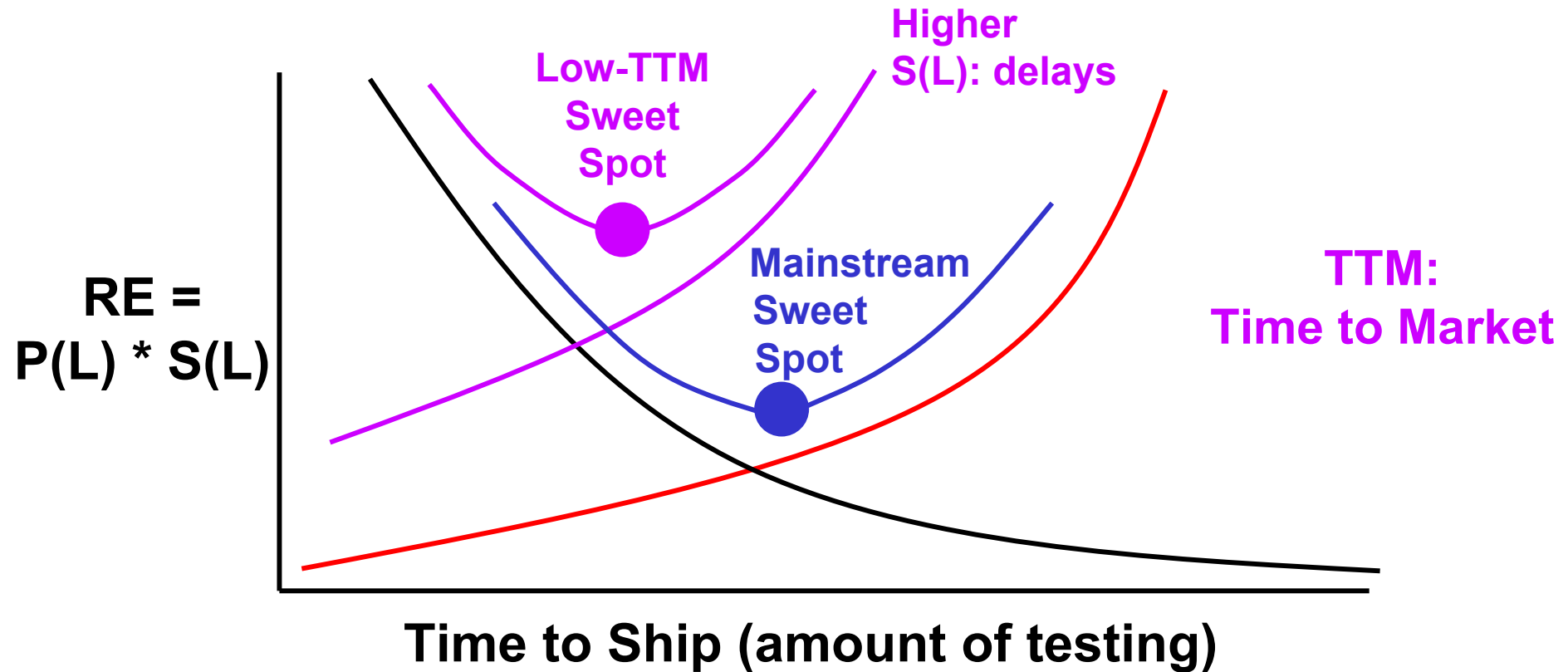
- Sum of Risk Exposures

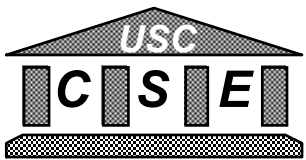


Comparative RE Profile: Safety-Critical System



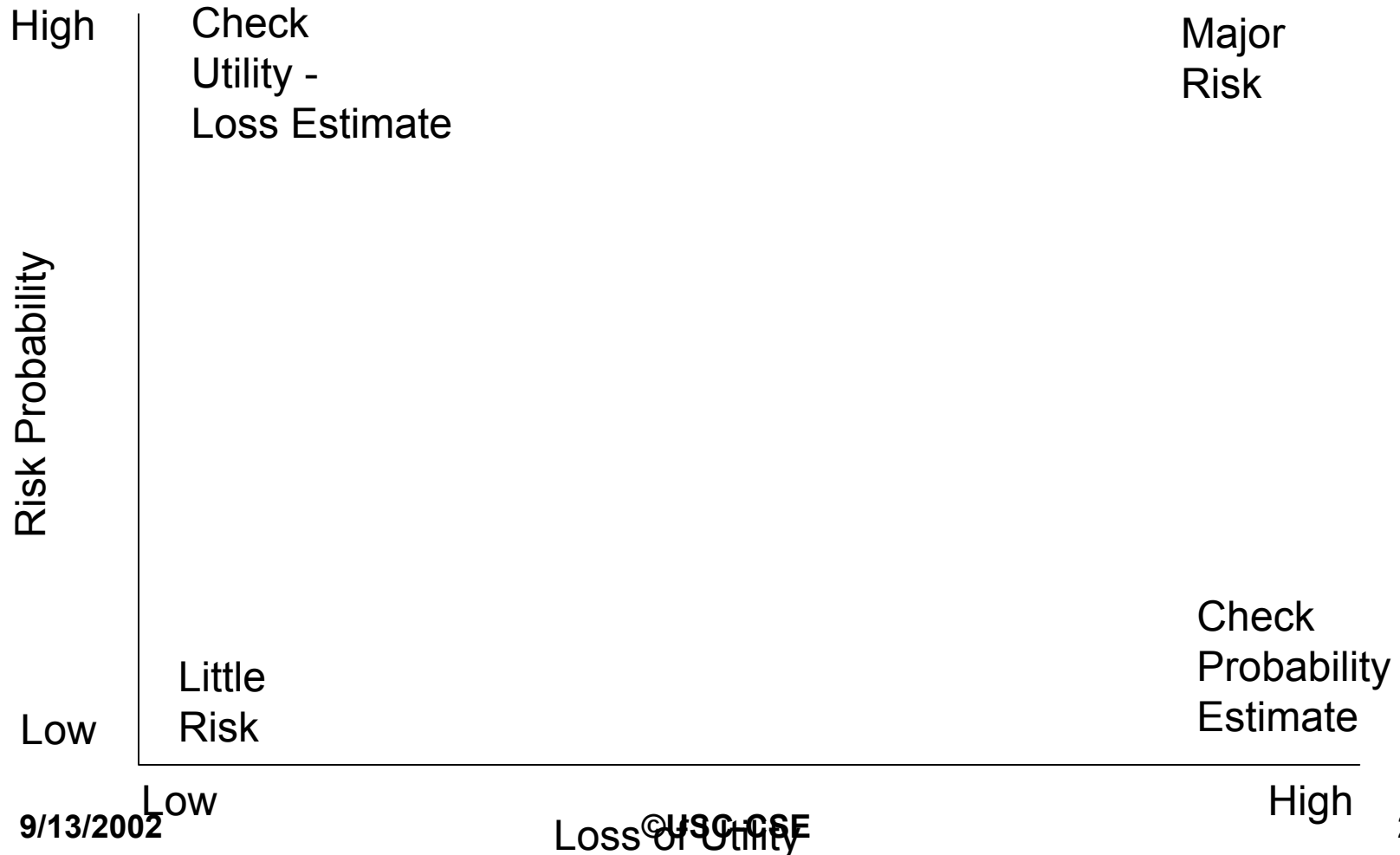
Comparative RE Profile: Internet Startup





Prioritizing Risks: Risk Exposure

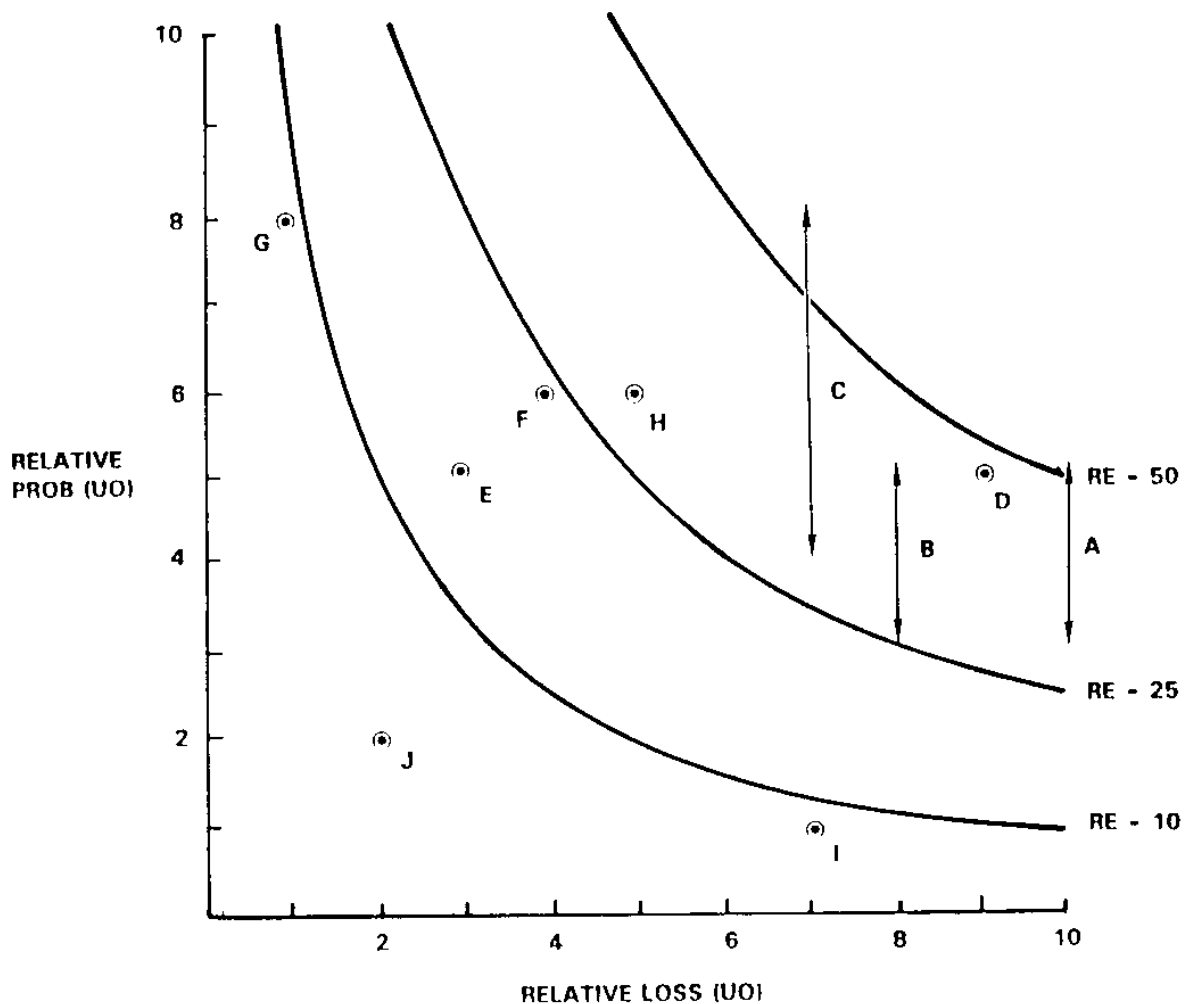
Risk Exposure - (Probability) (Loss of Utility)



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



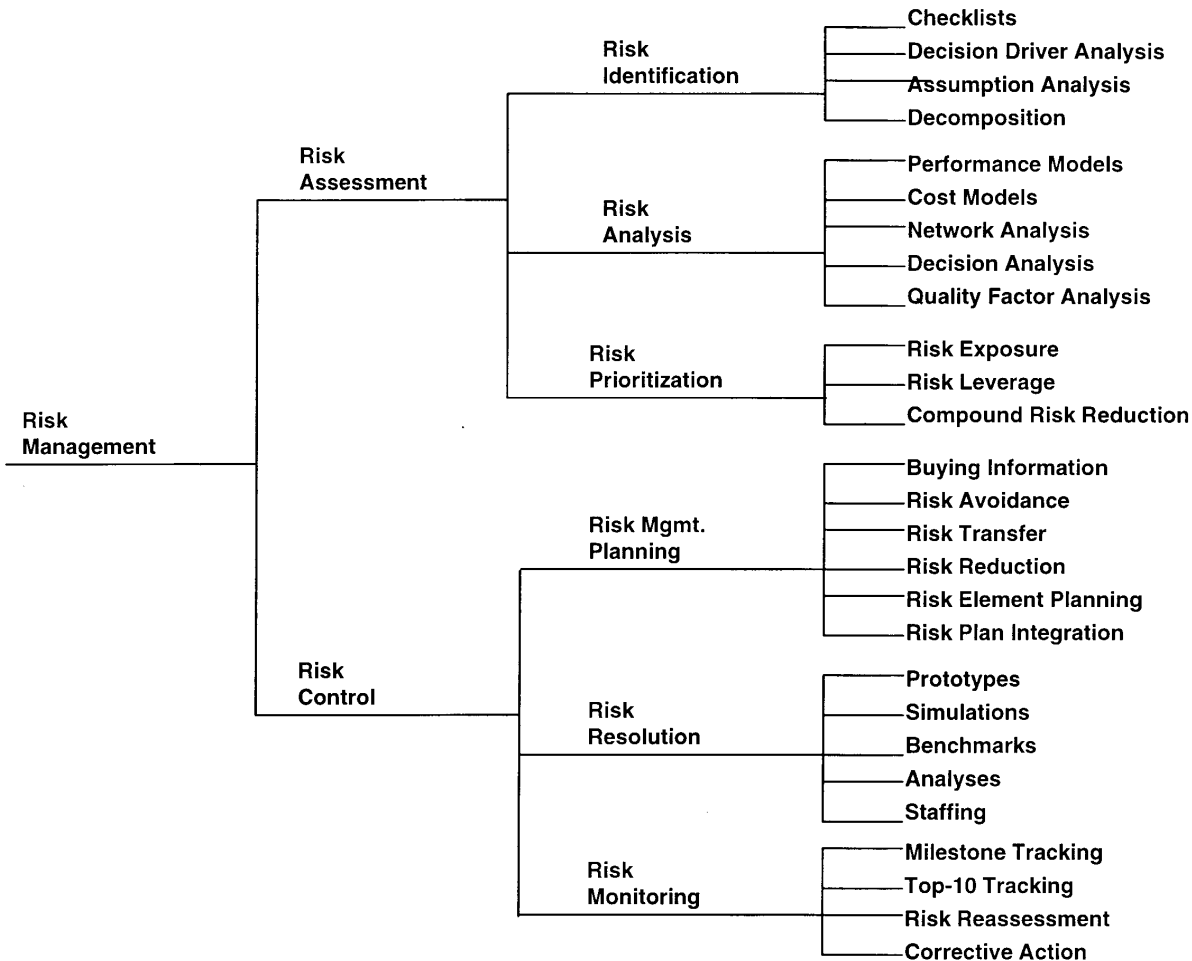
Risk Reduction Leverage (RRL)

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

· Spacecraft Example

	LONG DURATION TEST	FAILURE MODE TESTS
LOSS (UO)	\$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	$\frac{4-1}{2} = 1.5$	$\frac{4-1.4}{0.26} = 10$

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”)

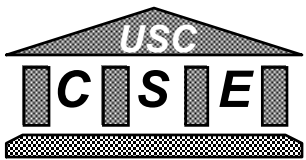
- Determine, reduce level of risk of the software fault tolerance features causing unacceptable performance
- Create a description of and a development plan for a set of low-risk fault tolerance features

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)* $(3 \text{ staff}) * (\$2\text{K}/\text{staff-week})$
 - \$0K - 3 Dedicated workstations (from project pool)
 - \$0K - 2 Target processors (from project pool)
 - \$0K - 1 Test co-processor (from project pool)
 - \$10K - Contingencies



The SAIV* Process Model

– Cross Talk, January 2002 (<http://www.stsc.hill.af.mil/crosstalk>)

1. Shared vision and expectations management
2. Feature prioritization
3. Schedule range estimation and core-capability determination
 - Top-priority features achievable within fixed schedule with 90% confidence
4. Architecting for ease of adding or dropping borderline-priority features
 - And for accommodating past-IOC directions of growth
5. Incremental development
 - Core capability as increment 1
6. Change and progress monitoring and control
 - Add or drop borderline-priority features to meet schedule

*Schedule As Independent Variable; Feature set as dependent variable
– Also works for cost, schedule/cost/quality as independent variable

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

Risk	Risk Aversion Options	Risk Monitoring
1. Changes of requirements from previous semester.	<p>Option 1: Propose a solution for the system (describing the requirements in details) to the users and having them commit to the requirements.</p> <p>Option 2: Adopt an incremental approach to the development by building a prototype first.</p>	<p>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.</p> <p>Option 2: This has an impact on the schedule and hence close monitoring on progress and effort are required.</p>
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.

Conclusions

- **Risk management starts on Day One**
 - Delay and denial are serious career risks
 - Data provided to support early investment
- **Win Win spiral model provides process framework for early risk resolution**
 - Stakeholder identification and win condition reconciliation
 - Anchor point milestones
- **Risk analysis helps determine “how much is enough”**
 - Testing, planning, specifying, prototyping,...
 - Buying information to reduce risk