# 소프트웨어공학 원리 (SEP521)

## Risk Management - I

**Jongmoon Baik**

# Questions to Consider

- Why am I talking to you about risk management?

- How many of you manage risks continuously or have you managed them throughout a project before?

If you don actively attack the risks?

The risks will actively attack you !

–Tom Gilb –

# What is A Risk?

- "The chance or possibility of suffering loss, injury, damage, or harm" – Webster's dictionary, 1981

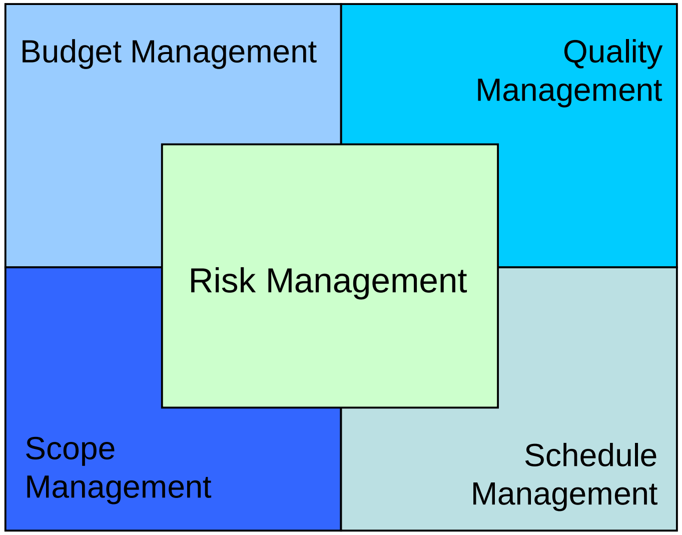
- "The measure of the probability and severity of adverse effects" – William Lawrance, "Of Acceptable Risk", 1976

- "problems that could cause some loss or threaten the success of projects, but which has not happened yet"
  - http://www.processimpact.com/articles/risk_mgmt.html

- No universally accepted definition

- All definitions share the following characteristics:
  - *Uncertainty* – an event may or may not happen
  - *Loss* – an event has unwanted consequences or losses

# Risk and Project



Risk Management is aimed addressing the risks within the project management environment .. Not just the technical risks

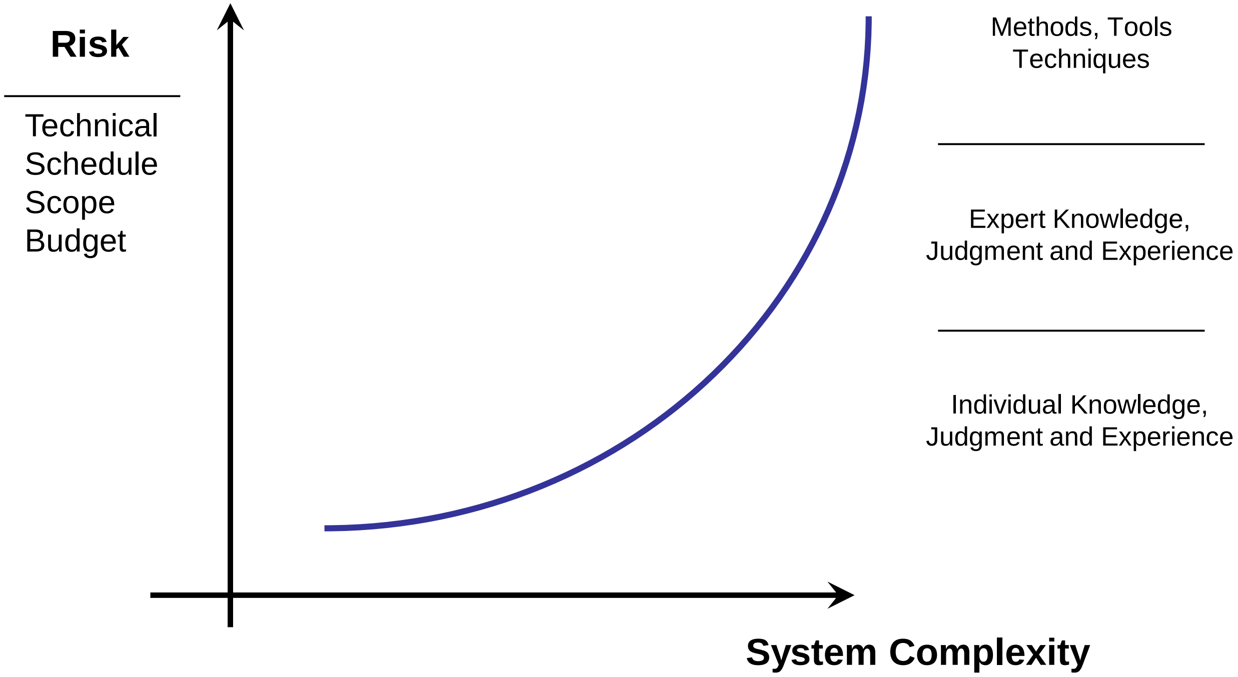
# Is Risk Necessarily Bad?

"Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning.

But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity."

# The Need to Manage Risk

- Project risk increases as the systems become more complex



# Risk Management

- Get recognized as best practice to reduce the surprise factor
  - Structured risk mgmt: to peek over the horizon at the traps that might be looming,
    - Provides visibility in threats to project success
    - Control risks across multiple projects
  - Take actions to minimize the likelihood or impact of these potential problems

- Inefficient risk management cause:
  - A continual state of project instability
  - Constant fire-fighting
  - Multiple schedule slippage

"Deal with a concern before it becomes a crisis"

# 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?

- 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

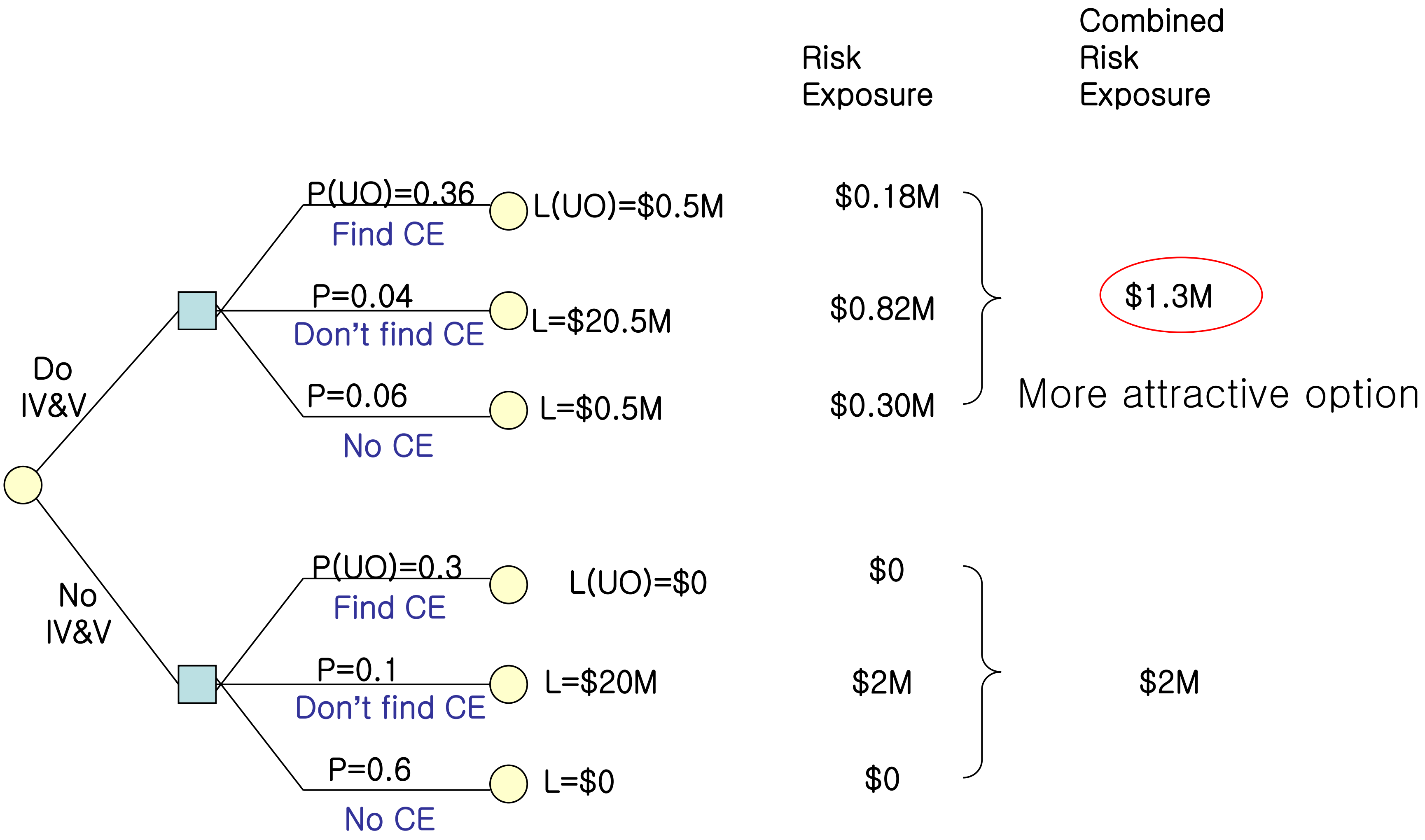
# Earlier, This Problem

- 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) [or P(UO)]
  - 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) [or L(UO)]

- Risk Exposure (RE)

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

# Risk-Analysis: Decision



Risk Exposure

Combined Risk Exposure

Do IV&V

P(UO)=0.36 — Find CE — L(UO)=$0.5M — $0.18M

P=0.04 — Don’t find CE — L=$20.5M — $0.82M

P=0.06 — No CE — L=$0.5M — $0.30M

$1.3M

More attractive option

No IV&V

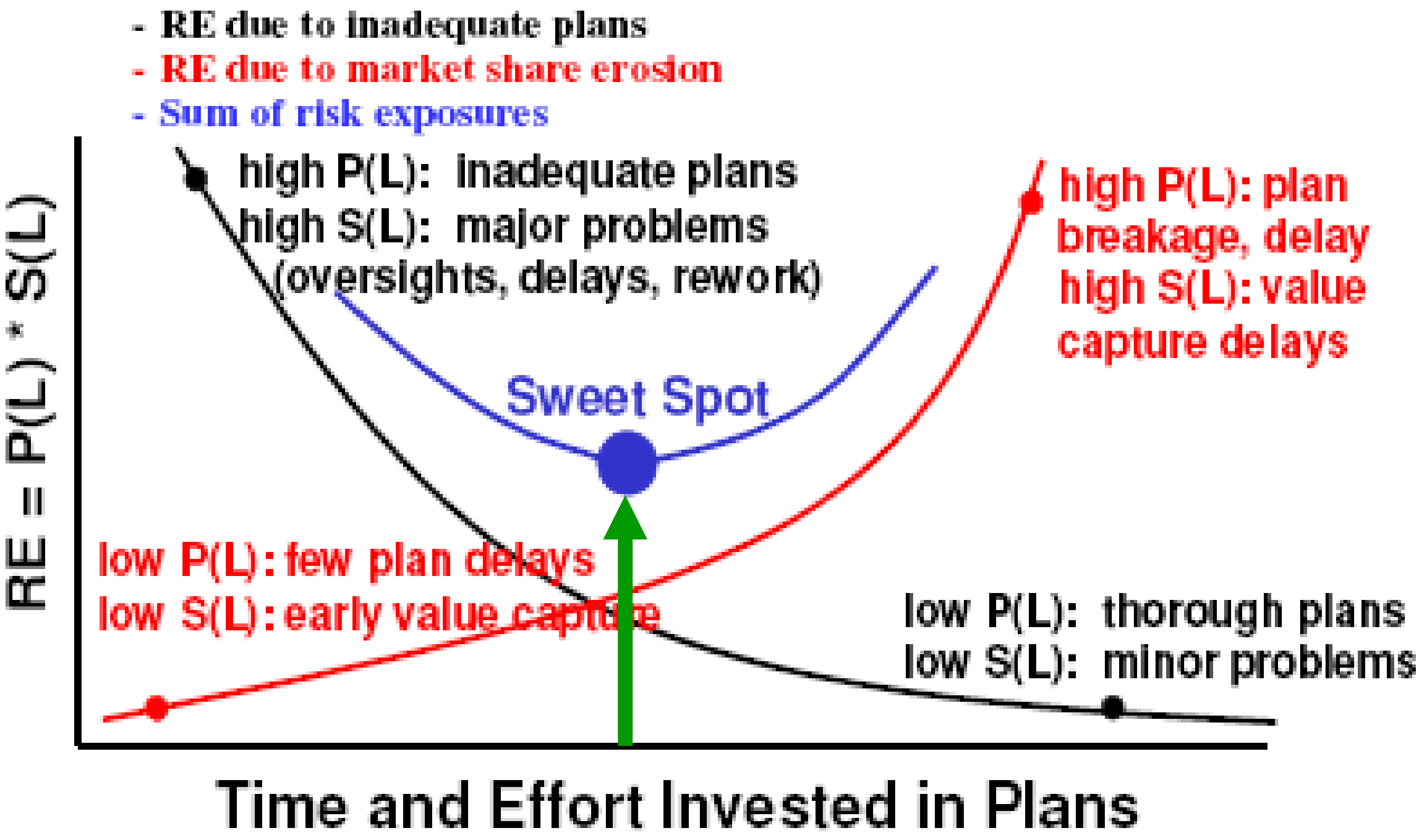
P(UO)=0.3 — Find CE — L(UO)=$0 — $0

P=0.1 — Don’t find CE — L=$20M — $2M

P=0.6 — No CE — L=$0 — $0

$2M

# Using Risk to Determine "HOW MUCH ENOUGH"



# RISK vs. PROBLEM

- Example:
  - A risk: threat of your top technical people being hired away by the competition
  - A problem: unable to hire right skilled staff
- Current, real problems
  - Requires prompt corrective actions
- Looming risk
  - Dealt with several different ways

# How to Deal with Risks?

- Buying Information
- Risk Avoidance
- Risk Transfer
- Risk Reduction
- Risk Acceptance

# Risk Management Strategies - I

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

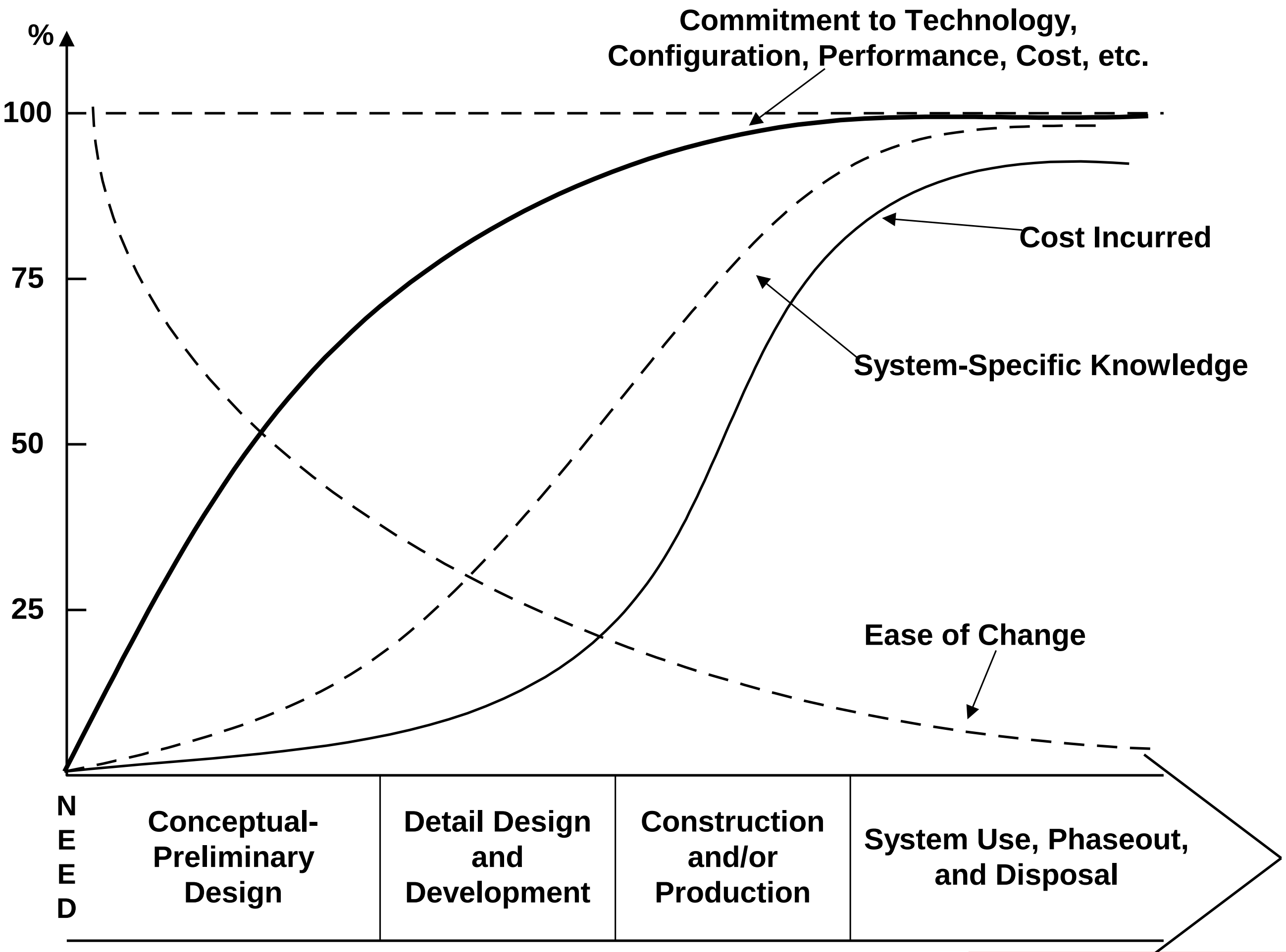
# Risk Management Strategies - II

- 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

# When You have to Start RM?

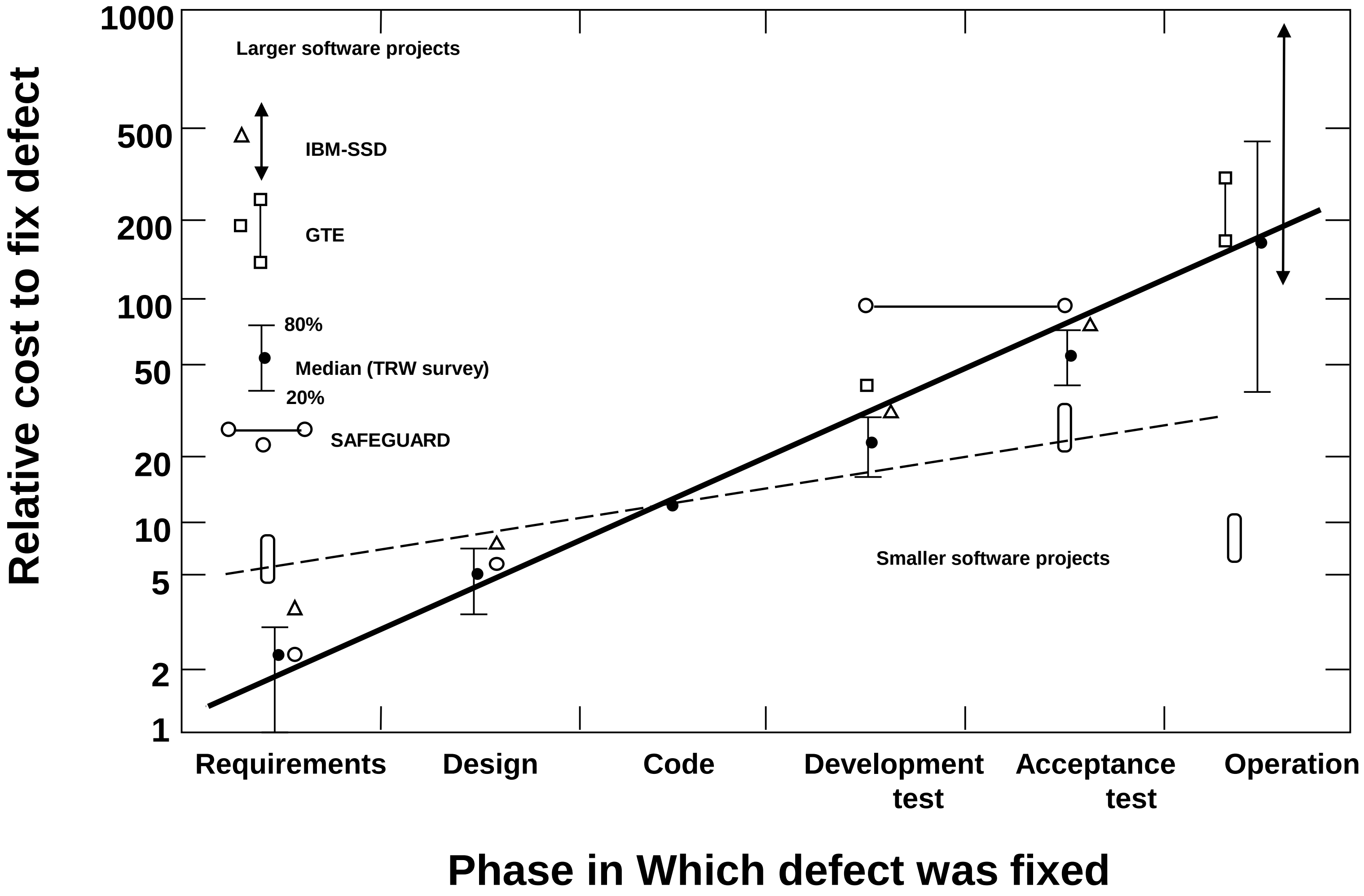
- The Answer is DAY ONE
  - Early & Risk Resolution
    - Data
    - Temptations to Avoid
- Early Risk Resolution with the WinWin Spiral Model
  - Identifying Stakeholders and Win Conditions
  - Model Clash and Win-Lose Risk Avoidance
  - Avoiding Cost/Schedule Risks with the SAIV Model

# Risk of Delaying Risk Management: Systems

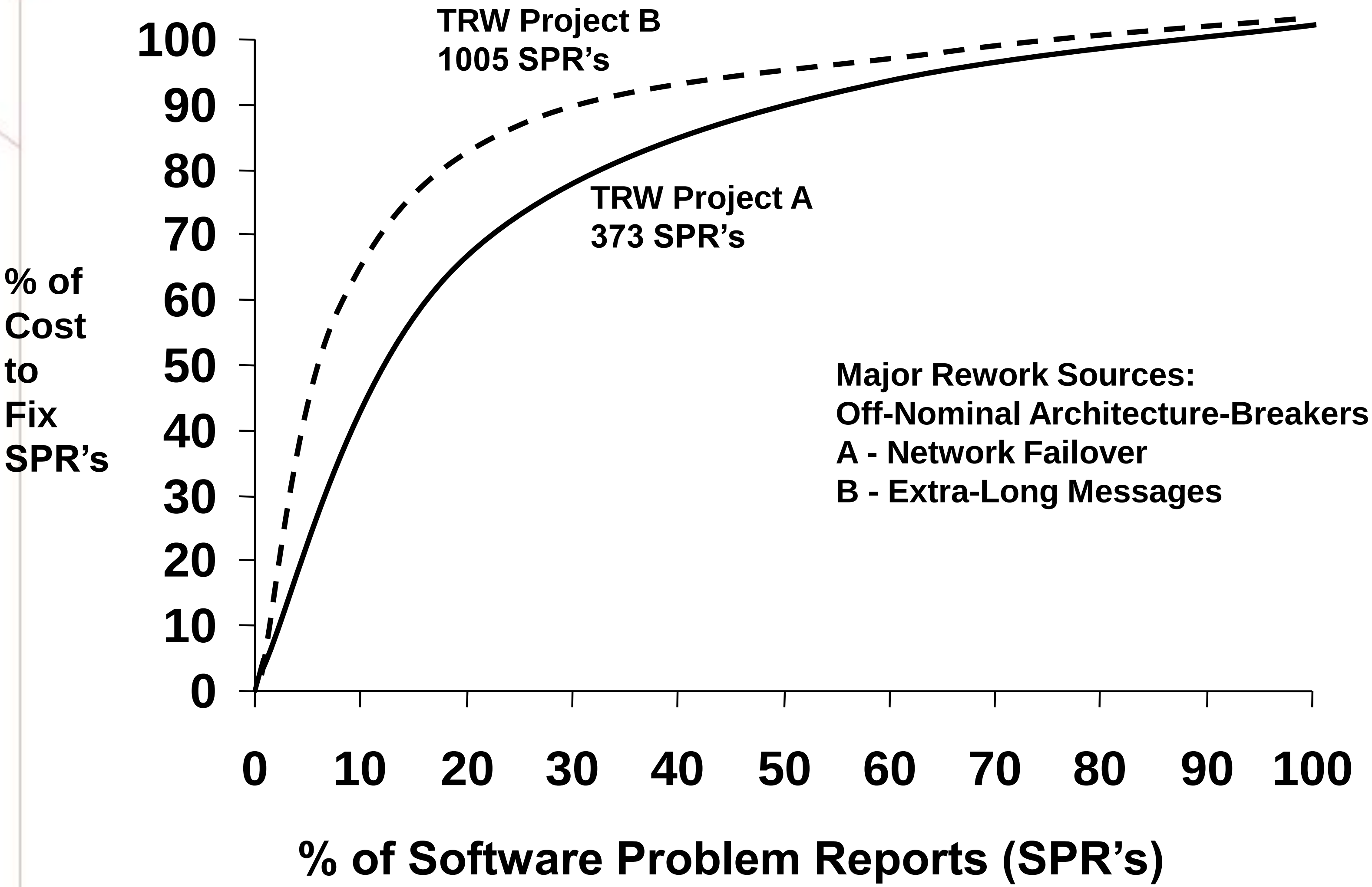


Blanchard– Fabrycky, 1998

# 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

# 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

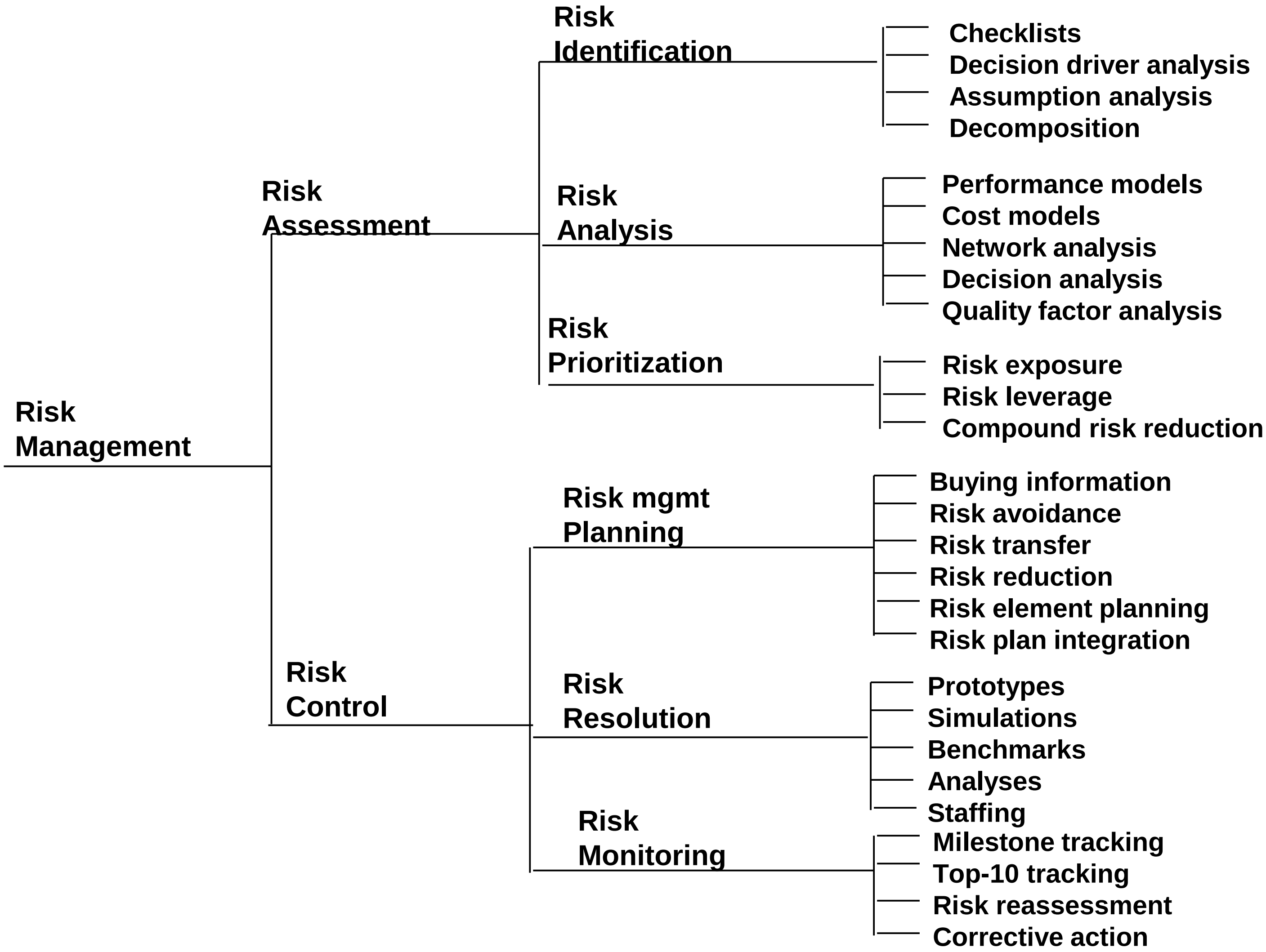
- We don't have time to think about the risks.  We need to:
  - Get some code running right away
  - Put on a socko demo for the customers
  - Be decisive.  Lead.  Make commitments.

- 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

# 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

WE'LL NEED A RISK ANALYSIS ON THIS PROJECT BEFORE I CAN APPROVE IT.
www.unitedmedia.com
S. Adams
RISK 1: INDECISIVENESS
RISK 2: OVERANALYSIS
RISK 3: CLUELESSNESS
RISK 4: MICROMANAGEMENT...
CLICK
CLICK
CLICK
11/8/97
© 1997 United Feature Syndicate, Inc.
I DON'T UNDERSTAND THESE RISKS.
THAT'S NUMBER THIRTY-SIX.

# Risk Identification

- Produce lists of the project specific risk items likely to compromise a project's success

- Typical risk Identification Techniques
  - Risk-item checklists
  - Decision driver analysis
  - Comparison with experience (Assumption Analysis)
  - Win-lose, lose-lose situations
  - Decomposition
    - Pareto 80 – 20 phenomena
    - Task dependencies
    - Murphy's law
    - Uncertainty areas
  - Analysis of Model Clashes

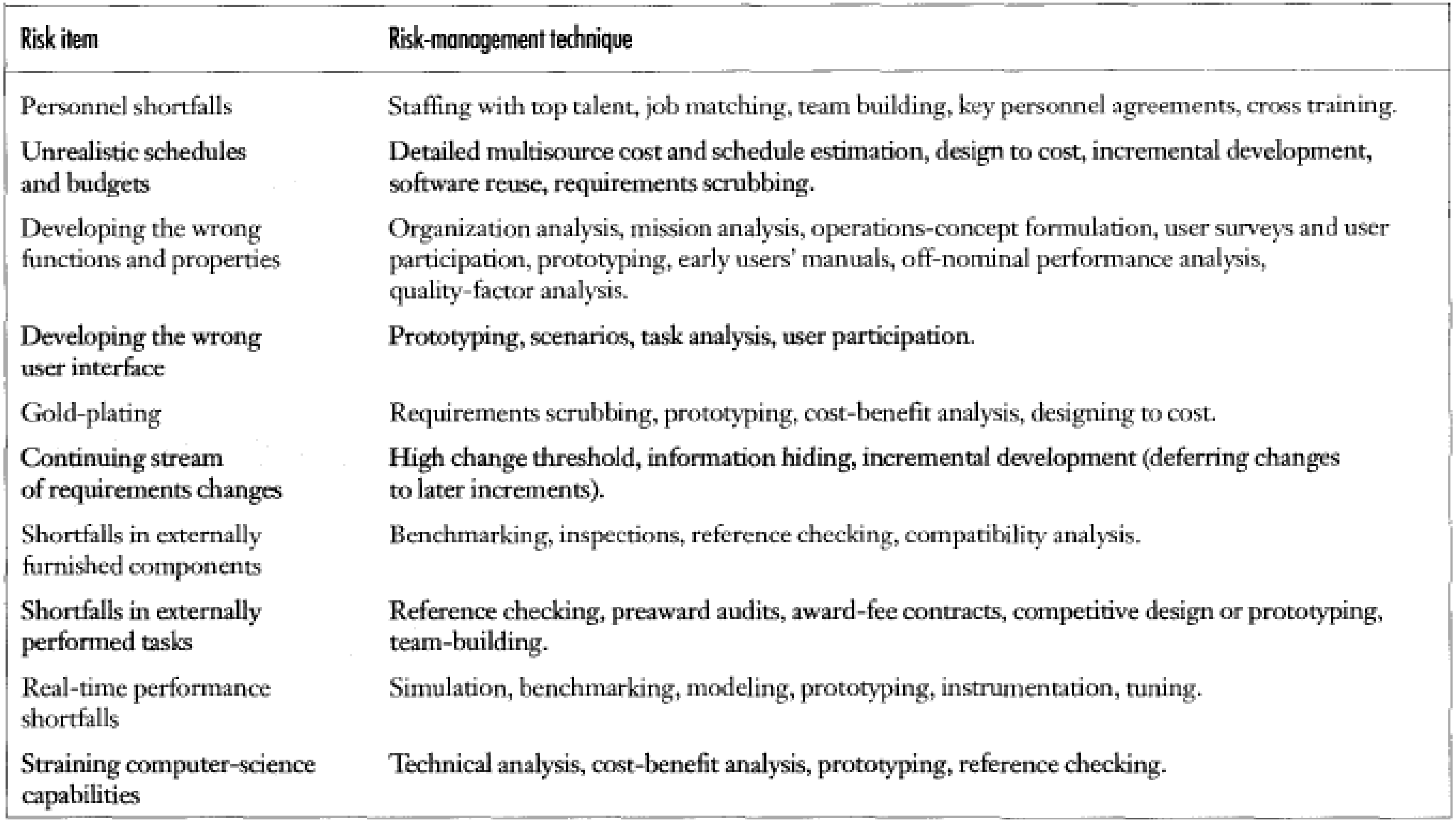
# Example: Risk Item

- 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?

# Example: Examination of 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

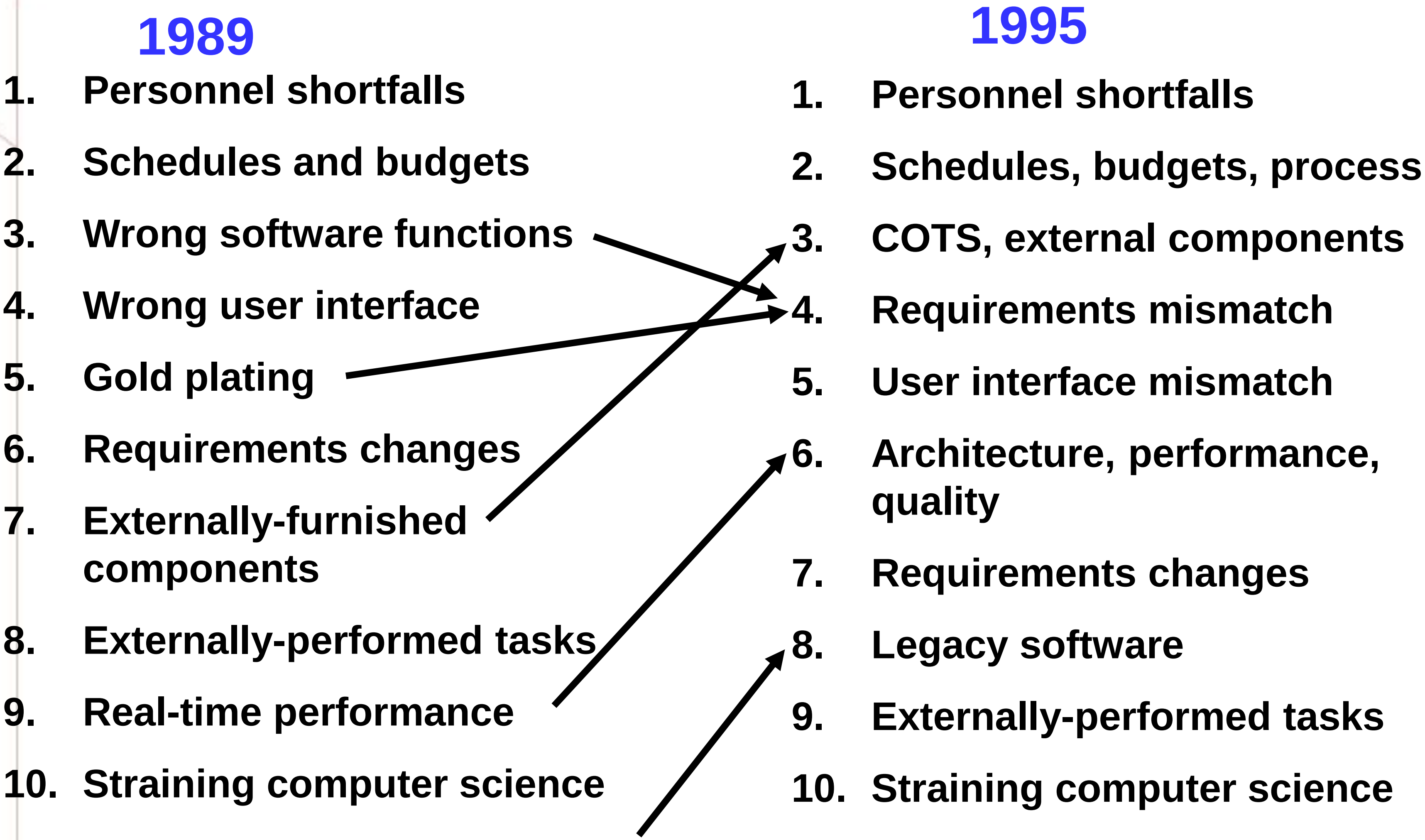
# Top 10 S/W Risk Items



| Risk item | Risk-management technique |
|---|---|
| Personnel shortfalls | Staffing with top talent, job matching, team building, key personnel agreements, cross training. |
| Unrealistic schedules and budgets | Detailed multisource cost and schedule estimation, design to cost, incremental development, software reuse, requirements scrubbing. |
| Developing the wrong functions and properties | Organization analysis, mission analysis, operations-concept formulation, user surveys and user participation, prototyping, early users' manuals, off-nominal performance analysis, quality-factor analysis. |
| Developing the wrong user interface | Prototyping, scenarios, task analysis, user participation. |
| Gold-plating | Requirements scrubbing, prototyping, cost-benefit analysis, designing to cost. |
| Continuing stream of requirements changes | High change threshold, information hiding, incremental development (deferring changes to later increments). |
| Shortfalls in externally furnished components | Benchmarking, inspections, reference checking, compatibility analysis. |
| Shortfalls in externally performed tasks | Reference checking, preaward audits, award-fee contracts, competitive design or prototyping, team-building. |
| Real-time performance shortfalls | Simulation, benchmarking, modeling, prototyping, instrumentation, tuning. |
| Straining computer-science capabilities | Technical analysis, cost-benefit analysis, prototyping, reference checking. |

Barry Boehm, IEEE Software January, 1991

# Top 10 Risk Items: 1989 vs. 1995



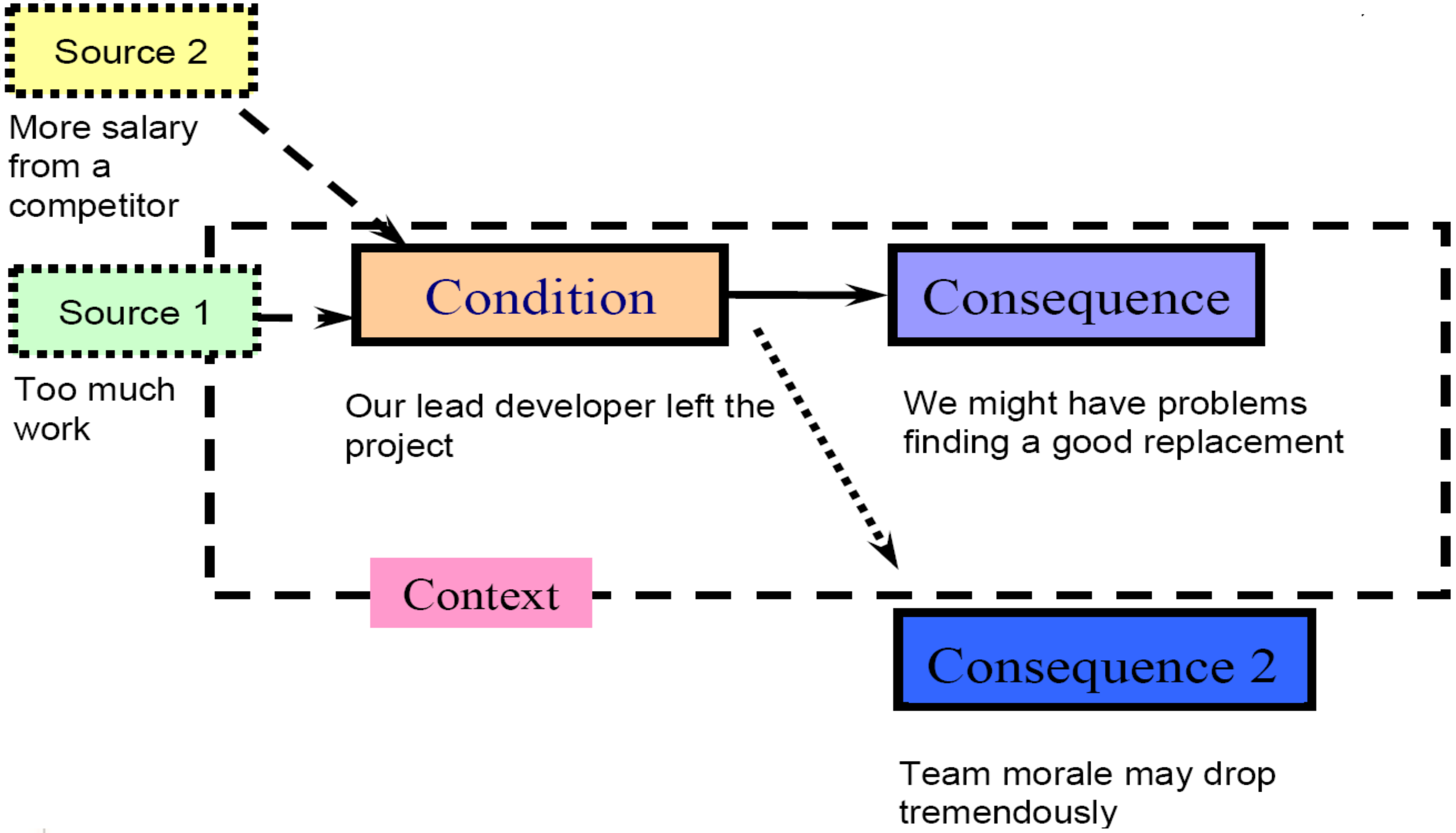
**1989**

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

# Risk Context & Source(s)



Source 2

More salary from a competitor

Source 1

Too much work

Condition

Our lead developer left the project

Consequence

We might have problems finding a good replacement

Context

Consequence 2

Team morale may drop tremendously

# Q & A