Software Risk Management :

Overview and Recent Developments

Barry Boehm, USC

COCOMO/SCM Forum #17 Tutorial

October 22,2002

(boehm@sunset.usc.edu)

(http://sunset.usc.edu)

Outline

- What is Software Risk Management?
- What can it help you do?
- When should you do it?
- How should you do it?
- What are its new trends and opportunities?
- Conclusions
- Top-10 Risk Survey

Is This A Risk?

- We just started integrating the software
  - and we found out that COTS* products A and B just can’t talk to each other
- We’ve got too much tied into A and B to change
- Our best solution is to build wrappers around A and B to get them to talk via CORBA**
- This will take 3 months and $300K
- It will also delay integration and delivery by at least 3 months

*COTS: Commercial off-the-shelf

**CORBA: Common Object Request Broker Architecture

Is This A Risk?

- We just started integrating the software
  - and we found out that COTS* products A and B just can't talk to each other
- We've got too much tied into A and B to change

****

- No, it is a problem
  - Being dealt with reactively
- Risks involve uncertainties
  - And can be dealt with pro-actively
  - Earlier, this problem was a risk

Earlier, This Problem Was A Risk

- A and B are our strongest COTS choices
  - But there is some chance that they can't talk to each other
  - Probability of loss P(L)
- If we commit to using A and B
  - And we find out in integration that they can't talk to each other
  - We'll add more cost and delay delivery by at least 3 months
  - Size of loss S(L)
- We have a risk exposure of

$$RE = P(L) * S(L)$$

How Can Risk Management Help You Deal With Risks?

- Buying information
- Risk avoidance
- Risk transfer
- Risk reduction
- Risk acceptance

Risk Management Strategies: Buying Information

- Let's spend $30K and 2 weeks prototyping the integration of A and B
- This will buy information on the magnitude of P(L) and S(L)
- If RE = P(L) * S(L) is small, we'll accept and monitor the risk
- If RE is large, we'll use one/some of the other strategies

Other Risk Management Strategies

- Risk Avoidance
  - COTS product C is almost as good as B, and it can talk to A
  - Delivering on time is worth more to the customer than the small performance loss
- Risk Transfers
  - If the customer insists on using A and B, have them establish a risk reserve.
  - To be used to the extent that A and B can't talk to each other
- Risk Reduction
  - If we build the wrappers and the CORBA corrections right now, we add cost but minimize the schedule delay
- Risk Acceptance
  - If we can solve the A and B interoperability problem, we'll have a big competitive edge on the future procurements
  - Let's do this on our own money, and patent the solution

Is Risk Management Fundamentally Negative?

- It usually is, but it shouldn't be
- As illustrated in the Risk Acceptance strategy, it is equivalent to Opportunity Management

  Opportunity Exposure OE = P(Gain) * S(Gain)
  = Expected Value
- Buying information and the other Risk Strategies have their Opportunity counterparts
  - P(Gain): Are we likely to get there before the competition?
  - S(Gain): How big is the market for the solution?

What Else Can Risk Risk Management Help You Do?

- Determine "How much is enough?" for your products and processes
  - Functionality, documentation, prototyping, COTS evaluation, architecting, testing, formal methods, agility, discipline, …
  - What's the risk exposure of doing too much?
  - What's the risk exposure of doing too little?
- Tailor and adapt your life cycle processes
  - Determine what to do next (specify, prototype, COTS evaluation, business case analysis)
  - Determine how much of it is enough
  - Examples: Risk-driven spiral model and extensions (win-win, anchor points, RUP, MBASE, CeBASE Method)
- Get help from higher management
  - Organize management reviews around top-10 risks

Outline

- What is Software Risk Management?
- What can it help you do?
- When should you do it?
- How should you do it?
- What are its new trends and opportunities?
- Conclusions
- Top-10 Risk Survey

Risk Management Starts on Day One

- Early and Late Risk Resolution
  - Quotes, Notes, and Data
  - Temptations to Avoid
- Early Risk Resolution with the WinWin Spiral Model
  - Identifying Stakeholders and Win Conditions
  - Model Clash and Win-Lose Risk Avoidance
  - Avoiding Cost/Schedule Risks with the SAIV Model

Early Risk Resolution Quotes

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

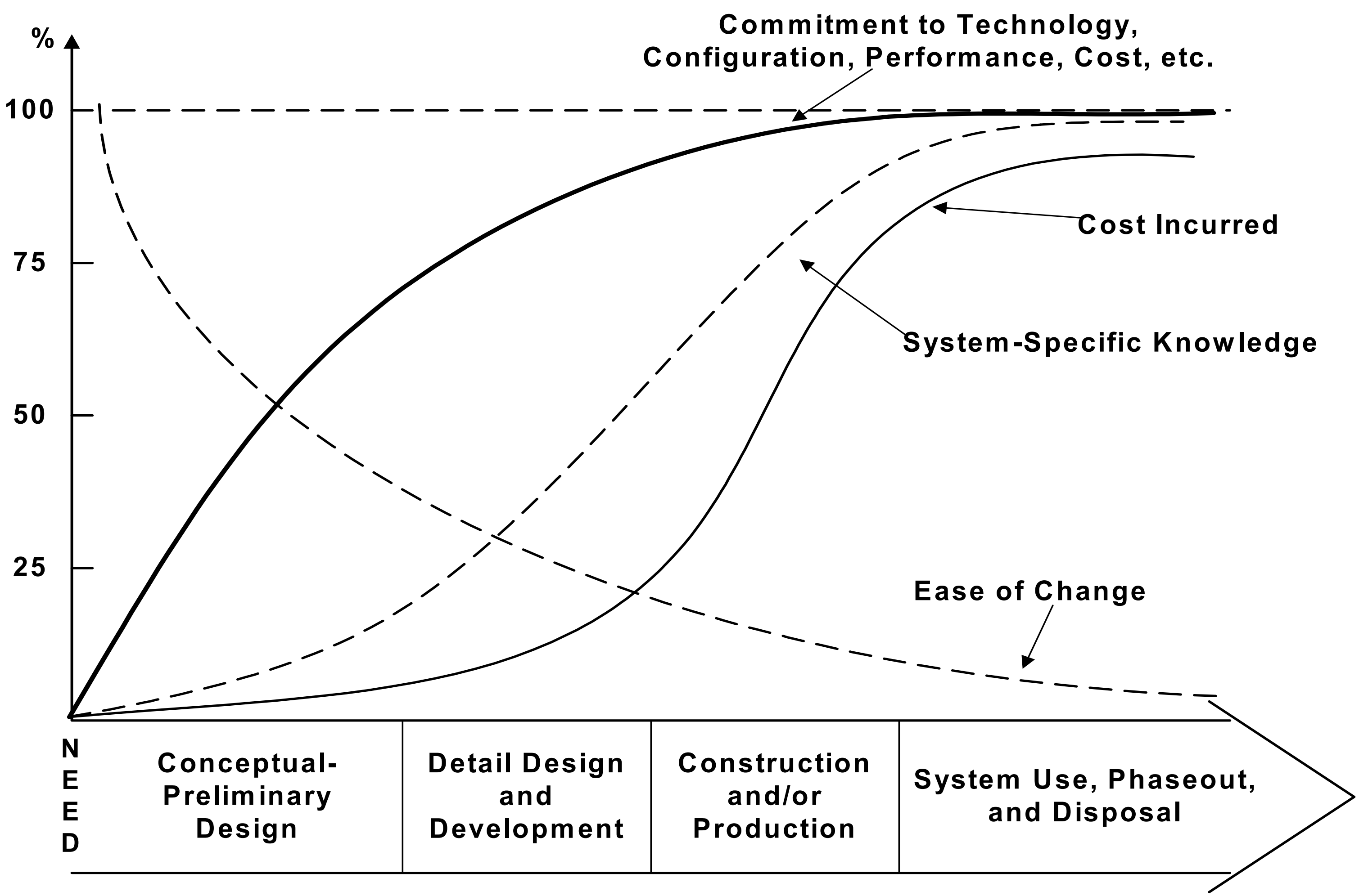
Robert Spinrad, VP-Xerox, 1988

"If you don't actively attack the risks, the risks will actively attack you."

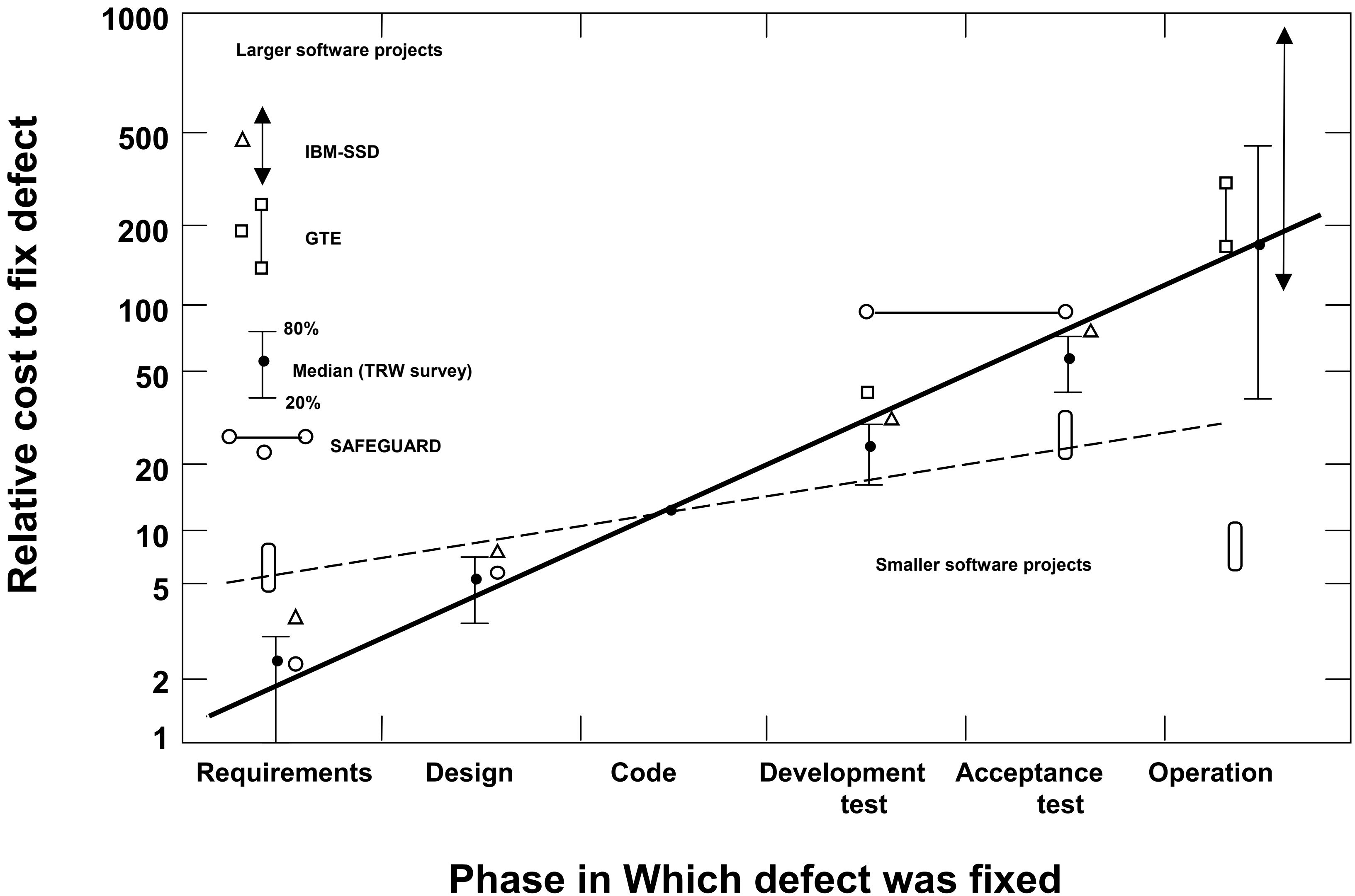
Tom Gilb, 1988

Risk of Delaying Risk Management: Systems

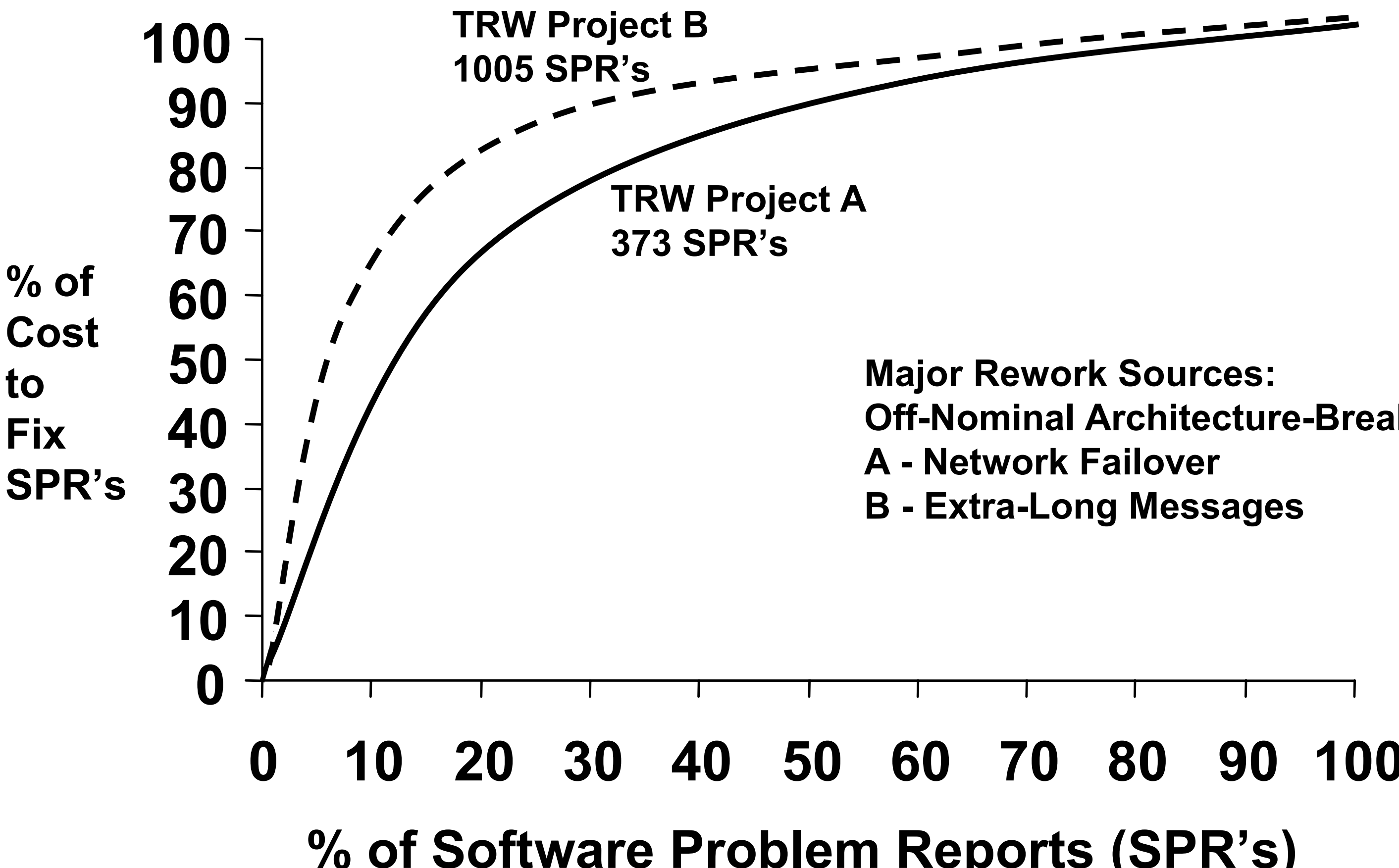
—Blanchard- Fabrycky, 1998



Risk of Delaying Risk Management: Software



Steeper Cost-to-fix for High-Risk Elements

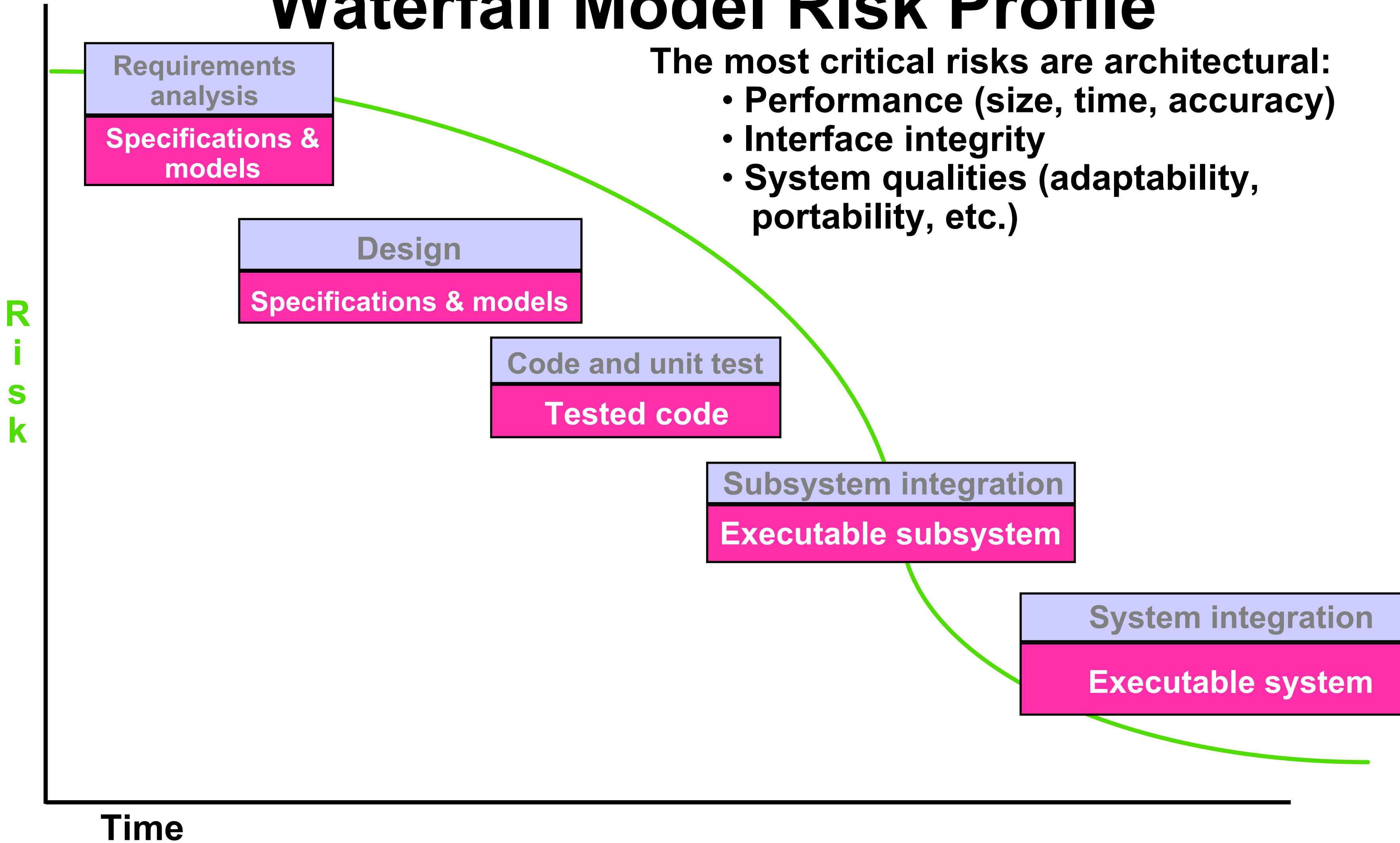


Major Rework Sources:
Off-Nominal Architecture-Breakers
A - Network Failover
B - Extra-Long Messages

Waterfall Model Risk Profile

The most critical risks are architectural:

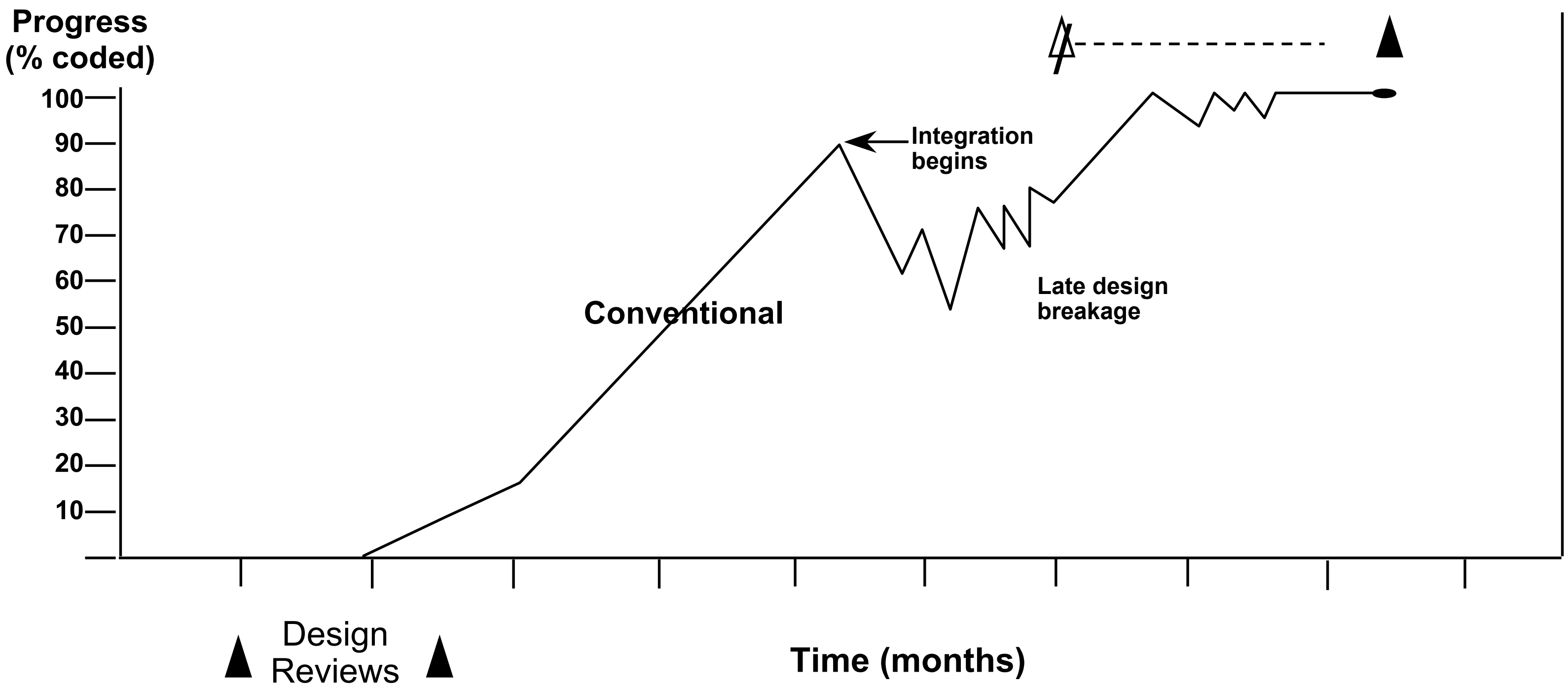
- Performance (size, time, accuracy)
- Interface integrity
- System qualities (adaptability, portability, etc.)



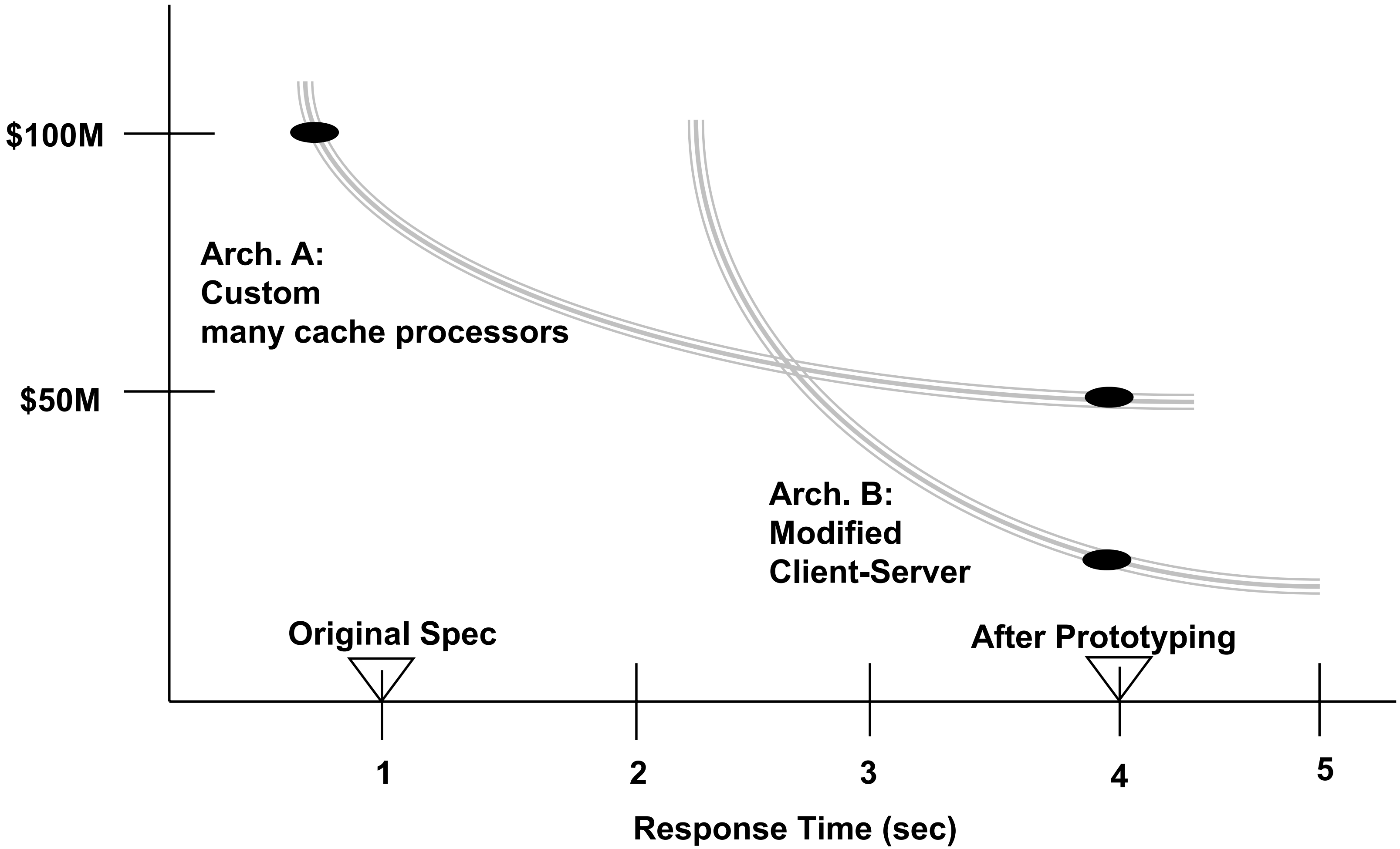
Conventional Software Process

Problem: Late Tangible Design Assessment

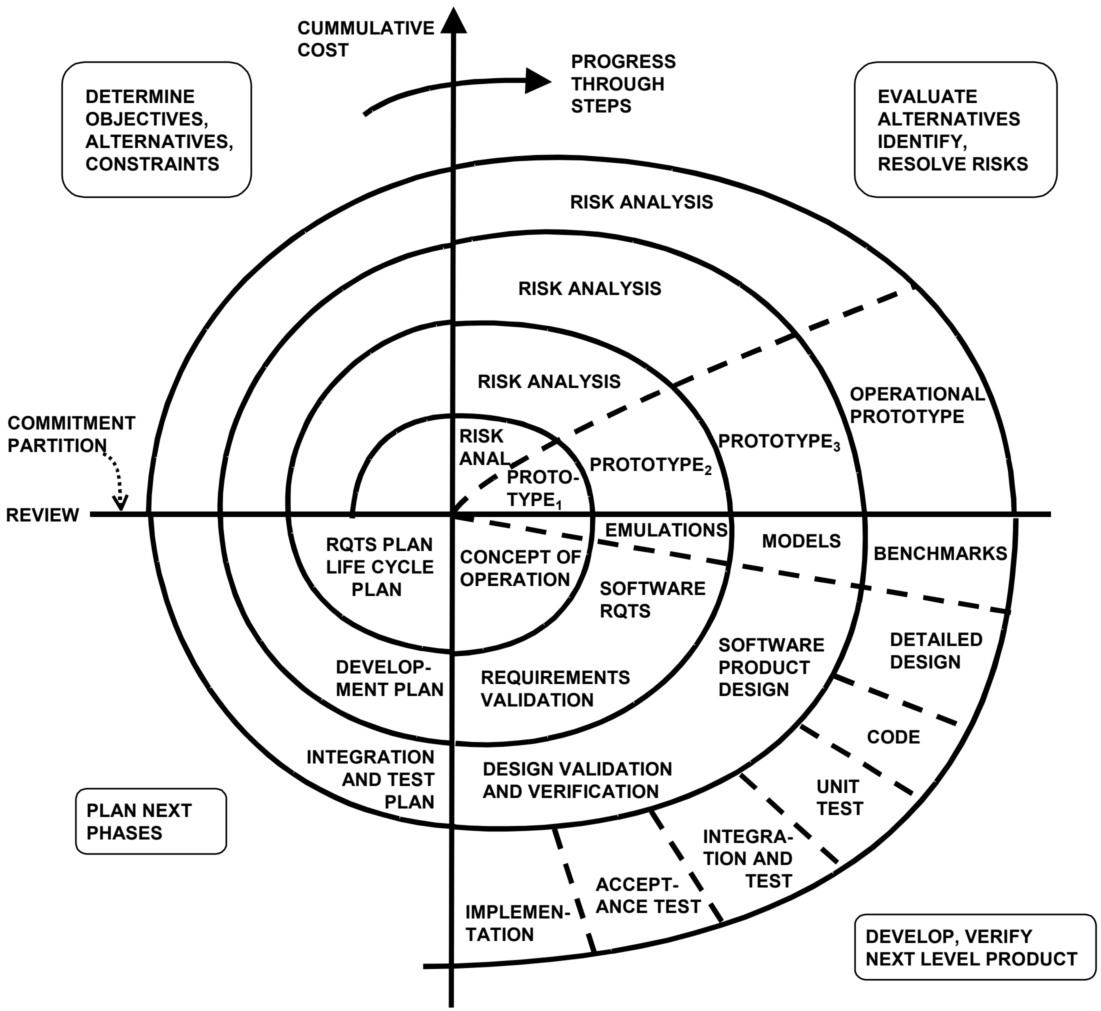
Standard sequence of events:
- Early and successful design review milestones
- Late and severe integration trauma
- Schedule slippage



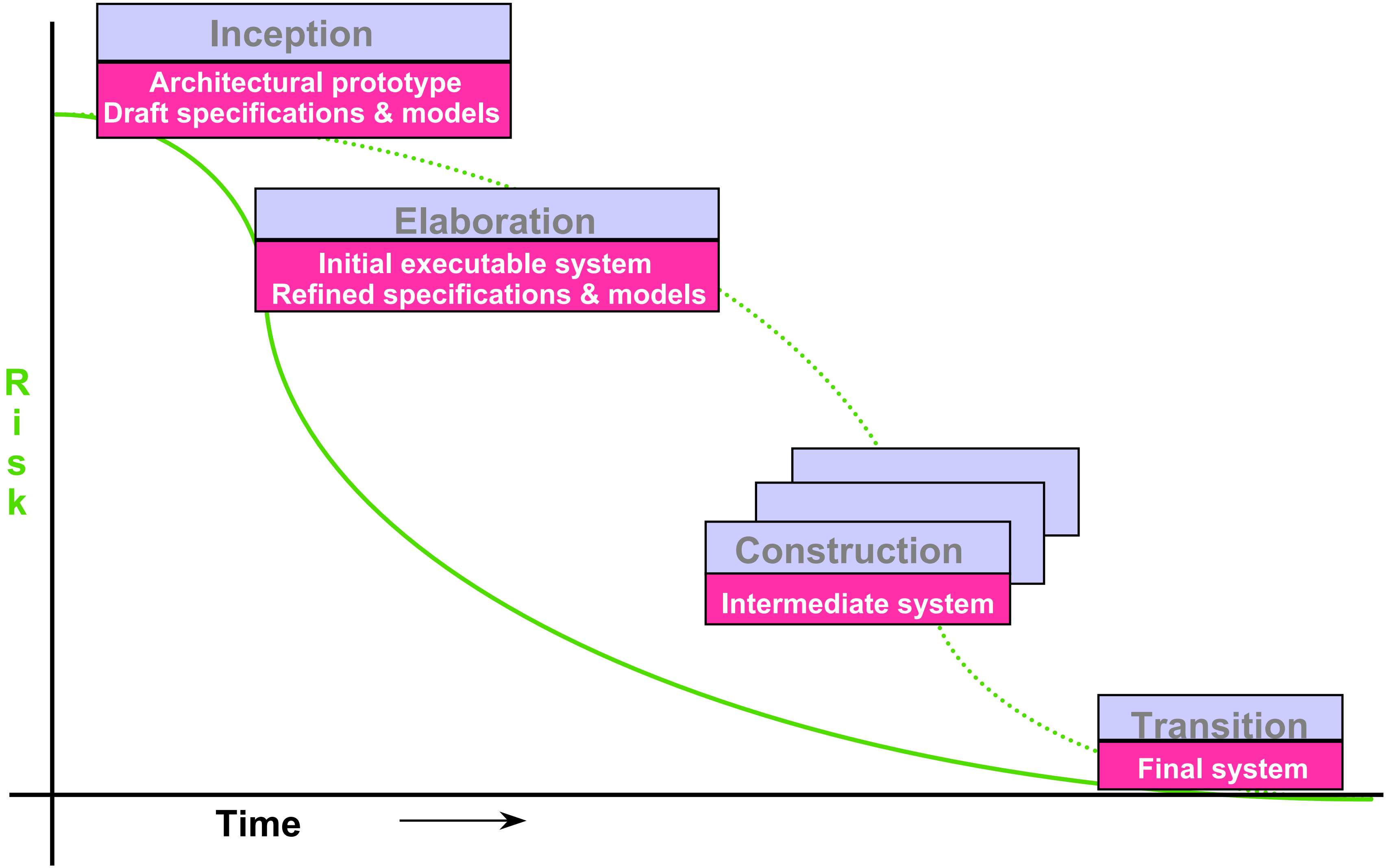
Sequential Engineering Neglects Risk



Spiral Model of Software Process

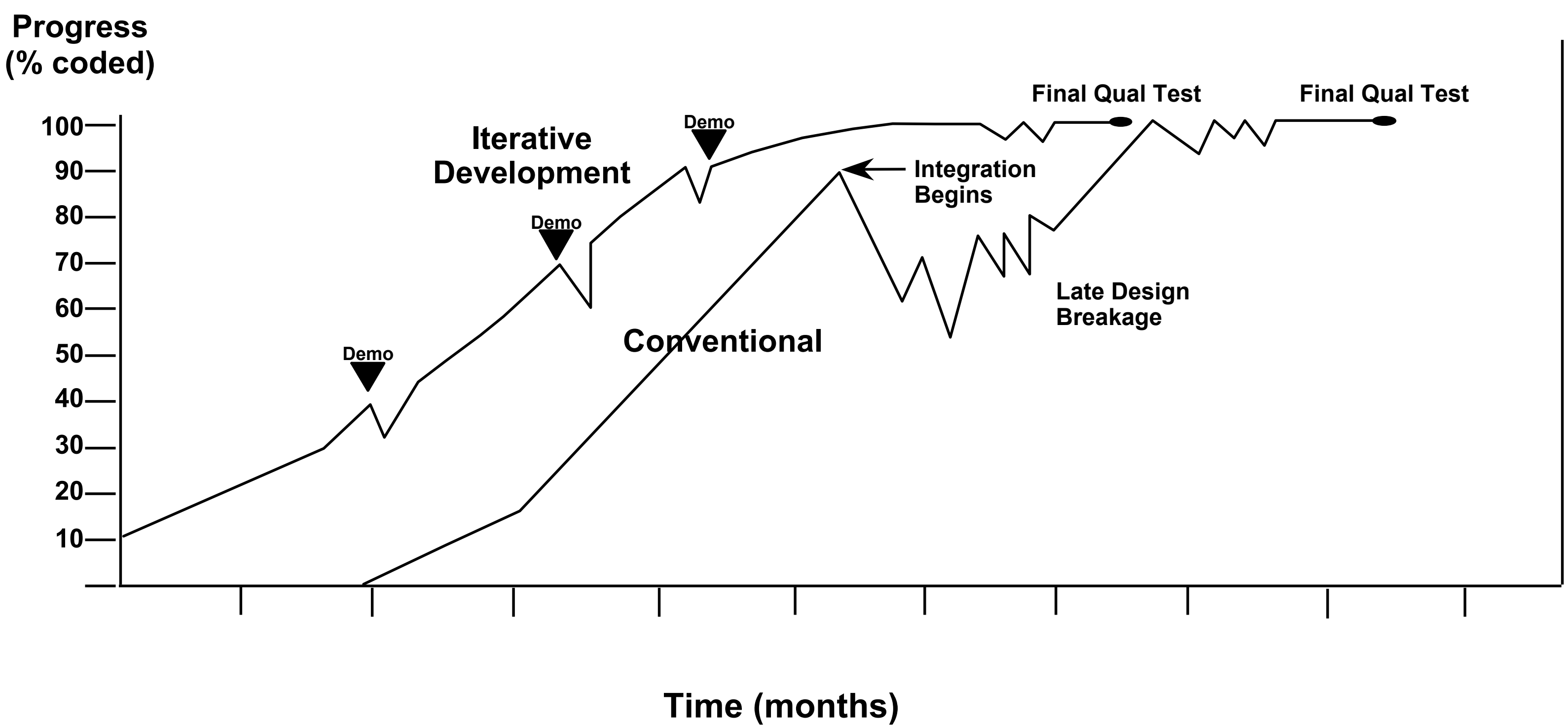


Spiral/MBASE/Rational Risk Profile



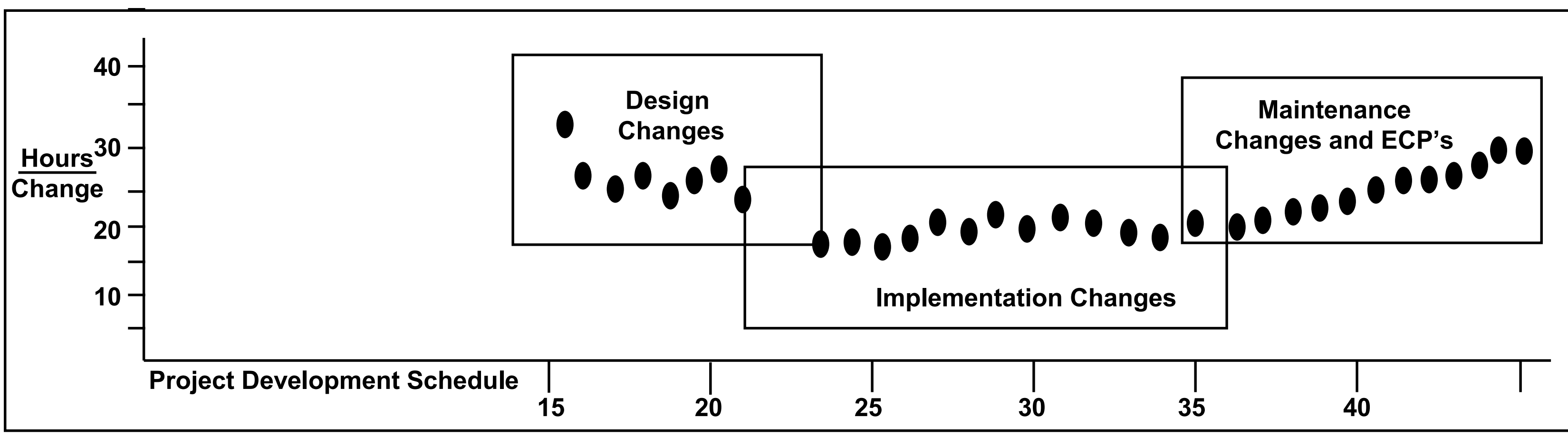
Project Results

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 - I

- It's too early to think about risks. We need to:
  - Finalize the requirements
  - Maximize our piece of the pie
  - Converge on the risk management organization, forms, tools, and procedures. Don't put the cart before the horse.
- The real horse is the risks, and it's leaving the barn
  - Don't sit around laying out the seats on the cart while this happens. Work this concurrently.

Day One Temptations to Avoid - II

- We don't have time to think about the risks. We need to:
  – Get some code running right away
  – Put on a socko demo for the customers
  – Be decisive. Lead. Make commitments.
- The Contrarian's Guide to Leadership (Sample, 2002)
  – Never make a decision today that can be put off till tomorrow.
  – Don't form opinions if you don't have to. Think gray.

Day One Temptations to Avoid - III

- 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

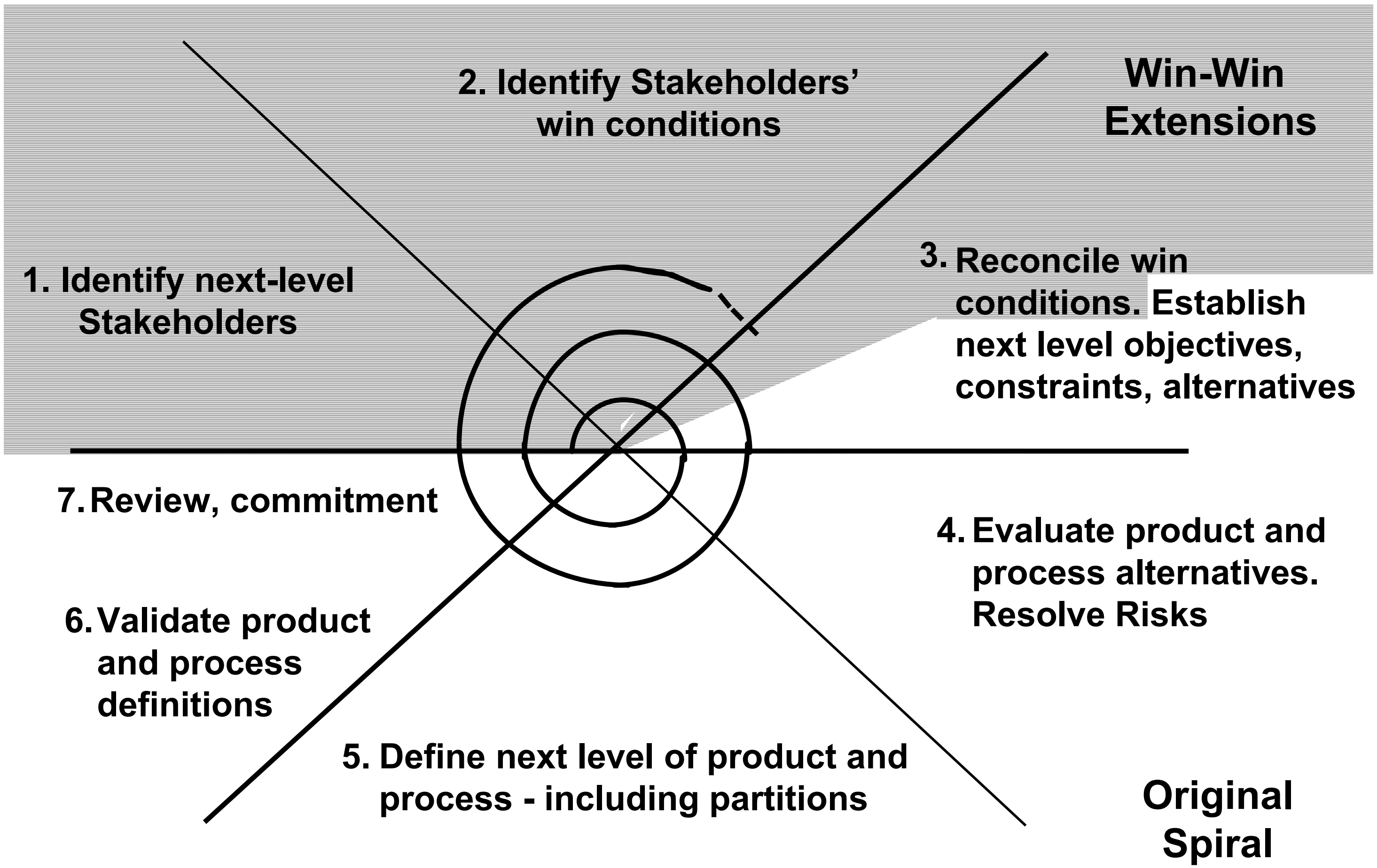
Risk Management Starts on Day One

- Early and Late Risk Resolution
  - Quotes, Notes, and Data
  - Temptations to Avoid
- Early Risk Resolution with the Win Win Spiral Model
  - Identifying Stakeholders and Win Conditions
  - Model Clash and Win-Lose Risk Avoidance
  - Avoiding Cost/Schedule Risks with the SAIV Model

Experience with the Spiral Model

- Good for getting risks resolved early
- Good for tailoring process to situation
- Good for accommodating COTS, reuse, prototyping
- Where do objectives, constraints, and alternatives come from?
  - WinWin Spiral Model
- No common life cycle milestones
  - Anchor Points
- Need to avoid model clashes
  - MBASE

The WinWin Spiral Model



Life Cycle Anchor Points

- Common System/Software stakeholder commitment points
  - Defined in concert with Government, industry affiliates
  - Coordinated with Rational's Unified Software Development Process
- Life Cycle Objectives (LCO)
  - Stakeholders' commitment to support system architecting
  - Like getting engaged
- Life Cycle Architecture (LCA)
  - Stakeholders' commitment to support full life cycle
  - Like getting married
- Initial Operational Capability (IOC)
  - Stakeholders' commitment to support operations
  - Like having your first child

Win Win Spiral Anchor Points

(Risk-driven level of detail for each element)

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	• Top-level system objectives and scope - System boundary - Environment parameters and assumptions - Evolution parameters • Operational concept - Operations and maintenance scenarios and parameters - Organizational life-cycle responsibilities (stakeholders)	• Elaboration of system objectives and scope of increment • Elaboration of operational concept by increment
System Prototype(s)	• Exercise key usage scenarios • Resolve critical risks	• Exercise range of usage scenarios • Resolve major outstanding risks
Definition of System Requirements	• Top-level functions, interfaces, quality attribute levels, including: - Growth vectors and priorities - Prototypes • Stakeholders' concurrence on essentials	• Elaboration of functions, interfaces, quality attributes, and prototypes by increment - Identification of TBD's((to-be-determined items) • Stakeholders' concurrence on their priority concerns
Definition of System and Software Architecture	• Top-level definition of at least one feasible architecture - Physical and logical elements and relationships - Choices of COTS and reusable software elements • Identification of infeasible architecture options	• Choice of architecture and elaboration by increment - Physical and logical components, connectors, configurations, constraints - COTS, reuse choices - Domain-architecture and architectural style choices • Architecture evolution parameters
Definition of Life-Cycle Plan	• Identification of life-cycle stakeholders - Users, customers, developers, maintainers, interoperators, general public, others • Identification of life-cycle process model - Top-level stages, increments • Top-level WWWWWHH* by stage	• Elaboration of WWWWWHH* for Initial Operational Capability (IOC) - Partial elaboration, identification of key TBD's for later increments
Feasibility Rationale	• Assurance of consistency among elements above - via analysis, measurement, prototyping, simulation, etc. - Business case analysis for requirements, feasible architectures	• Assurance of consistency among elements above • All major risks resolved or covered by risk management plan

*WWWWWHH: Why, What, When, Who, Where, How, How Much

Initial Operational Capability (IOC)

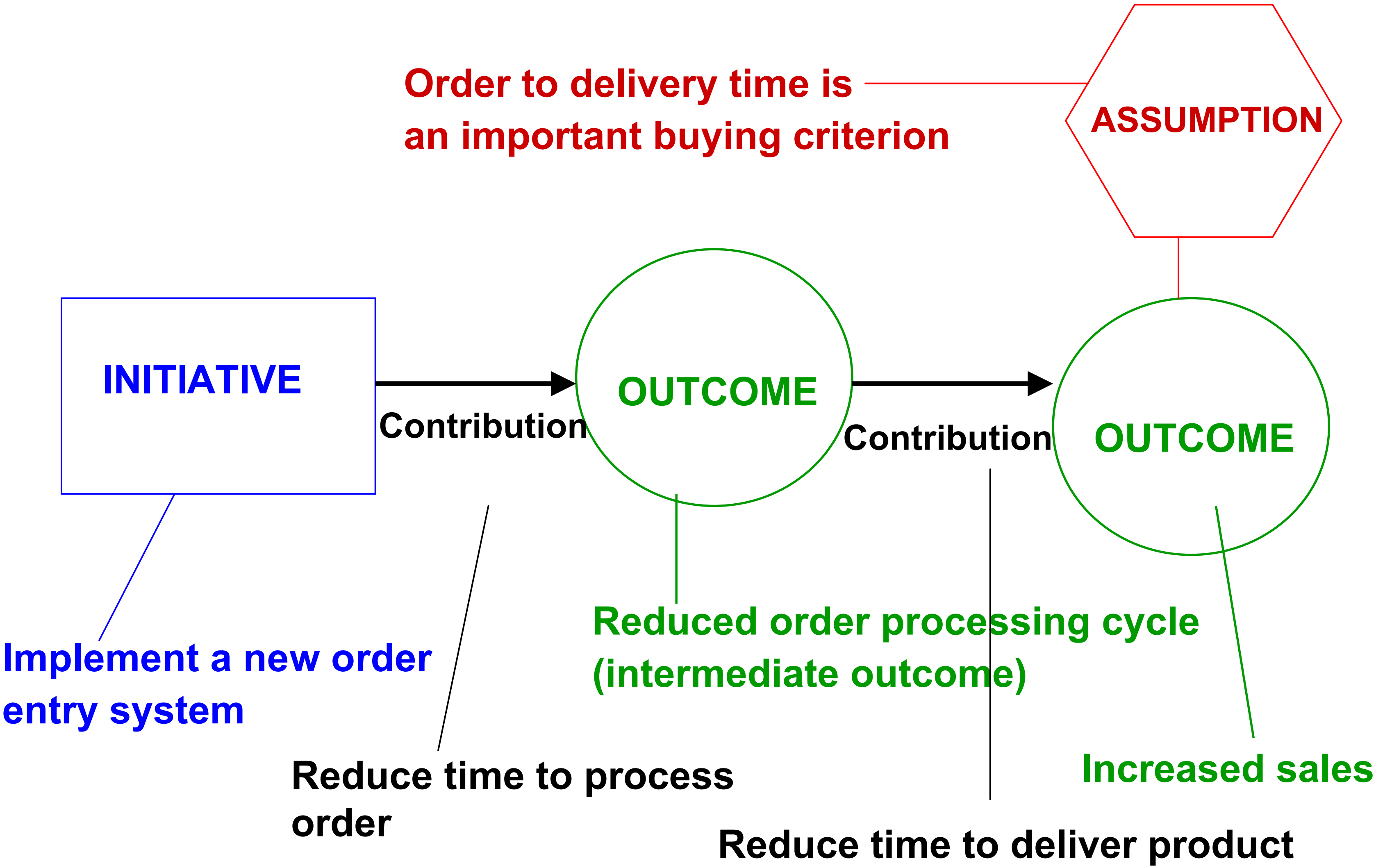
- **Software preparation**
  - Operational and support software
  - Data preparation, COTS licenses
  - Operational readiness testing
- **Site preparation**
  - Facilities, equipment, supplies, vendor support
- **User, operator, and maintainer preparation**
  - Selection, teambuilding, training

The Information Paradox (Thorp)

- No correlation between companies' IT investments and their market performance

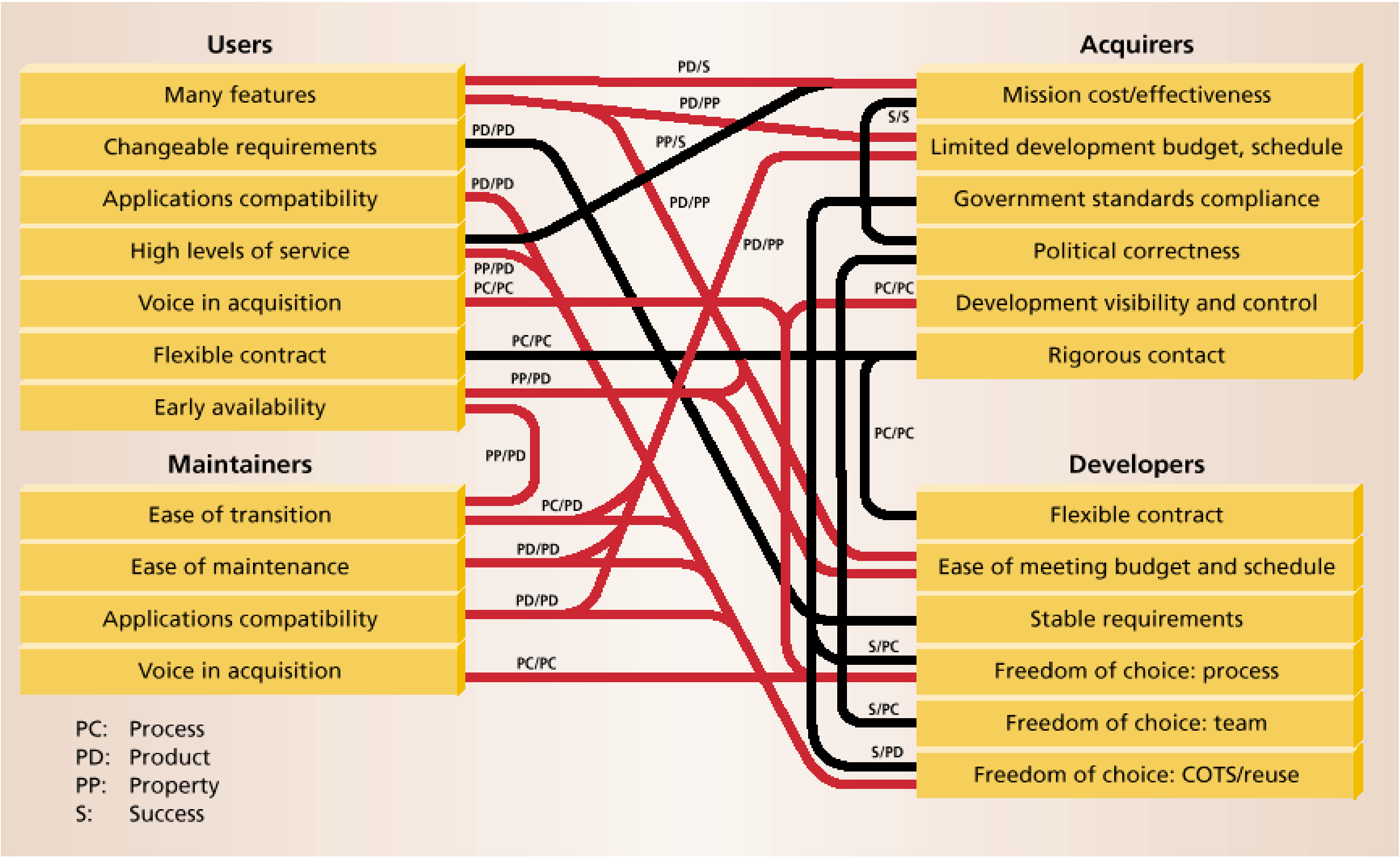
- Field of Dreams
  - Build the (field; software)
  - and the great (players; profits) will come
- Need to integrate software and systems initiatives

DMR/BRA* Results Chain



*DMR Consulting Group's Benefits Realization Approach

The Model-Clash Spider Web: Master Net



EasyWinWin OnLine Negotiation Steps

Review and Expand Negotiation Topics (Group Outliner)
Jointly review and define the scope of the negotiation. Identify the negotiation topics for your EasyWinWin activity.

Brainstorm Stakeholder Interests (Electronic Brainstorming)
Collect ideas about Win Conditions for your EasyWinWin activity

Converge on Win Conditions (Categorizer)
Jointly craft and organize a succinct list of win conditions.

Capture Glossary of Terms (Topic Commenter)
Define important terms of the domain.

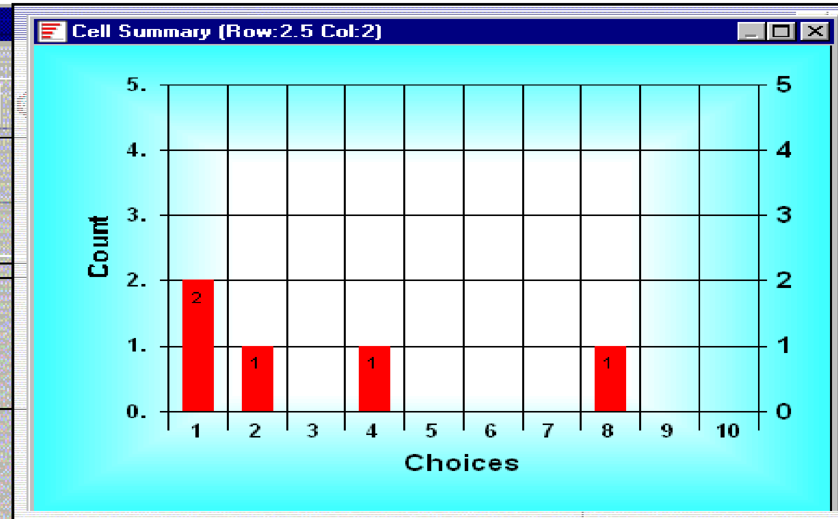
Prioritize Win Conditions (Alternative Analysis)
Determine the business importance and the ease of implementation of all win conditions. Reveal issues and constraints.

WinWin Tree (Group Outliner)
Identify Issues and Options. Negotiate Agreements.

Organize Negotiation Results (Categorizer)
Categorize the results using the negotiation topics.

Red cells indicate lack of consensus.

Oral discussion of cell graph reveals unshared information, unnoticed assumptions, hidden issues, constraints, etc.



	Features	Importance	Ease of Implementation	Total	Mean
2.	Application Capabilities				
2.1	W2 Integrate banner ads with email and chat	10.00	6.50	16.50	8.25
2.2	W3 The banner will provide a link to the universit	10.00	10.00	20.00	10.00
2.3	W4 Interface for advertisers to select their sched	8.67	3.00	11.67	5.83
2.4	W5 Default banner of bookstore if no other events	8.00	10.00	18.00	9.00
2.5	W6 The site management must have a website which	10.00	10.00	20.00	10.00
2.6	W7 Different kinds of advertising, including sales	10.00	10.00	20.00	10.00
2.7	W8 Flexible text on banners	10.00	5.00	15.00	7.50
2.8	W9 Display address of the bookstore, a map of it a	4.00	7.50	11.50	5.75
2.9	W10 Ads must be hyperlinked so that users can clic	7.33	6.00	13.33	6.67
2.10	W11 Link to bookstore site (incl book's prices)	9.33	10.00	19.33	9.67
2.11	W12 Web statistics tracking to determine number of	8.00	4.00	12.00	6.00
2.12	W13 Input of banner contents to admin via email	5.50	10.00	15.50	7.75

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

Hazardous Spiral Look-Alikes

-Cross Talk, May 2001 (http://www.stsc.hill.af.mil/crosstalk)

- Incremental sequential waterfalls with significant COTS, user interface, or technology risks
- Sequential spiral phases with key stakeholders excluded from phases
- Risk-insensitive evolutionary or incremental development
- Evolutionary development with no life-cycle architecture
- Insistence on complete specs for COTS, user interface, or deferred-decision situations
- Purely logical object-oriented methods with operational, performance, or cost risks
- Impeccable spiral plan with no commitment to managing risks

Outline

- What is Software Risk Management?
- What can it help you do?
- When should you do it?
- How should you do it?
- What are its new trends and opportunities?
- Conclusions
- Top-10 Risk Survey

Software Risk Management

- Risk Management
  - Risk Assessment
    - Risk Identification
      - Checklists
      - Decision driver analysis
      - Assumption analysis
      - Decomposition
    - Risk Analysis
      - Performance models
      - Cost models
      - Network analysis
      - Decision analysis
      - Quality factor analysis
    - Risk Prioritization
      - Risk exposure
      - Risk leverage
      - Compound risk reduction
  - Risk Control
    - Risk mgmt Planning
      - Buying information
      - Risk avoidance
      - Risk transfer
      - Risk reduction
      - Risk element planning
      - Risk plan integration
    - Risk Resolution
      - Prototypes
      - Simulations
      - Benchmarks
      - Analyses
      - Staffing
    - Risk Monitoring
      - Milestone tracking
      - Top-10 tracking
      - Risk reassessment
      - Corrective action

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

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?

Risk ID: Examining Decision Drivers

- Political versus Technical
  - Choice of equipment
  - Choice of subcontractor
  - Schedule, Budget
  - Allocation of responsibilities
- Marketing versus Technical
  - Gold plating
  - Choice of equipment
  - Schedule, budget
- Solution-driven versus Problem-driven
  - In-house components, tools
  - Artificial intelligence
  - Schedule, Budget
- Short-term versus Long-term
  - Staffing availability versus qualification
  - Reused software productions engineering
  - Premature SRR, PDR
- Outdated Experience

Potential Win-lose, Lose-lose Situations

	"Winner"	Loser
• Quick, cheap, sloppy product	• Developer • Customer	• User
• Lots of bells and whistles	• Developer • Customer	• Customer
• Driving too hard a bargain	• Developer • Customer	• Developer

Actually, nobody wins in these situations

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

The Top Ten Software Risk Items

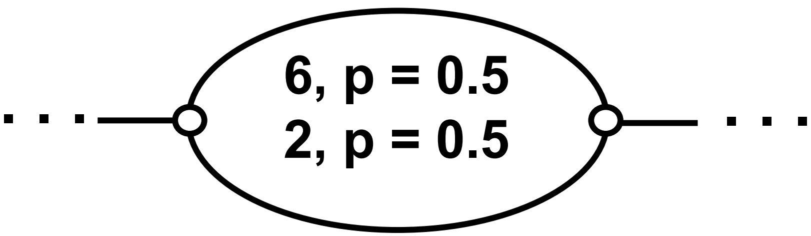
Risk Item	Risk Management Techniques
1. Personnel Shortfalls	Staffing with top talent; key personnel agreements; incentives; team-building; training; tailoring process to skill mix; peer reviews
2. Unrealistic schedules and budgets	Business case analysis; design to cost; incremental development; software 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

Network Schedule Risk Analysis

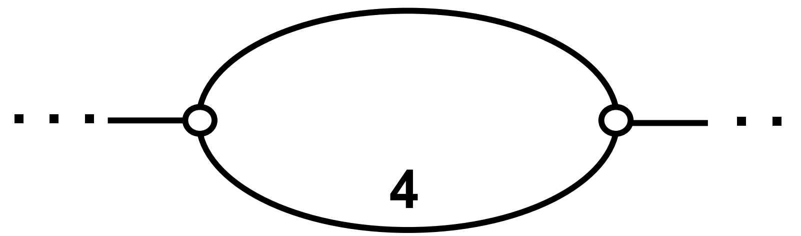
2, p = 0.5



6, p = 0.5

4 Equally Likely Outcomes

4 months



EV = 4 months

2
2 → 2

2
6 → 6

6
2 → 6

6
6 → 6

20

$$EV = \frac{20}{4} = 5 \text{ MONTHS}$$

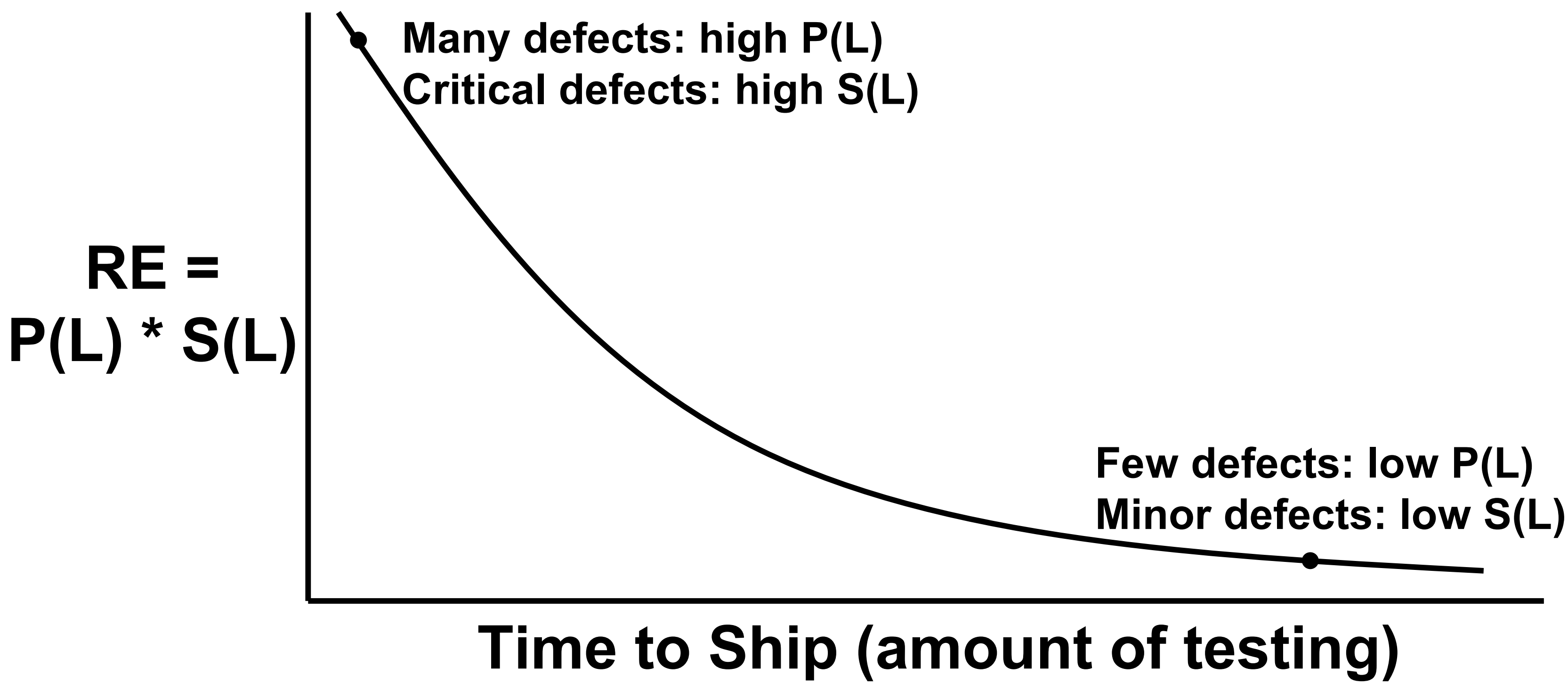
Using Risk to Determine "How Much Is Enough" - testing, planning, specifying, prototyping…

- 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

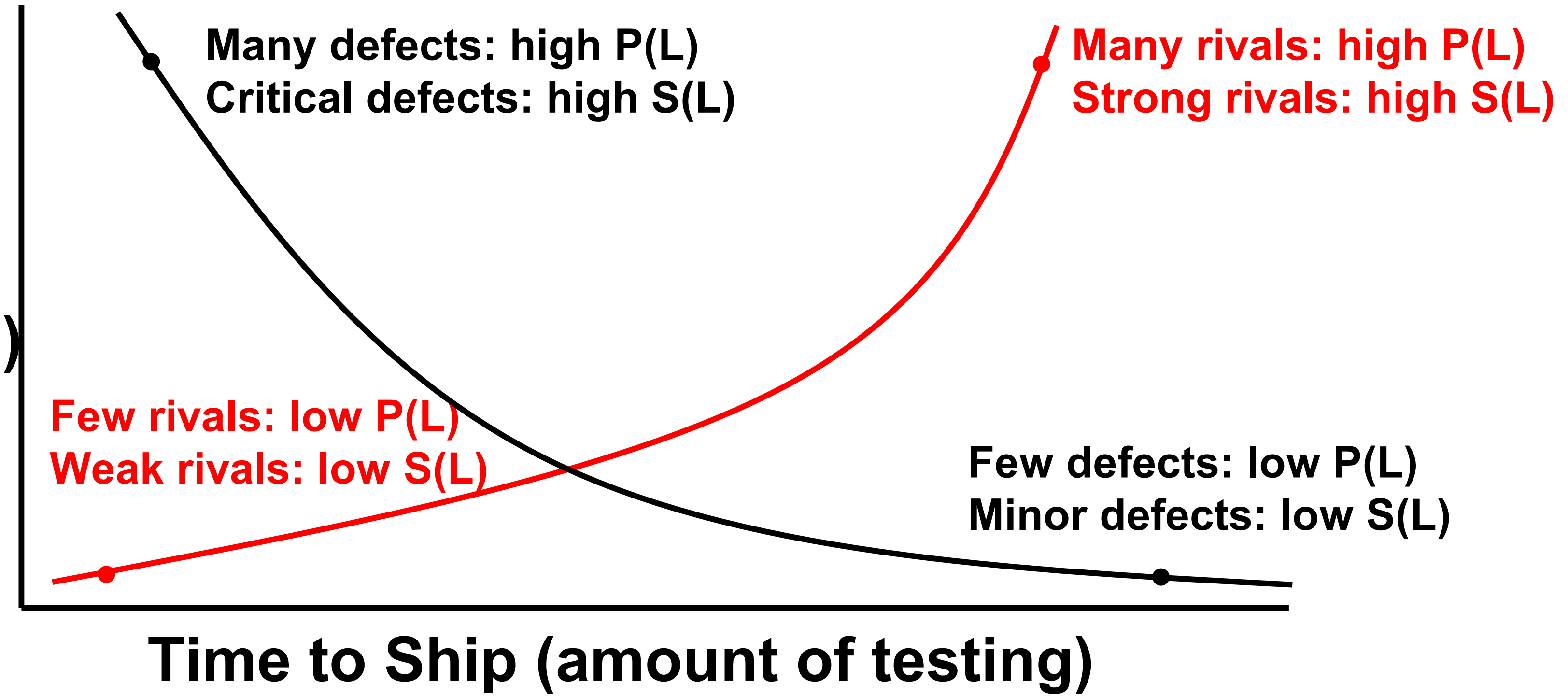
- Loss due to unacceptable dependability



Example RE Profile: Time to Ship

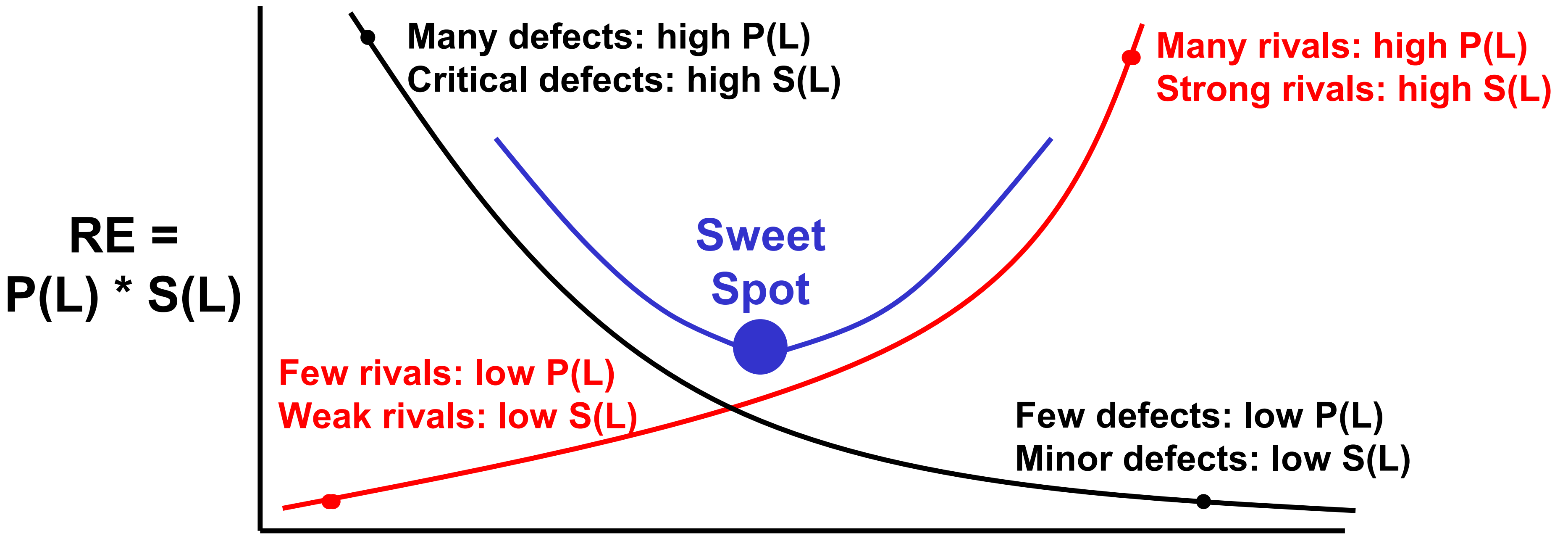
- Loss due to unacceptable dependability
- Loss due to market share erosion

RE =
P(L) * S(L)



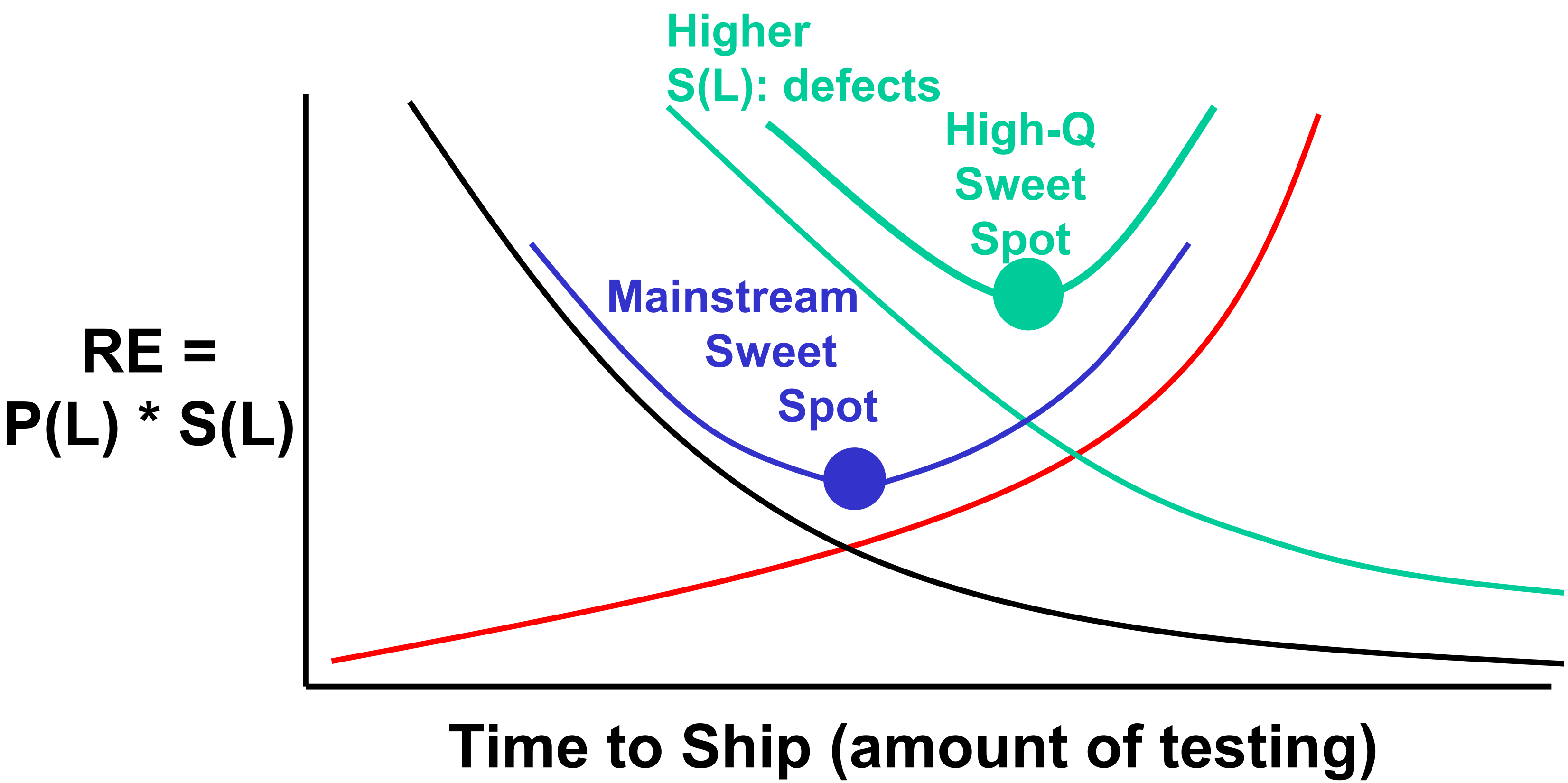
Example RE Profile: Time to Ship

- Sum of Risk Exposures

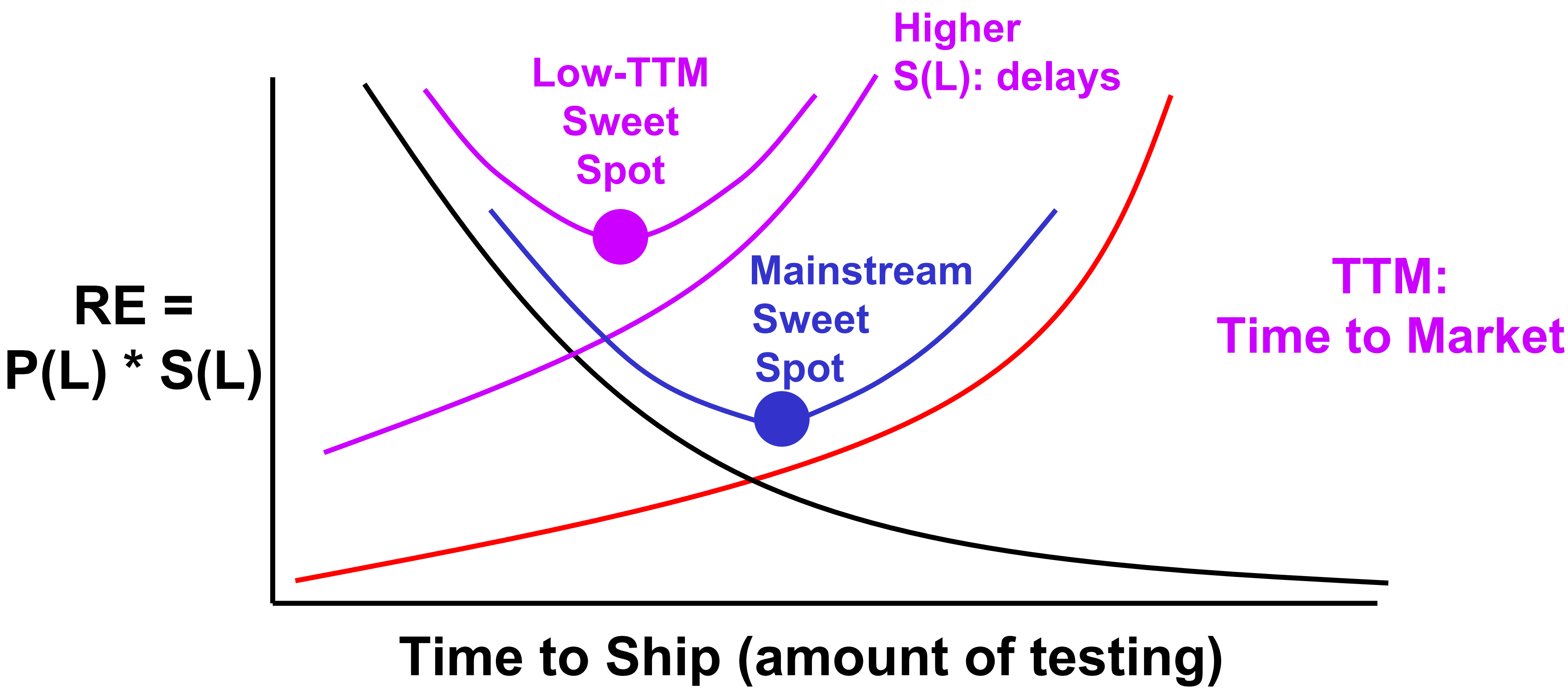


Time to Ship (amount of testing)

Comparative RE Profile:
Safety-Critical System

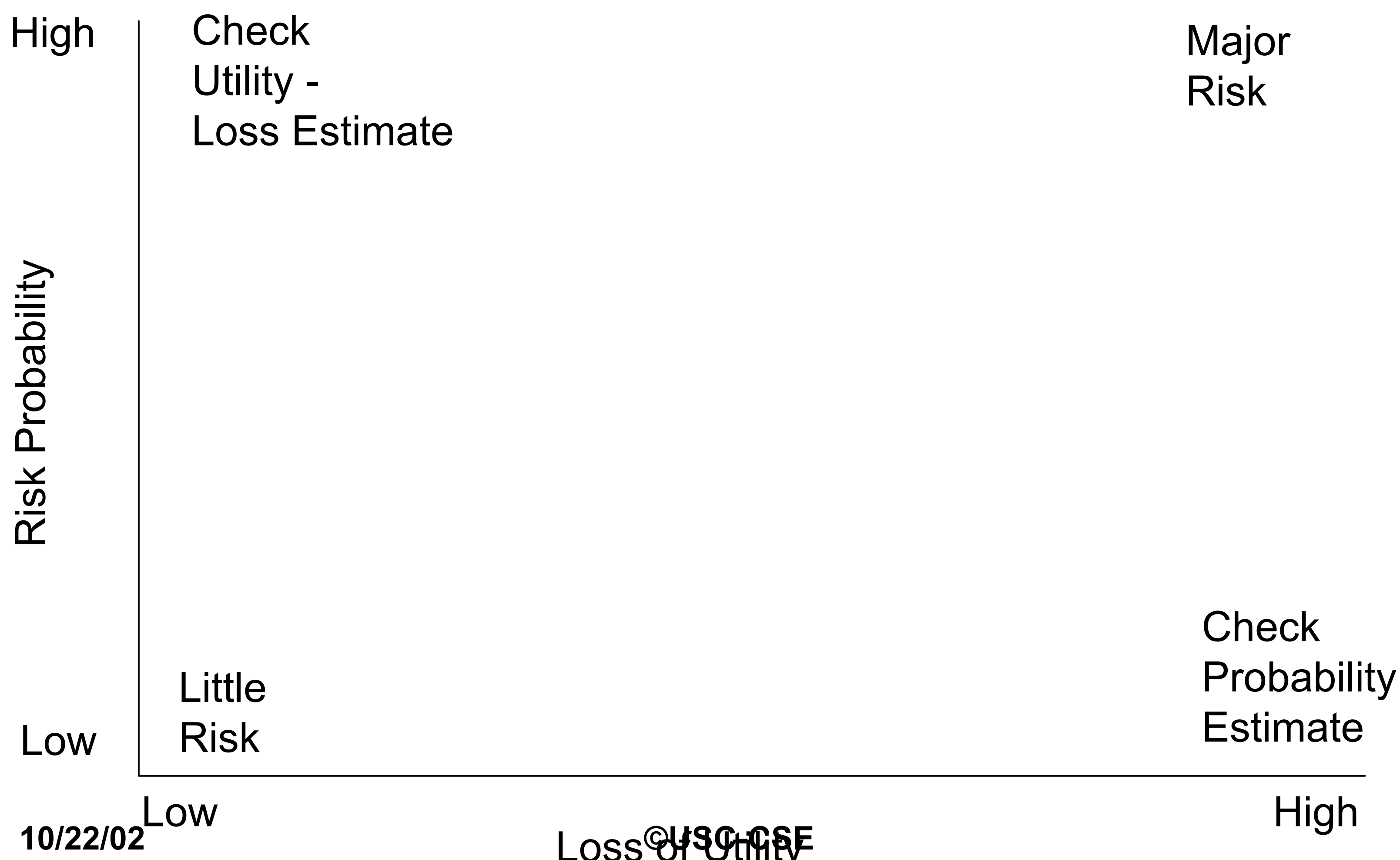


Comparative RE Profile: Internet Startup



Prioritizing Risks: Risk Exposure

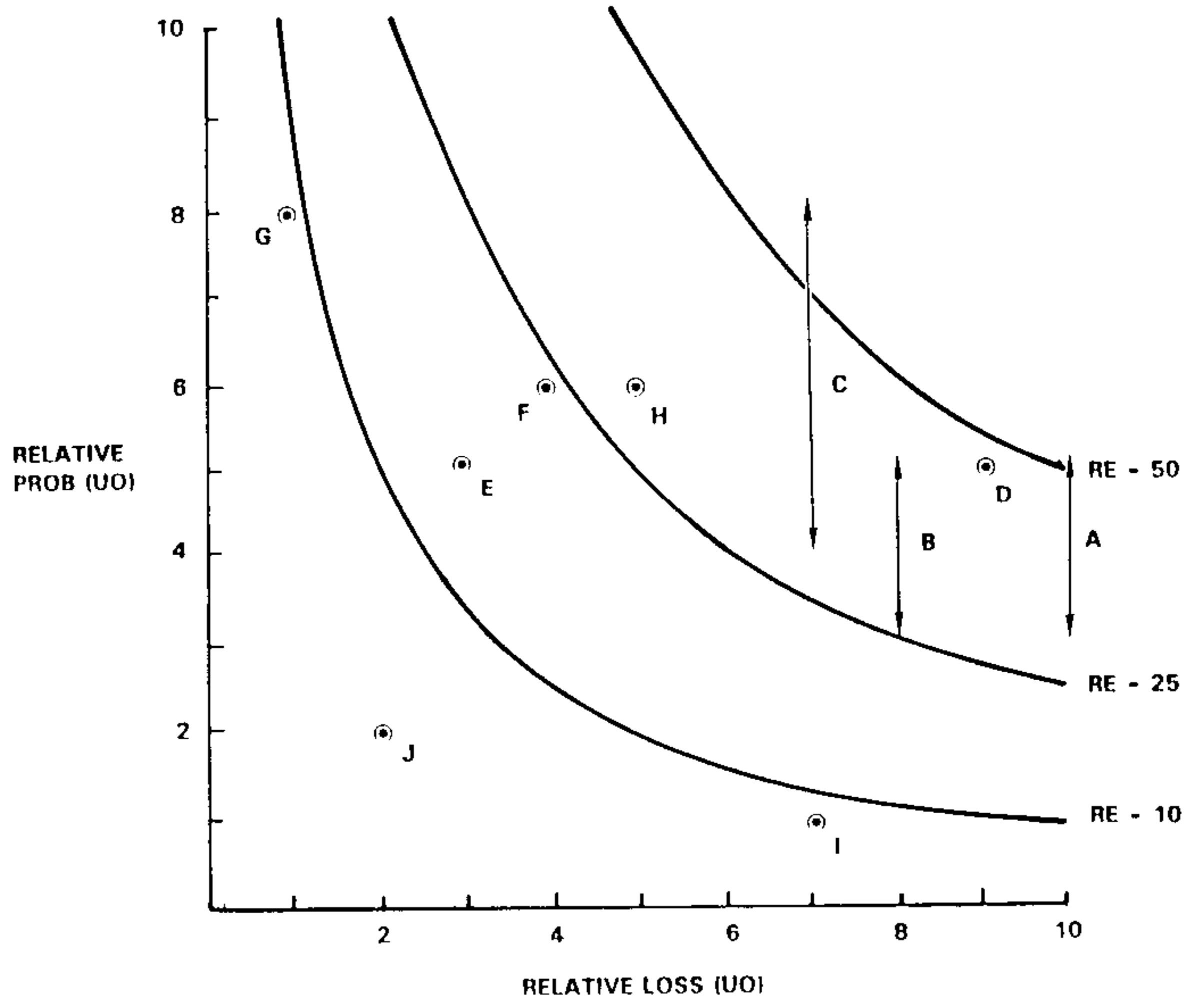
Risk Exposure - (Probability) (Loss of Utility)



Risk Exposure Factors
(Satellite Experiment Software)

Unsatisfactory Outcome (UO)	Prob (UO)	Loss (UO)	Risk Exposure
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 $(\text{UO})_B$ RE_B	\$20M 0.2 \$4M	\$20M 0.2 \$4M
PROB $(\text{UO})_A$ RE_A	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
  - Risk Assessment
    - Risk Identification
      - Checklists
      - Decision Driver Analysis
      - Assumption Analysis
      - Decomposition
    - Risk Analysis
      - Performance Models
      - Cost Models
      - Network Analysis
      - Decision Analysis
      - Quality Factor Analysis
    - Risk Prioritization
      - Risk Exposure
      - Risk Leverage
      - Compound Risk Reduction
  - Risk Control
    - Risk Mgmt. Planning
      - Buying Information
      - Risk Avoidance
      - Risk Transfer
      - Risk Reduction
      - Risk Element Planning
      - Risk Plan Integration
    - Risk Resolution
      - Prototypes
      - Simulations
      - Benchmarks
      - Analyses
      - Staffing
    - Risk Monitoring
      - Milestone Tracking
      - Top-10 Tracking
      - Risk Reassessment
      - Corrective Action

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)

  <u>$10K</u> - 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 manger-priority items

Risk Reassessment

Corrective Action

Project Top 10 Risk Item List: Satellite Experiment Software

Risk Item	Mo. Ranking This	Mo. Ranking Last	Mo. Ranking #Mo.	Risk Resolution Progress
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

Complex Systems of Systems (CSOS): Software Benefits, Risks, and Strategies

- CSOS characteristics and software benefits
- Software benefits and risks
- Software risks and strategies
- Conclusions

CSOS Characteristics and Software Benefits (relative to hardware)

CSOS Characteristics	Software Benefits (relative to hardware)
• Many component systems and contractors with wide variety of users and usage scenarios—including legacy systems	• Ease of accommodating many combinations of options • Ease of tailoring various system and CSOS versions
• Need to rapidly accommodate frequent changes in missions, environment, technology, and interoperating systems	• Rapidly adaptable • Rapidly upgradeable • Near-free COTS technology upgrades
• Need for early capabilities	• Flexibility to accommodate concurrent and incremental development

CSOS Software Benefits, Risks, and Strategies

- Accommodating many combinations of options
  - Development speed; integration; cross-system KPP's
- Accommodating many combinations of systems and contractors
  - Subcontractor specifications, incompatibilities, change management
- Rapid tailoring and upgrade of many combinations of options
  - Version control and synchronous upgrade propagation
- Flexibility, rapid adaptability, incremental development
  - Subcontractor chain increment synchronization; requirements and architecture volatility
- Near-free COTS technology upgrades
  - COTS upgrade synchronization; obsolescence; subcontractor COTS management
- Compound risks

Many CSOS Options and Software Development Speed

- Risk #1: Limited speed of CSOS Software Development
  - Many CSOS scenarios require close coupling of complex software across several systems and subsystems
  - Well-calibrated software estimation models agree that there are limits to development speed in such situations
  - Estimated development schedule in months for closely coupled SW with size measured in equivalent KSLOC (thousands of source lines of code):

    Months =~ 5 * $\sqrt[3]{\text{KSLOC}}$

- KSLOC	300	500	1000	5000
- Months	33	40	50	85

Risk #1: Limited Speed of CSOS Software Development

- Strategy #1a. Architect the CSOS software to be able to develop independent software units in parallel and have them cleanly integrate at the end.
- Strategy #1b. Focus scope of Initial Operational Capability (IOC) on top-priority, central-risk elements.
  - Prioritize the IOC feature content to enable a Schedule-as-Independent-Variable (SAIV)* development process:
    - Estimate maximum size of software buildable with high confidence within available schedule
    - Define core-capability IOC content based on priorities, end-to-end usability, and need for early development of central-risk software
    - Architect for ease of adding and dropping borderline-priority features
    - Monitor progress; add or drop features to meet schedule

* Can be used for a Cost-as-Independent Variable (CAIV) process as well

Many CSOS Options and Software Integration

- Risk #2. Critical dimensions of software integration may begin late and cause overruns. CSOS software needs to be integrated:
  – by CSC/CSCI hierarchy
  – by cross-IPT software exercises which incrementally ingrate the software
  – by critical threads and scenarios (nominal and off-nominal);
  – by number of platforms and processors involved; by homogeneous-to-heterogeneous platforms;
  – by friendly-to-adversarial environment; etc.
- Strategy #2a. Reflect all of these software integration dimensions in the system integration plans for each Build in each Increment.
- Strategy #2b. Integrate early via architectural analysis and best-possible versions of daily-build-and-test strategies. This involves significant cross-IPT coordination, and subcontractor provisions and procedures to provide intermediate versions of software for early integration testing.

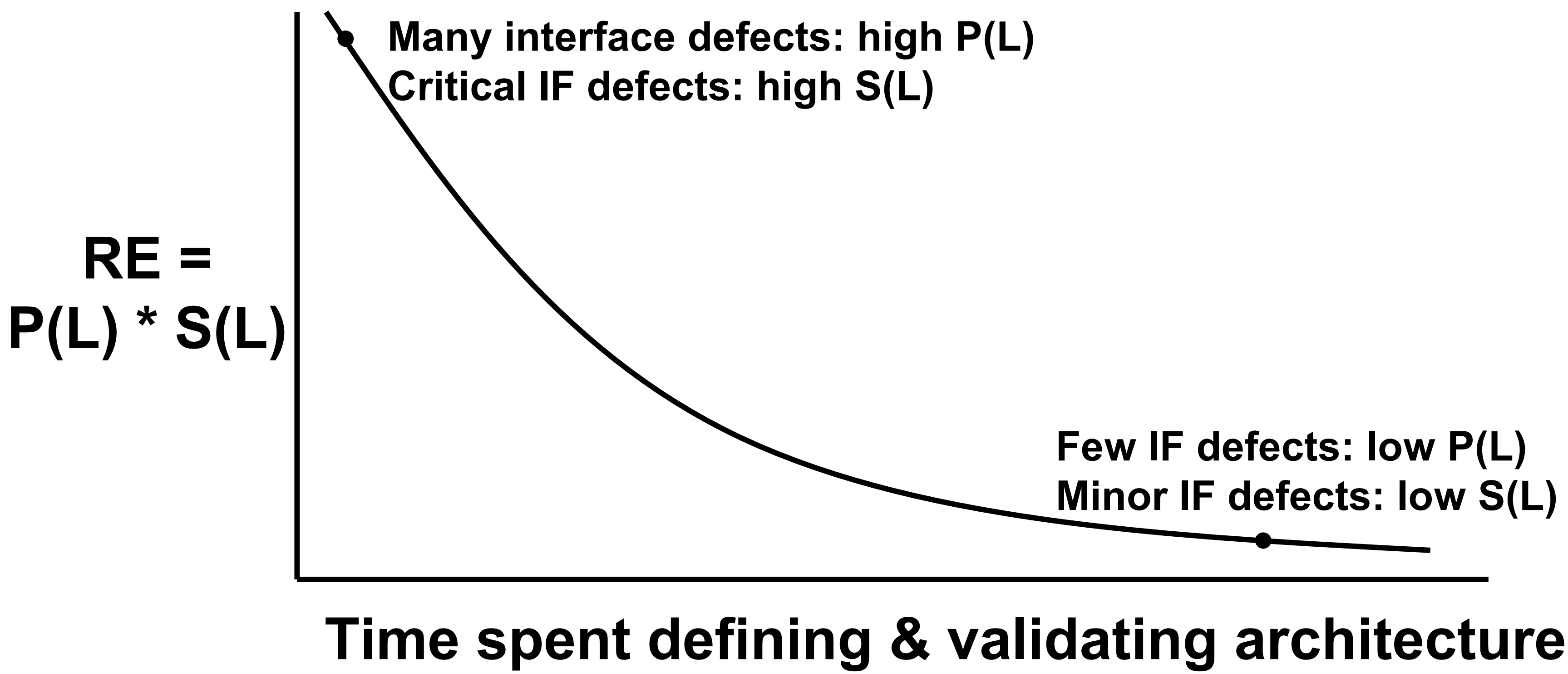
Many CSOS Options and Cross-System KPP's

- **Risk #3. Many CSOS software Key Performance Parameter (KPP) tradeoffs may be cross-cutting, success-critical, difficult to analyze, and incompletely formulated.**
- **Strategy #3. Focus significant modeling, simulation, and execution analyses on success-critical software KPP tradeoff issues. These should particularly include critical-infrastructure KPP tradeoff analyses, and adversary-oriented analyses and exercises.**

How Soon to Define Subcontractor Interfaces?

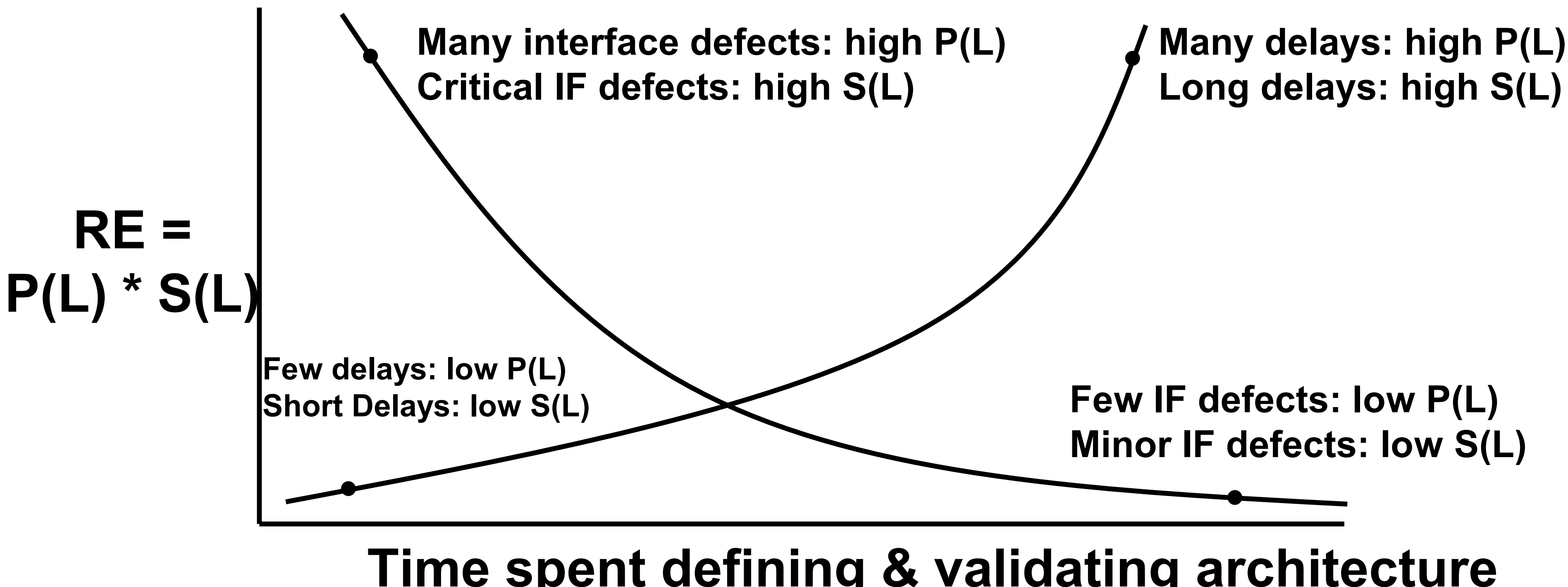
Risk exposure RE = Prob(Loss) * Size(Loss)

-Loss due to rework delays

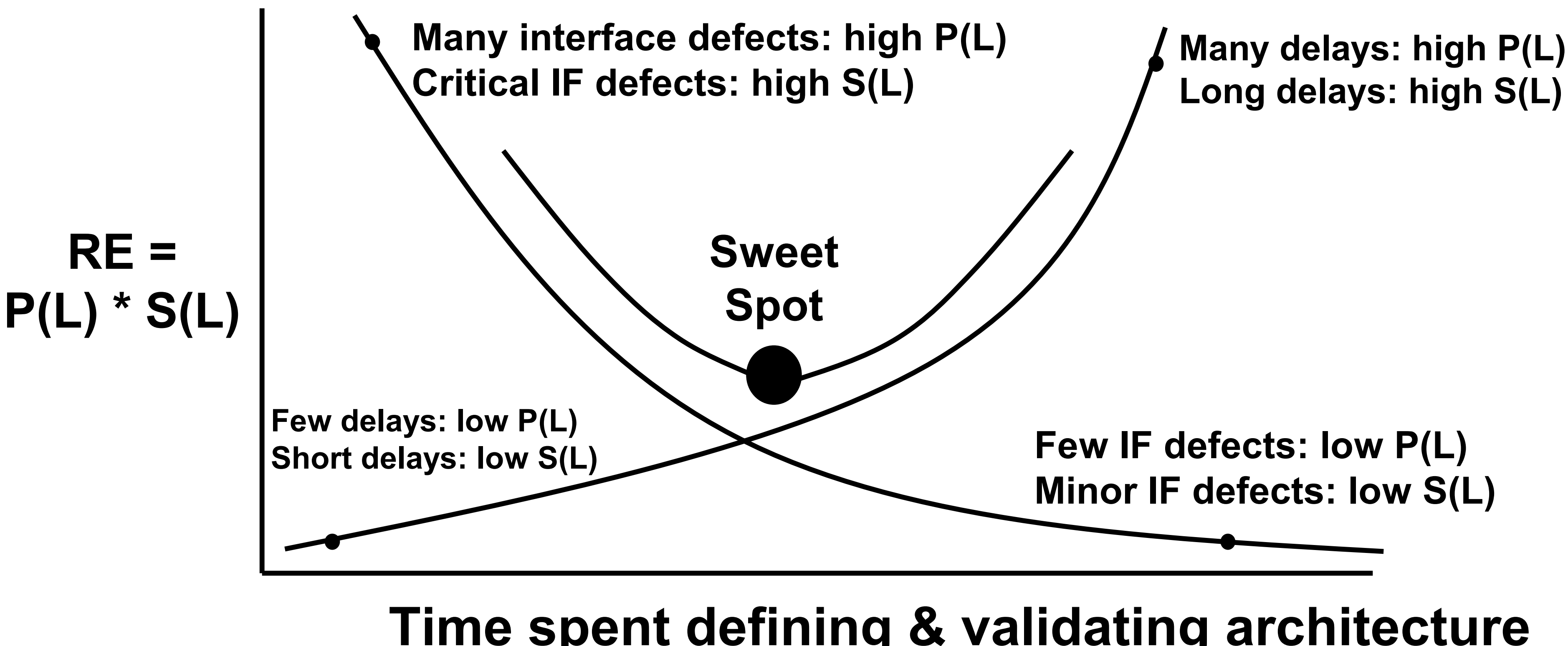


How Soon to Define Subcontractor Interfaces?

- Loss due to rework delays
- Loss due to late subcontact startups

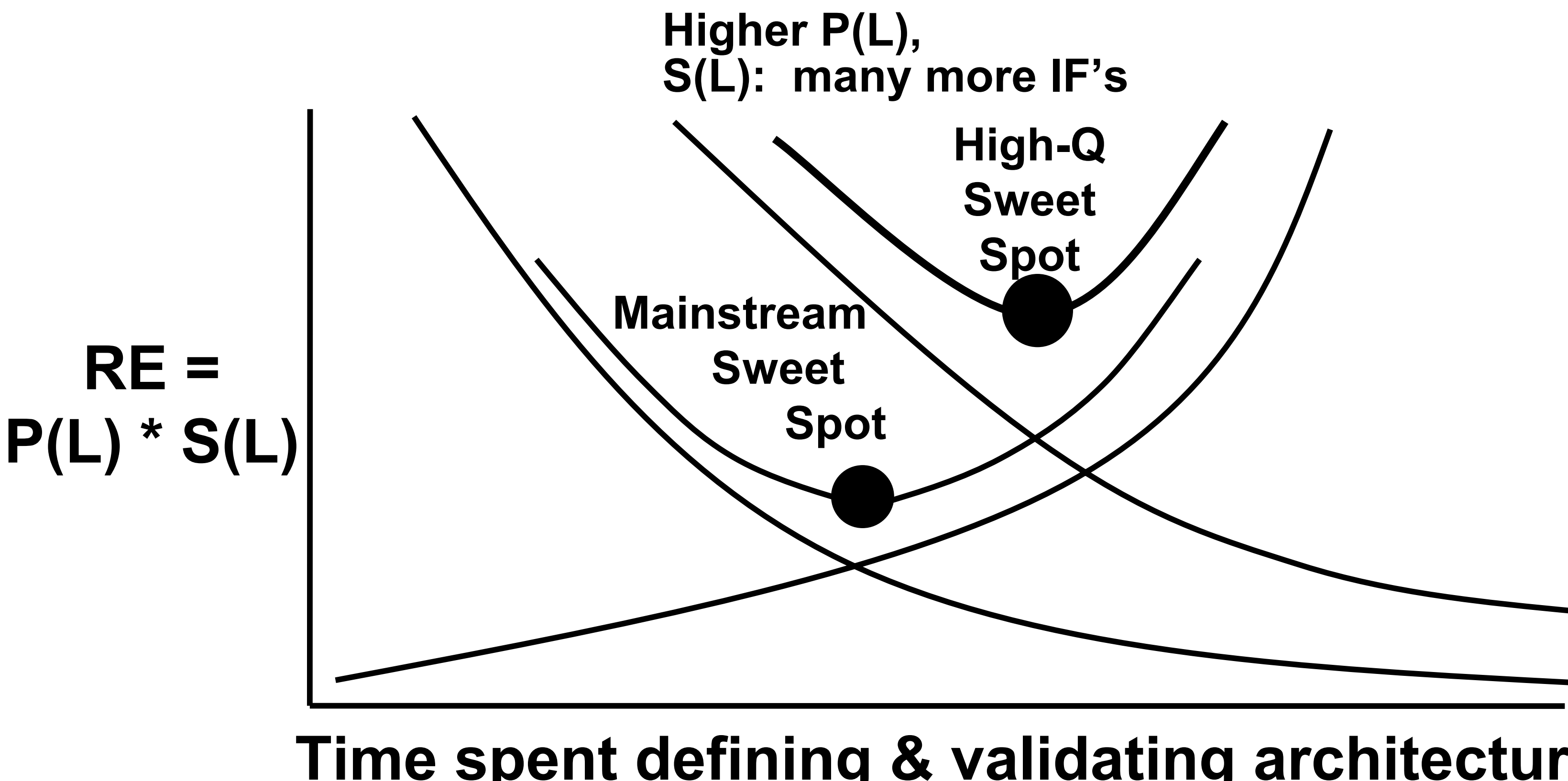


How Soon to Define Subcontractor Interfaces?
- Sum of Risk Exposures



How Soon to Define Subcontractor Interfaces?

-Very Many Subcontractors



Risk #4: Delayed CSOS availability due to many-subcontractor IF rework

Strategy #4: Invest more time in architecture definition

Rapid, Synchronous Software Upgrades

- Risk #5. Out-of-synchronization software upgrades will be a major source of operational losses
  - Software crashes, communication node outages, out-of-synch data, mistaken decisions
  - Extremely difficult to synchronize multi-version, distributed, mobile-platform software upgrades
  - Especially if continuous-operation upgrades needed
- Strategy #5a. Architect software to accommodate continuous-operation, synchronous upgrades
  - E.g., parallel operation of old and new releases while validating synchronous upgrade
- Strategy #5b. Develop operational procedures for synchronous upgrades in software support plans
- Strategy #5c. Validate synchronous upgrade achievement in operational test & evaluation

Rapid Adaptability to Change: Architecture

- **Risk #6. Software architecture may be over-optimized for performance vs. adaptability to change**
- **Strategy #6. Modularize software architecture around foreseeable sources of change**

  -Identify foreseeable sources of change

    -Technology, interfaces, pre-planned product improvements

  -Encapsulate sources of change within software modules

    -Change effects confined to single module

  -Not a total silver bullet, but incrementally much better

Rapid Adaptability to Change: Evolving Software Architecture

- **Risk #7. Software architecture will need to change & adapt to rapidly changing priorities and architecture drivers**
  - **new COTS releases;**
  - **evolving enterprise standards and policies;**
  - **emerging technologies and competitor threats**
- **Strategy #7a. Organize CSOS software effort to ensure the ability to rapidly analyze, develop, & implement software architecture changes. Empower a focal-point integrator of the software architecture and owner of the critical software infrastructure.**
- **Strategy #7b. Raise the organizational level of the owner of the software architecture & infrastructure (and the owner of CSOS software integration and test) to a very high level in the CSOS organizational structure.**

Rapid Adaptability to Change: Architecture Evolution and Subcontracting

- **Risk #8. The CSOS software architecture will inevitably change. Inflexible subcontracting will be a major source of delays and shortfalls.**
- **Strategy #8. Develop subcontract provisions enabling flexibility in evolving deliverables. Develop an award fee structure and procedures based on objective criteria for evaluating subcontractors' performance in:**
  - -Schedule Preservation
  - -Cost Containment
  - -Technical Performance
  - -Architecture and COTS Compatibility
  - -Continuous Integration Support
  - -Program Management
  - -Risk Management

Flexibility and Rapid Adaptability to Change: Definitive but Flexible Software Milestones

- **Risk #9. Flexible CSOS software process will be overconstrained by too-tight milestone exit criteria, but will be hard to monitor and can run out of control with too-loose milestone exit criteria.**
- **Strategy #9. Use the WinWin Spiral Model Life Cycle Objectives (LCO) and Live Cycle Architecture (LCA) milestone criteria. They provide risk-driven degrees of milestone deliverable content, and require demonstration of compatibility and feasibility via a Feasibility Rationale deliverable.**

Need Concurrently Engineered Milestone Packages

–Life Cycle Objectives (LCO); Life Cycle Architecture Package (LCA)

Operational Concept	**•Elaboration of system objectives and scope by increment** **•Elaboration of operational concept by increment**
System Prototype(s)	**•Exercise range of usage scenarios** **•Resolve major outstanding risks**
System Requirements	**•Elaboration of functions, interfaces, quality attributes, and prototypes by increment** **- Identification of TBD's (to be determined items)** **•Stakeholders' concurrence on their priority concerns**
System and Software Architecture	**•Choice of architecture and elaboration by increment** **- Physical and logical components, connectors, configurations, constraints** **-COTS, reuse choices** **- Domain architecture and architectural style choices** **• Architecture evolution parameters**
Life-Cycle Plan	**• Elaboration of WWWWWHH* for initial Operational Capability (IOC)** **- Partial elaboration, identification of key TBD's for later increments**
Feasibility Rationale	**•Assurance of consistency among elements above** **•All major risks resolved or covered by risk management plan.**

*WWWWWHH: Why, What, When, Who, Where, How, How Much

LCO (MS A) and LCA (MS B) Pass/Fail Criteria

– Cross Talk, December 2001

A system built to the given architecture will

- **Support the operational concept**
- **Satisfy the requirements**
- **Be faithful to the prototype(s)**
- **Be buildable within the budgets and schedules in the plan**
- **Show a viable business case**
- **Establish key stakeholders' commitment to proceed**

LCO: True for at least one architecture
LCA: True for the specific life cycle architecture;
All major risks resolved or covered by a risk management plan

Near-Free COTS Technology Upgrades: COTS Upgrade Synchronization and Obsolescence

- **Risk #10. If too many COTS software products, versions, and releases are contained in the various CSOS software elements, the software will not integrate. If unsupported COTS software releases are contained in the software elements, the integration will suffer serious delays. Given that COTS software products typically undergo new releases every 8-9 months, and become unsupported after 3 new releases, there are high risks that a tightly-budgeted delivery on a software subcontract spanning over 30 months will include unsupported COTS software products.**
- **Strategy #10a. Develop a management tracking scheme for all COTS software products in all CSOS software elements, and a strategy for synchronizing COTS upgrades.**
- **Strategy #10b. Develop contract and subcontract provisions and incentives to ensure consistency and interoperability across contractor and subcontractor-delivered COTS software products, and to ensure that such products are recent-release versions.**

CSOS Compound Risks

- **Risk #11. Serious inter-system compound risks will be discovered late. Compound risks are very frequently architecture-breakers, budget-breakers, and schedule-breakers. Examples include closely-couples immature technologies and closely-coupled, ambitious critical path tasks.**
- **Strategy #11a. Establish a hierarchical software risk tracking compound risk assessment scheme. The top level of the hierarchy would be the Top-10 system-wide software risks. These would tier down to system-level and subcontractor-level Top-10 risk lists. These are valuable both for overall software risk management and for compound risk assessment.**
- **Strategy #11b. Develop high-priority plans to decouple high-risk elements and to reduce their risk exposure.**
- **Strategy #11c. Establish a CSOS Software Risk Experience Base. This is extremely valuable in avoiding future instances of previously experienced risks.**

Conclusions: CSOS and Software

- Software is a critical enabling technology for Complex Systems of Systems(CSOS)
  - -It provides the means for dealing with many CSOS risks and opportunities
  - -Particularly those involving interoperablity and rapid adaptability to change
- Software's CSOS benefits come with their own risks
  - -Some of these are rarely encountered in hardware-intensive acquisitions
  - -They particularly affect ambitious CSOS schedules
- Strategies for dealing with CSOS software risks are becoming available
  - -Some require changes in traditional acquisition practices
- CSOS software risks are often cross-cutting and CSOS-wide
  - -Successfully resolving them often requires informed, pro-active, high-level management authority

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

Survey: Top 10 Risk Items, 1995 and 2002

1995		2002
1.	Personnel shortfalls	______ Straining computer science
2.	Schedules, budgets, process	______ Externally-performed tasks
3.	COTS, external components	______ Legacy software
4.	Requirements mismatch	______ Requirements changes
5.	User interface mismatch	______ Architecture, performance, quality, distribution/mobility
6.	Architecture, performance, quality, distribution/mobility	______ User interface mismatch
7.	Requirements changes	______ Requirements mismatch
8.	Legacy software	______ COTS, external components
9.	Externally-performed tasks	______ Schedules, budgets, process
10.	Straining computer science	______ Personnel shortfalls
		______ Rapid change
		______ Systems of systems
		______ Weak business case

Enter numbers from 1-10

Outline

- What is Software Risk Management?
- What can it help you do?
- When should you do it?
- How should you do it?
- ➡ What are its new trends and opportunities?
- Conclusions
- Top-10 Risk Survey