CMMI and the Balance of Discipline and Agility

Barry Boehm, USC

CMMI Technology Conference '02

November 13, 2002

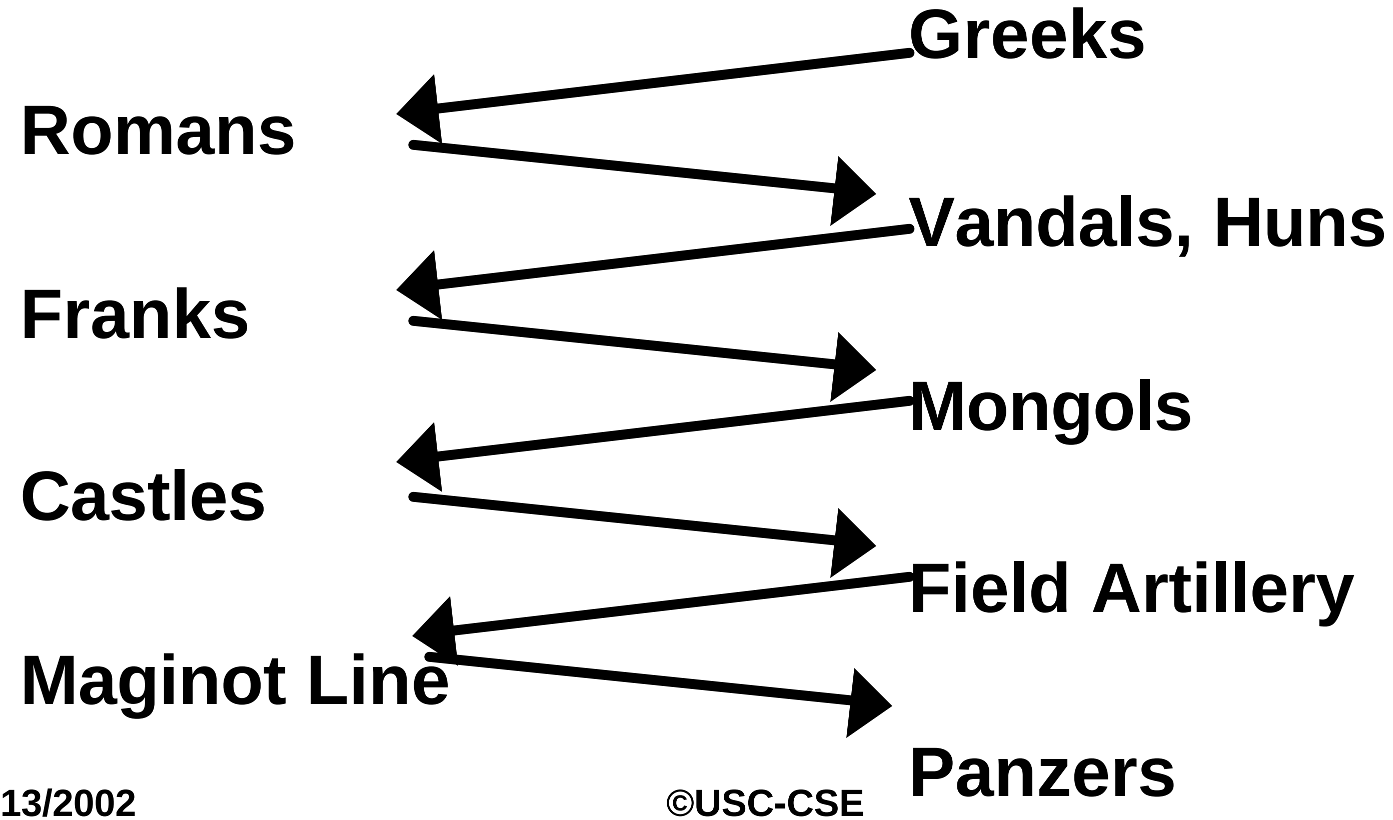
(boehm@; http://) sunset.usc.edu

Outline

- Clausewitz and De Marco: Armor vs. Mobility
  - Software CMM and Agile Methods
- Characteristics of Future Systems
  - Range of sizes and criticalities
  - All need to balance discipline and agility
- Using Risk to Balance Discipline and Agility
  - No one-size-fits-all solution
- Representative Future Example: Future Combat Systems
  - Complex system of systems (CSOS)
- Conclusions

Clausewitz and De Marco: Armor and Mobility

- Clausewitz: Armor and mobility alternate dominance



Greeks

Romans

Vandals, Huns

Franks

Mongols

Castles

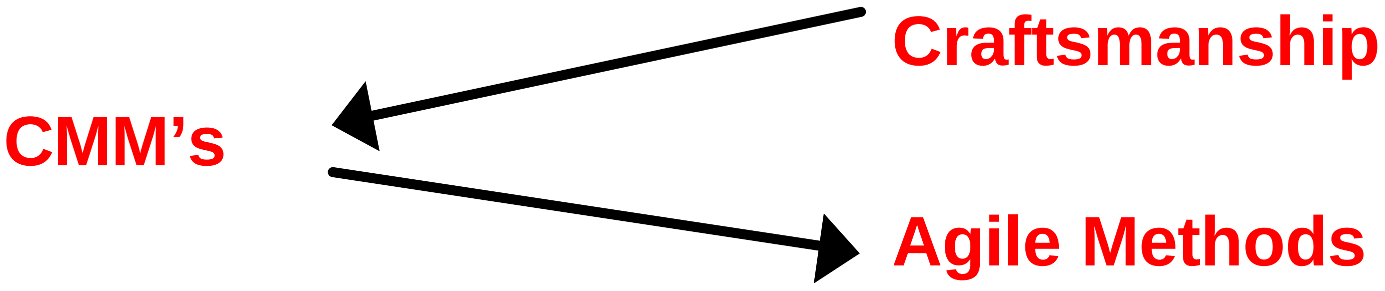
Field Artillery

Maginot Line

Panzers

Clausewitz and De Marco: Armor and Mobility

- Clausewitz: Armor and mobility alternate dominance
- De Marco: Same is true for software methods



Craftsmanship

CMM's

Agile Methods

- Whither CMMI?

The Agile Manifesto - I

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Various Agile Methods Available

- Adaptive Software Development (ASD)
- Agile Modeling
- Crystal methods
- Dynamic System Development Methodology (DSDM)

* eXtreme Programming (XP)

- Feature Driven Development
- Lean Development
- Scrum

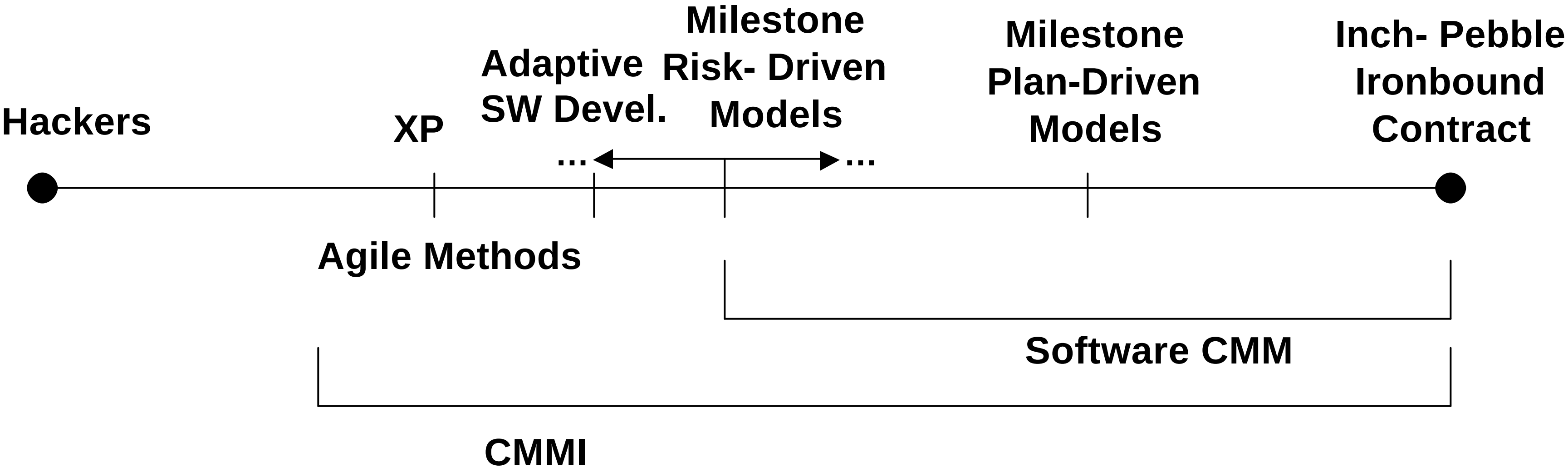
XP: The 12 Practices

- The Planning Game
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring

- Pair Programming
- Collective Ownership
- Continuous Integration
- 40-hour Week
- On-site Customer
- Coding Standards

-Used generatively, not imperatively

The Planning Spectrum



Hackers

XP

Adaptive
SW Devel.

Milestone
Risk- Driven
Models

Milestone
Plan-Driven
Models

Inch- Pebble
Ironbound
Contract

… …

Agile Methods

Software CMM

CMMI

Agile and Plan-Driven Home Grounds

Agile Home Ground	Plan-Driven Home Ground
• Agile, knowledgeable, collocated, collaborative developers	• Plan-oriented developers; mix of skills
• Above plus representative, empowered customers	• Mix of customer capability levels
• Reliance on tacit interpersonal knowledge	• Reliance on explicit documented knowledge
• Largely emergent requirements, rapid change	• Requirements knowable early; largely stable
• Architected for current requirements	• Architected for current and foreseeable requirements
• Refactoring inexpensive	• Refactoring expensive
• Smaller teams, products	• Larger teams, products
• Premium on rapid value	• Premium on high-assurance

Outline

- Clausewitz and De Marco: Armor vs. Mobility
  - Software CMM and Agile Methods
- ➔ Characteristics of Future Systems
  - Range of sizes and criticalities
  - All need to balance discipline and agility
- Using Risk to Balance Discipline and Agility
  - No one-size-fits-all solution
- Representative Future Example: Future Combat Systems
  - Complex system of systems (CSOS)
- Conclusions

Information Technology Trends

Traditional Development	Current/Future Trends
Standalone systems	Everything connected-maybe
Stable requirements	Rapid requirements change
Rqts. determine capabilities	COTS capabilities determine rqts.
Control over evolution	No control over COTS evolution
Enough time to keep stable	Ever-decreasing cycle times
Small to big systems	Plus very big systems of systems
Repeatability-oriented process, maturity models	Adaptive process models

The "Separation of Concerns" Legacy

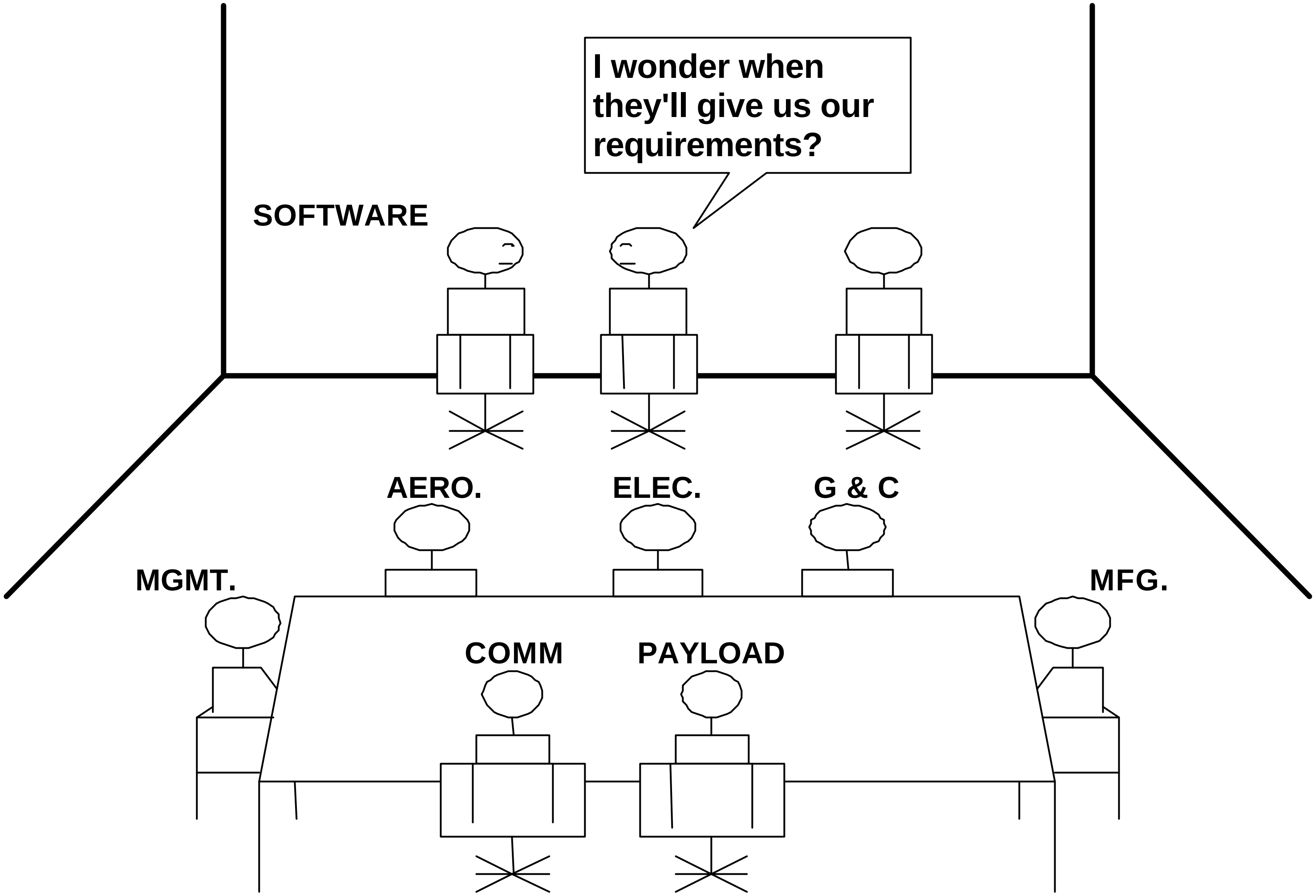
"The notion of 'user' cannot be precisely defined, and therefore has no place in CS or SE."

– Edsger Dijkstra, ICSE 4, 1979

"Analysis and allocation of the system requirements is not the responsibility of the SE group but is a prerequisite for their work."

– Mark Paulk at al., SEI Software CMM v.1.1, 1993

Resulting Project Social Structure



The CMMI Software Paradigm

- System and software engineering are integrated
  - Software has a seat at the center table
- Requirements, architecture, and process are developed concurrently
  - Along with prototypes and key capabilities
- Developments done by integrated teams
  - Collaborative vs. adversarial process
  - Based on shared vision, negotiated stakeholder

How Much Planning Is Enough?

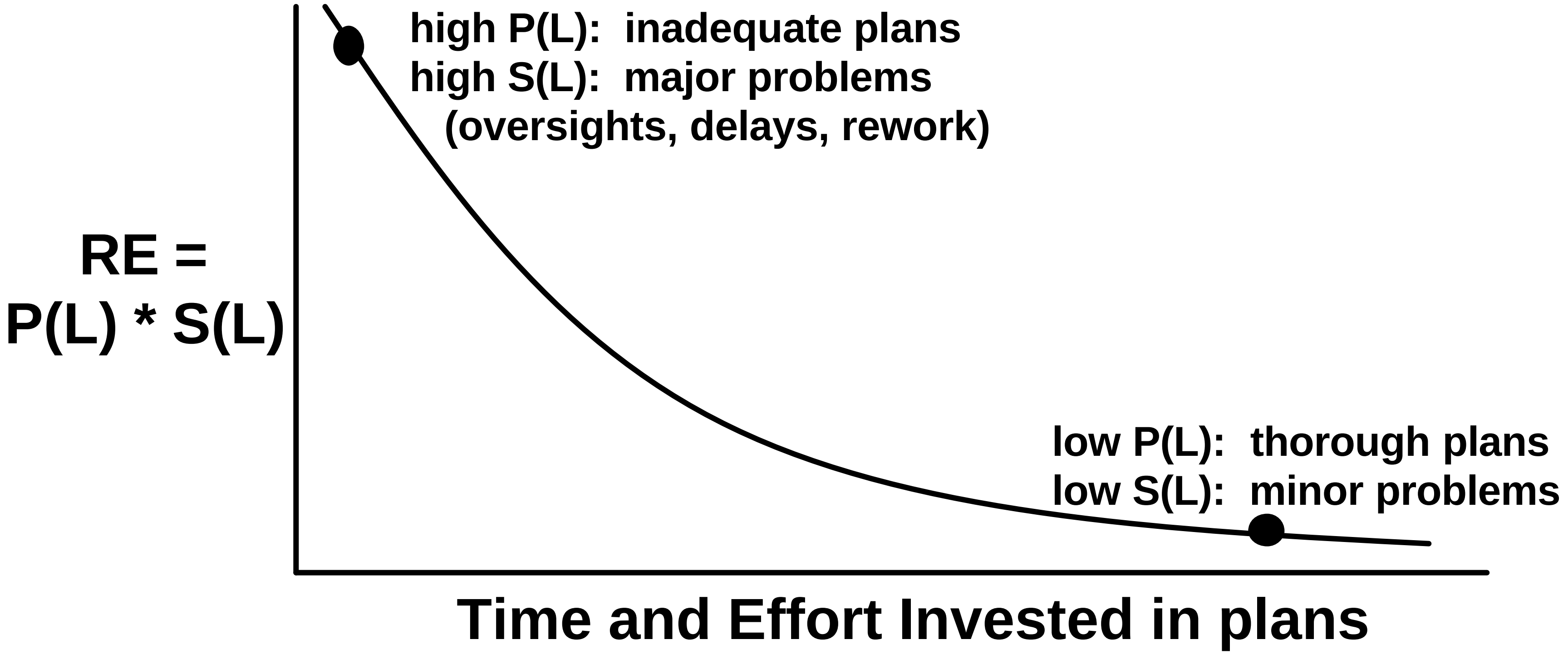
- A risk analysis approach

- Risk Exposure RE = Prob (Loss) * Size (Loss)
  - "Loss" – financial; reputation; future prospects, …
- For multiple sources of loss:

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

Example RE Profile: Planning Detail

- Loss due to inadequate plans



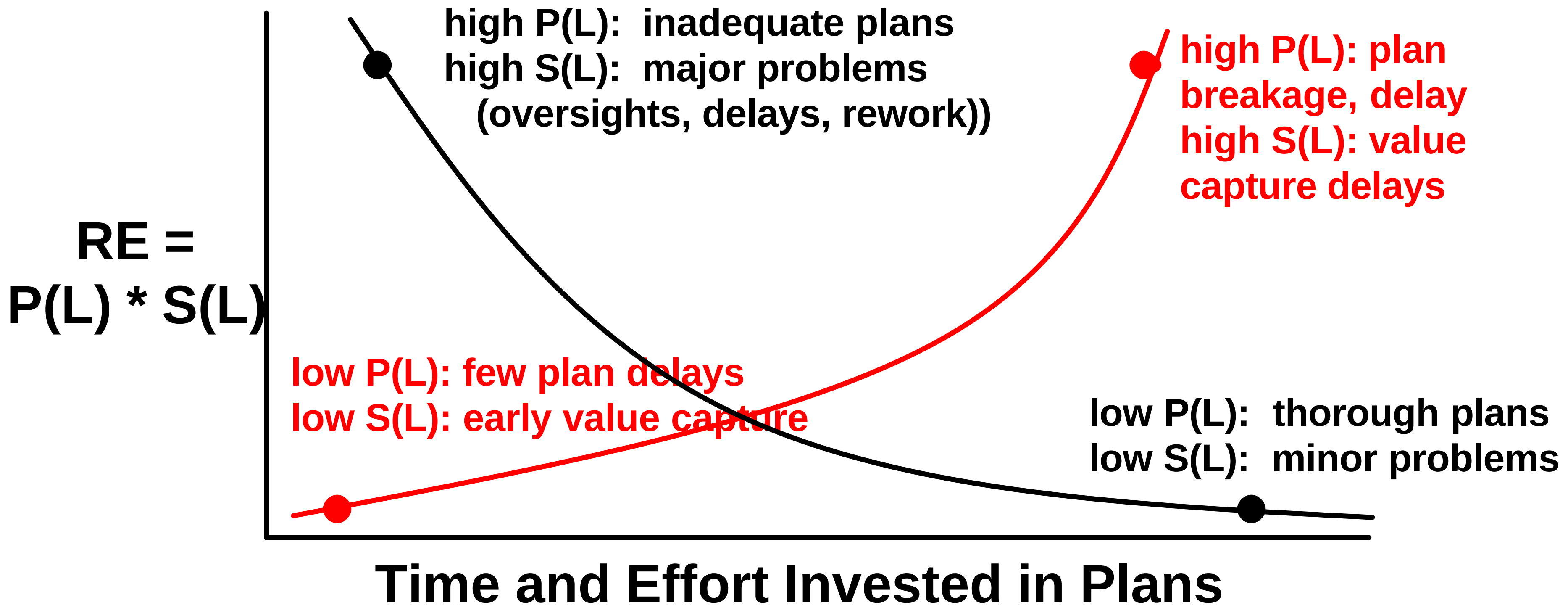
high P(L): inadequate plans
high S(L): major problems
(oversights, delays, rework)

low P(L): thorough plans
low S(L): minor problems

Example RE Profile: Planning Detail

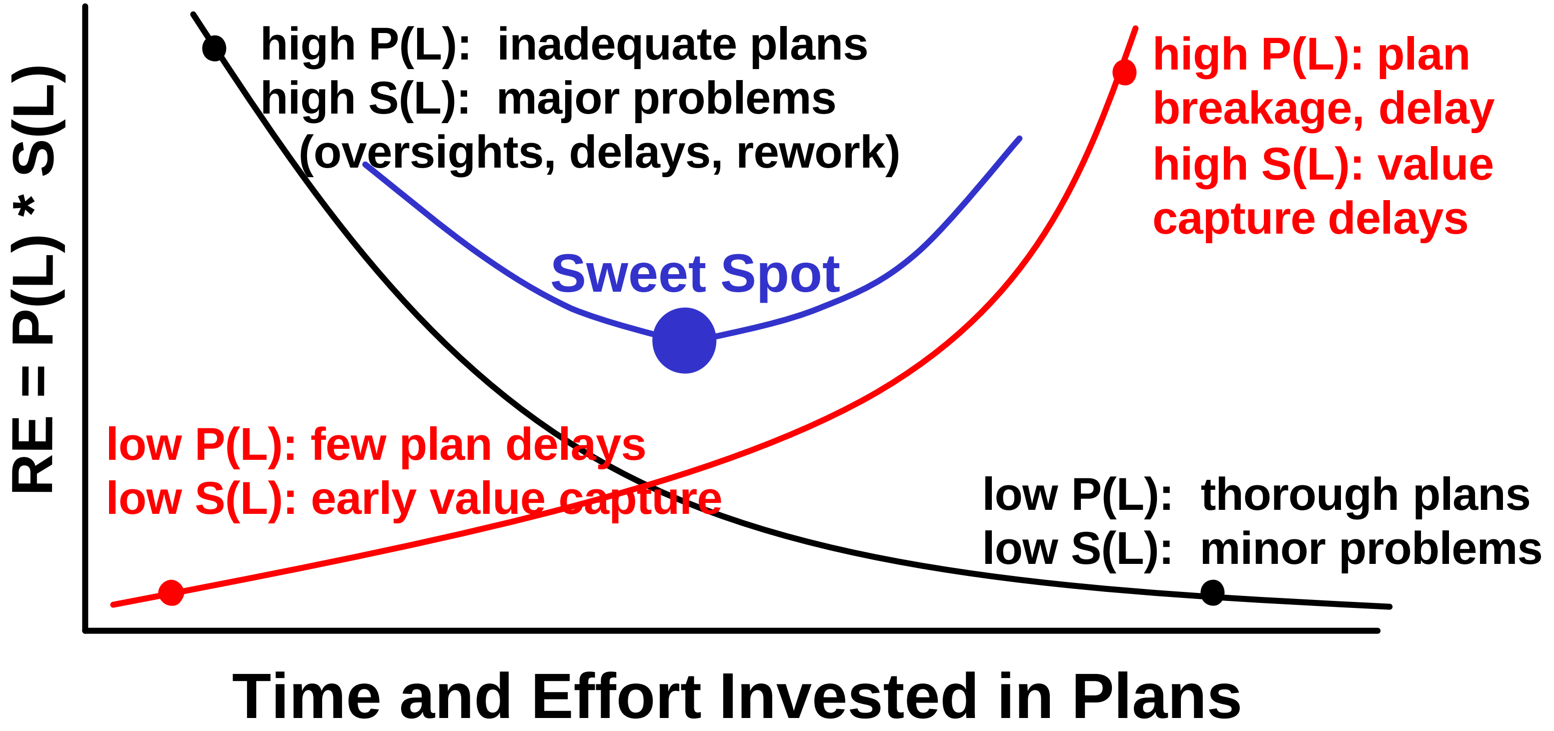
- Loss due to inadequate plans

- Loss due to market share erosion

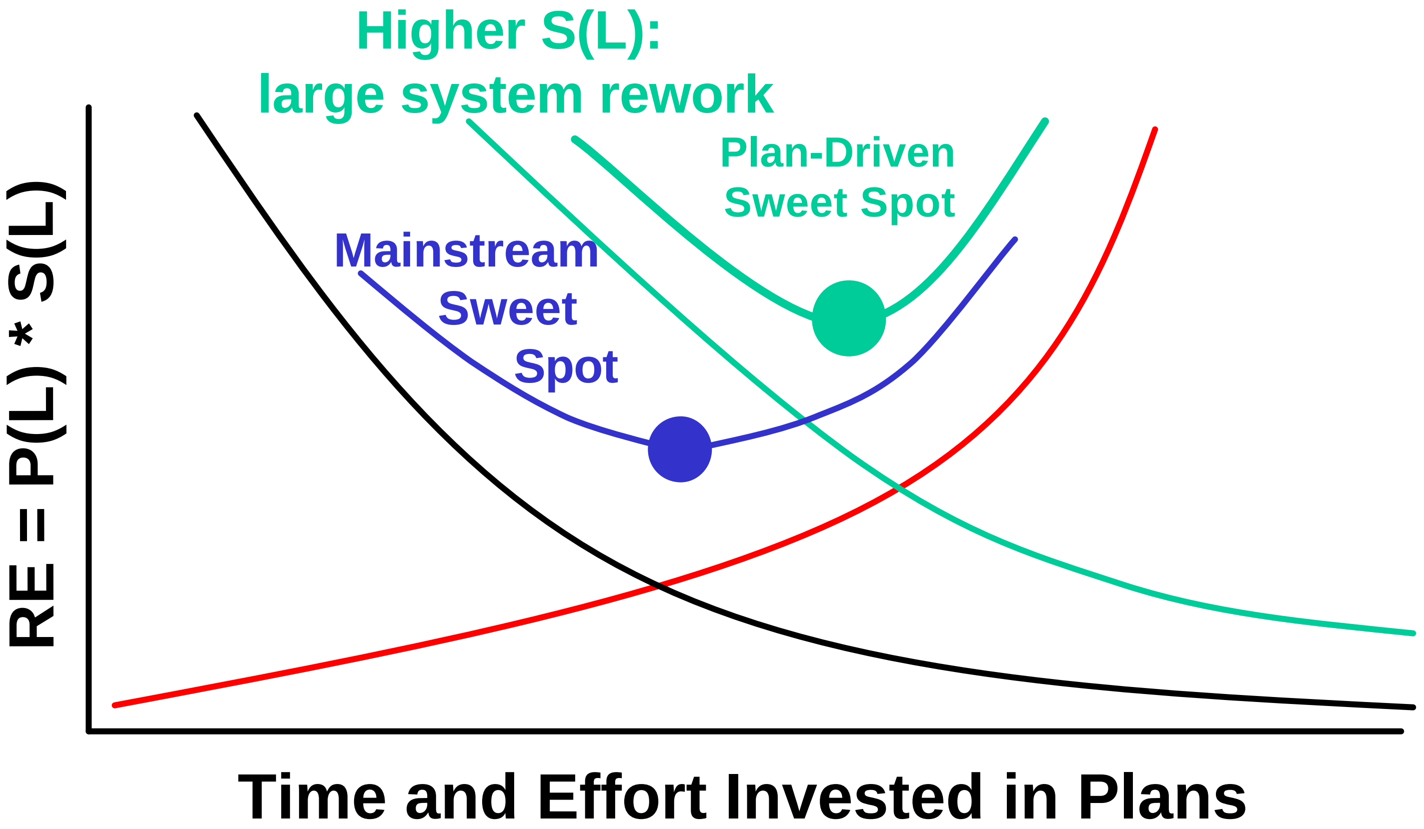


Example RE Profile: Time to Ship

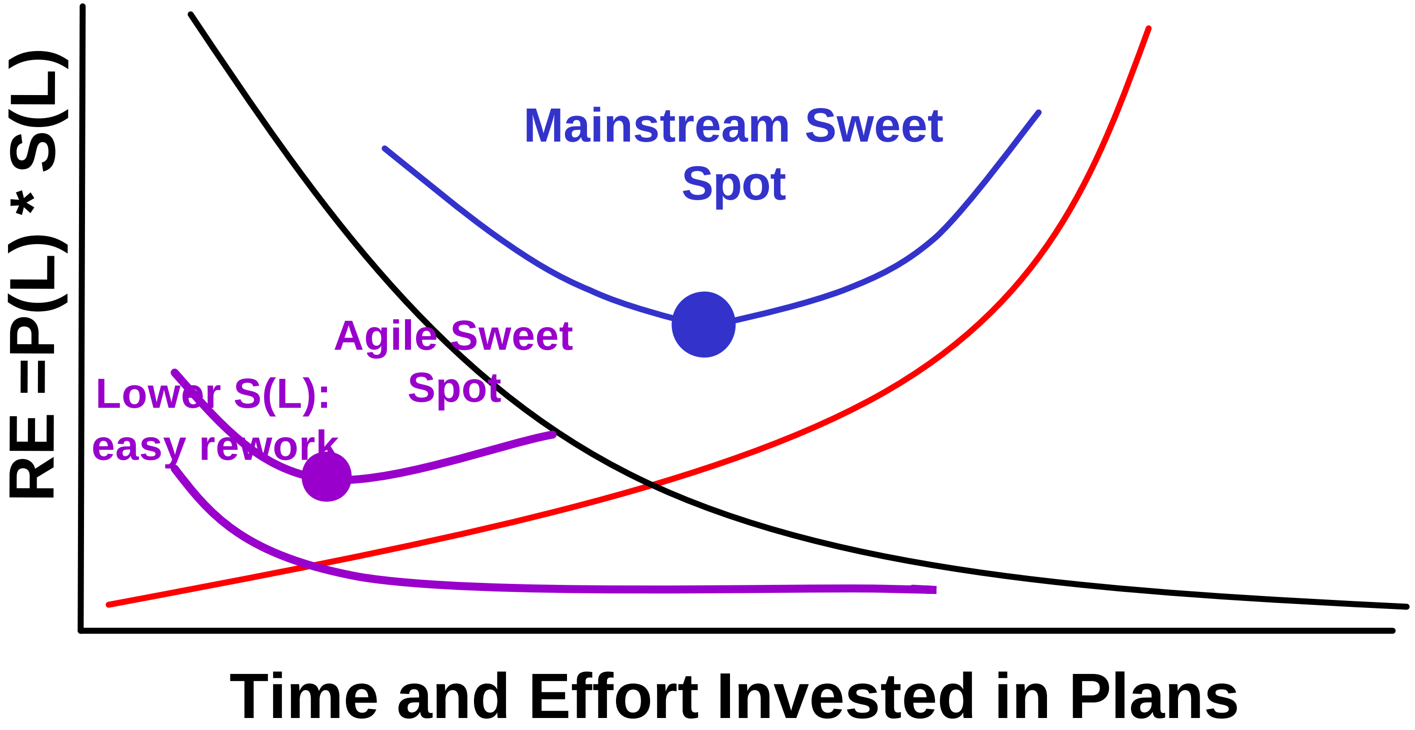
- Sum of Risk Exposures



Comparative RE Profile:
Plan-Driven Home Ground



Comparative RE Profile:
Agile Home Ground



Outline

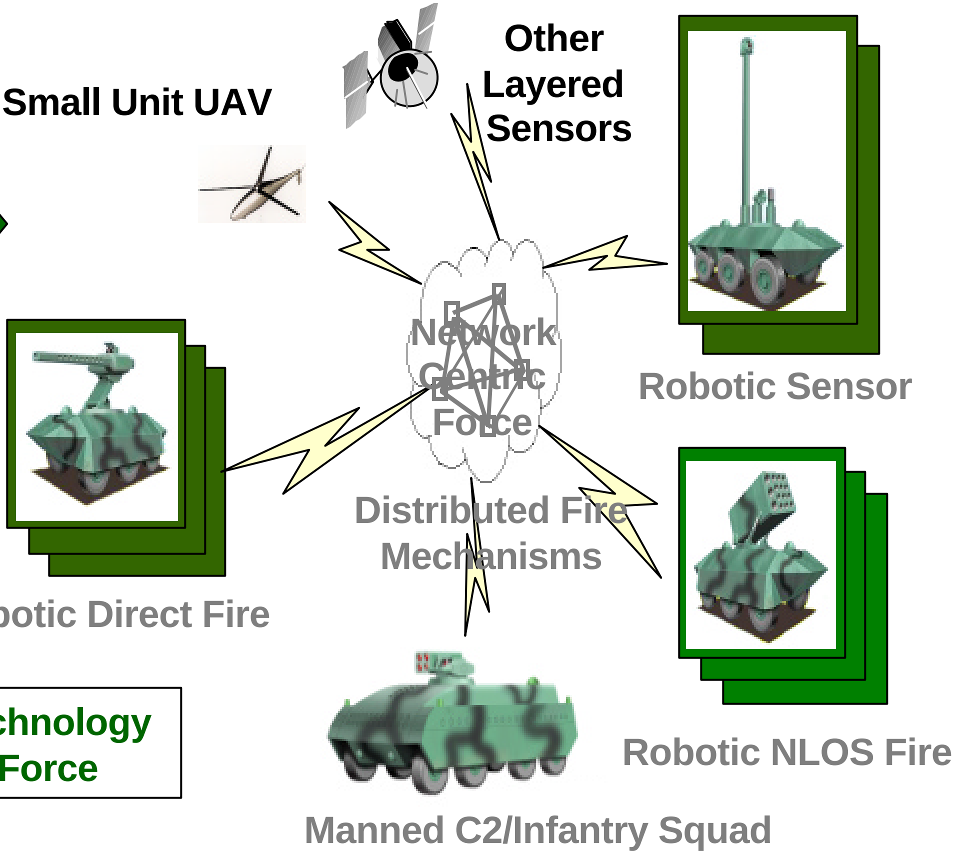
- Clausewitz and De Marco: Armor vs. Mobility
  - Software CMM and Agile Methods
- Characteristics of Future Systems
  - Range of sizes and criticalities
  - All need to balance discipline and agility
- Using Risk to Balance Discipline and Agility
  - No one-size-fits-all solution
- → Representative Future Example: Future Combat Systems
  - Complex system of systems (CSOS)
- Conclusions

Future Combat Systems:
A Network-Centric Example

From This...

To This...

Network Centric Distributed Platforms



Small Unit UAV

Other Layered Sensors

Network Centric Force

Robotic Sensor

Robotic Direct Fire

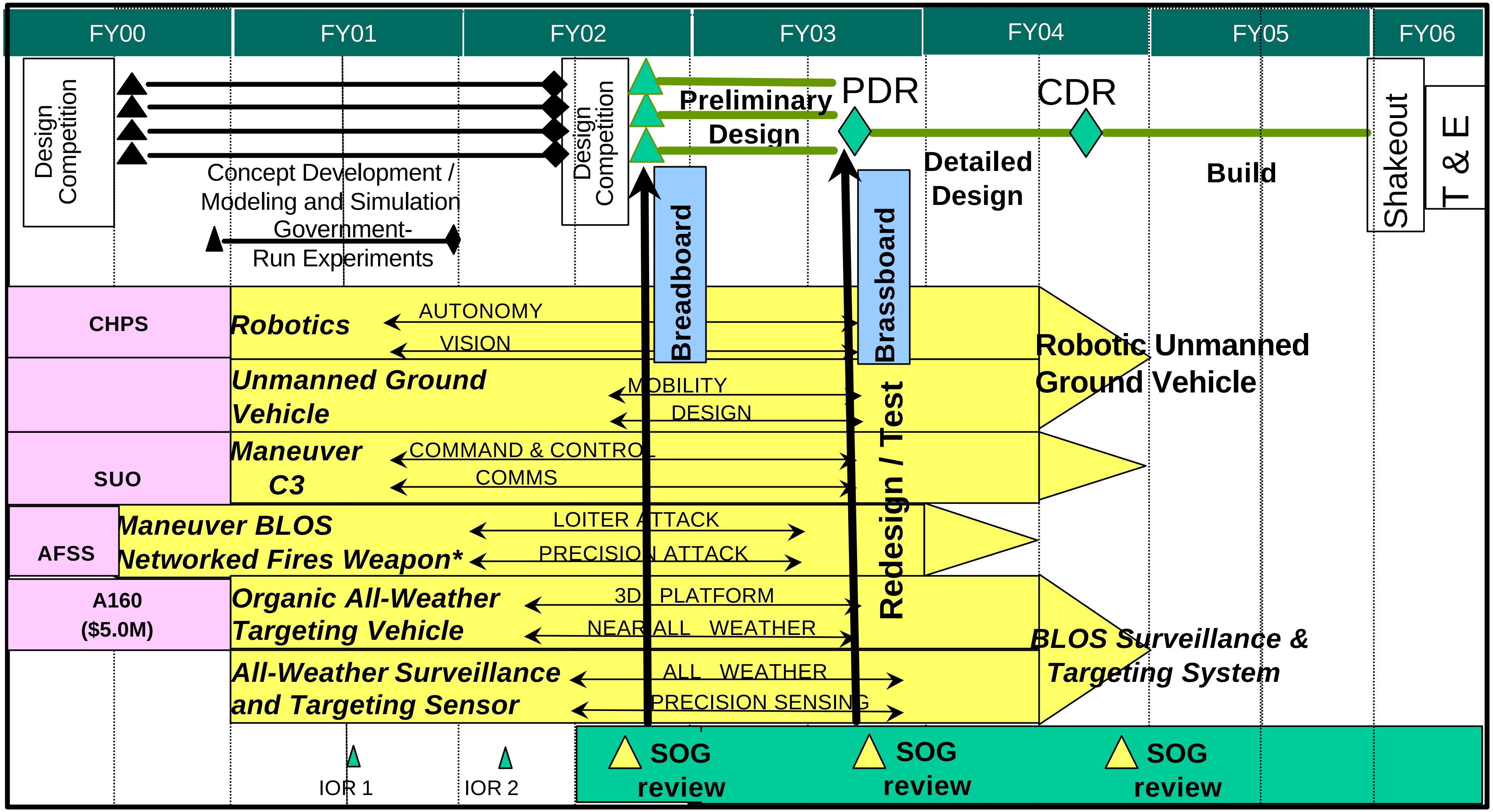
Distributed Fire Mechanisms

Robotic NLOS Fire

Manned C2/Infantry Squad

Exploit Battlefield Non-Linearities using Technology to Reduce the Size of Platforms and the Force

Total Collaborative Effort to Support FCS



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

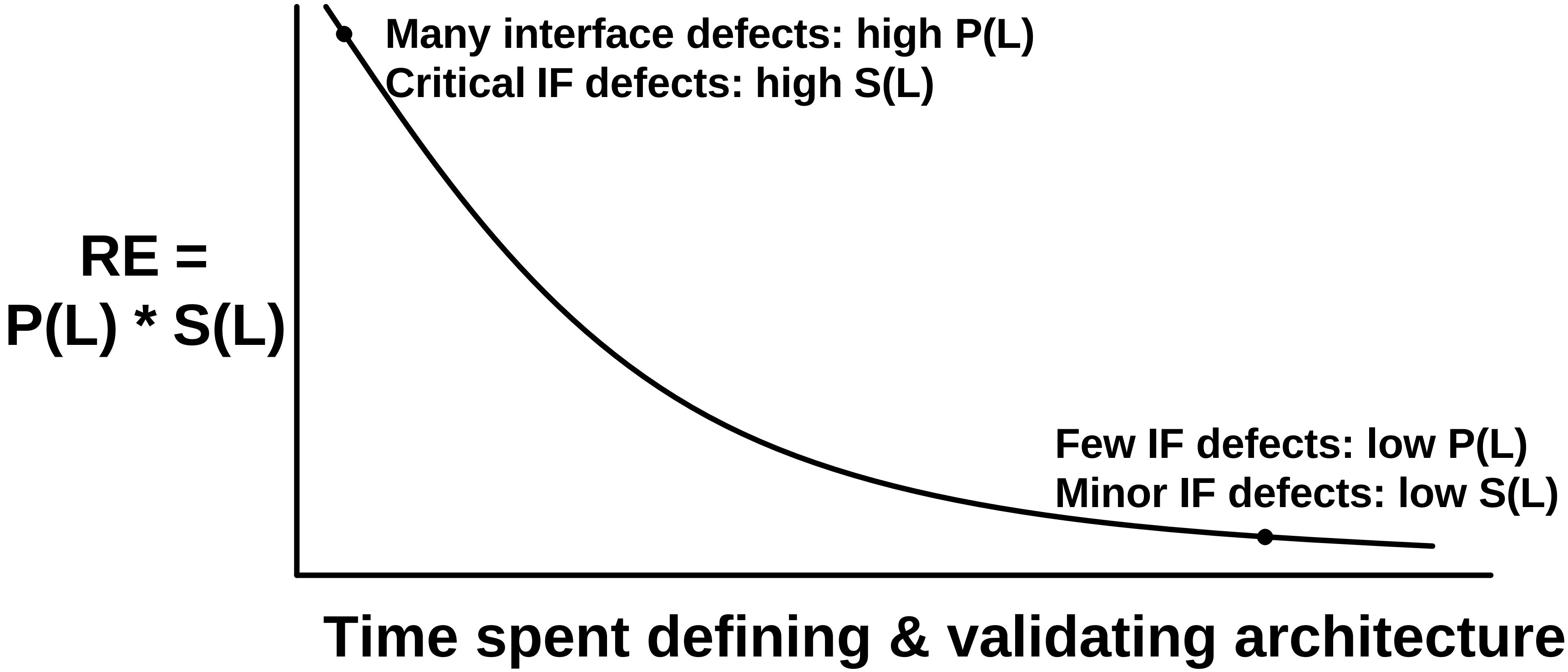
CSOS Software Benefits, Risks, and Strategies

- 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

How Soon to Define Subcontractor Interfaces?

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

-Loss due to rework delays



Many interface defects: high P(L)
Critical IF defects: high S(L)

RE =
P(L) * S(L)

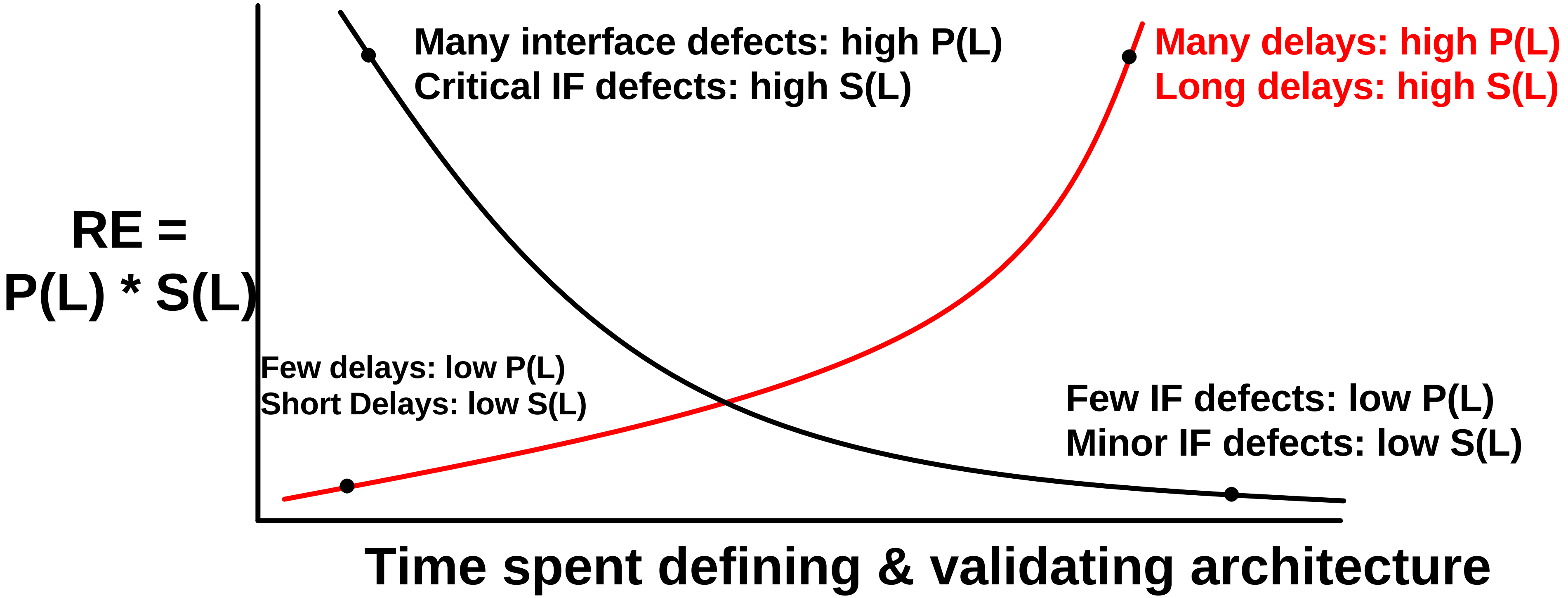
Few IF defects: low P(L)
Minor IF defects: low S(L)

Time spent defining & validating architecture

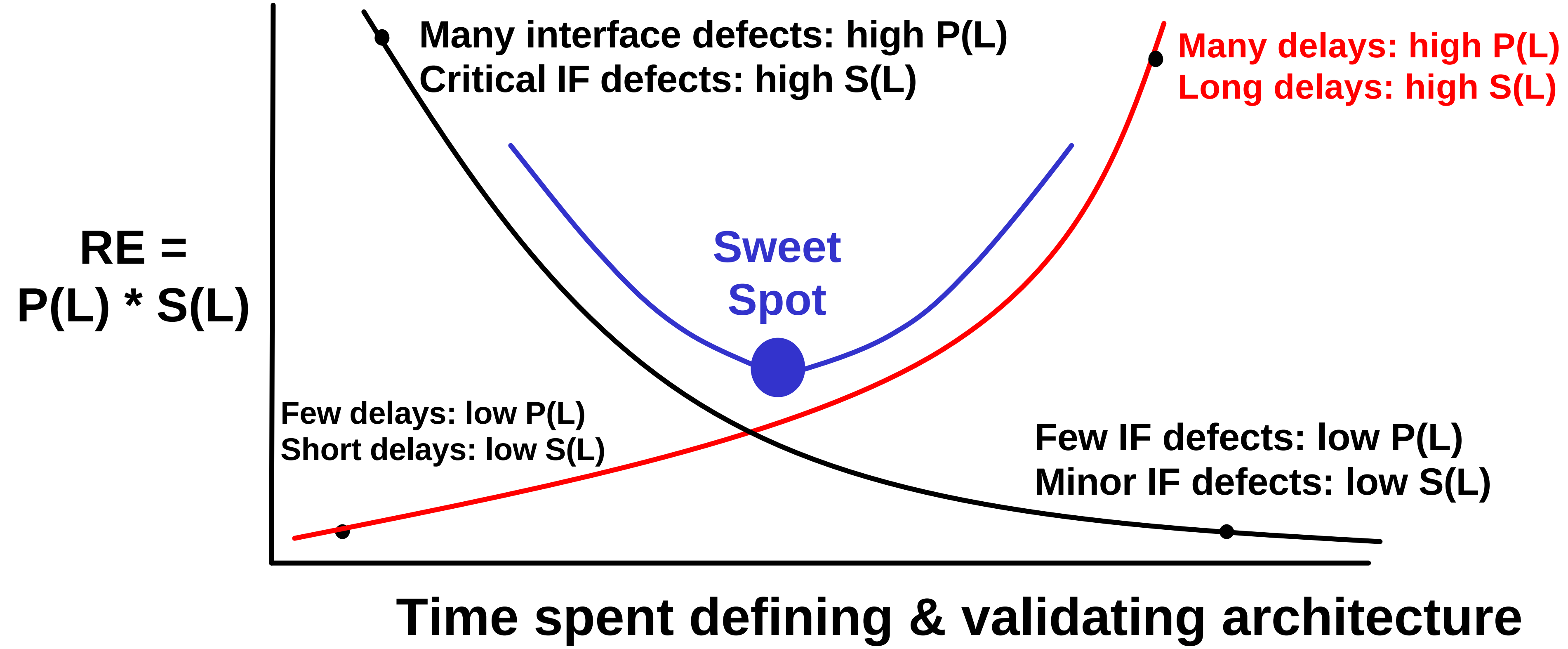
How Soon to Define Subcontractor Interfaces?

- Loss due to rework delays

- Loss due to late subcontact startups

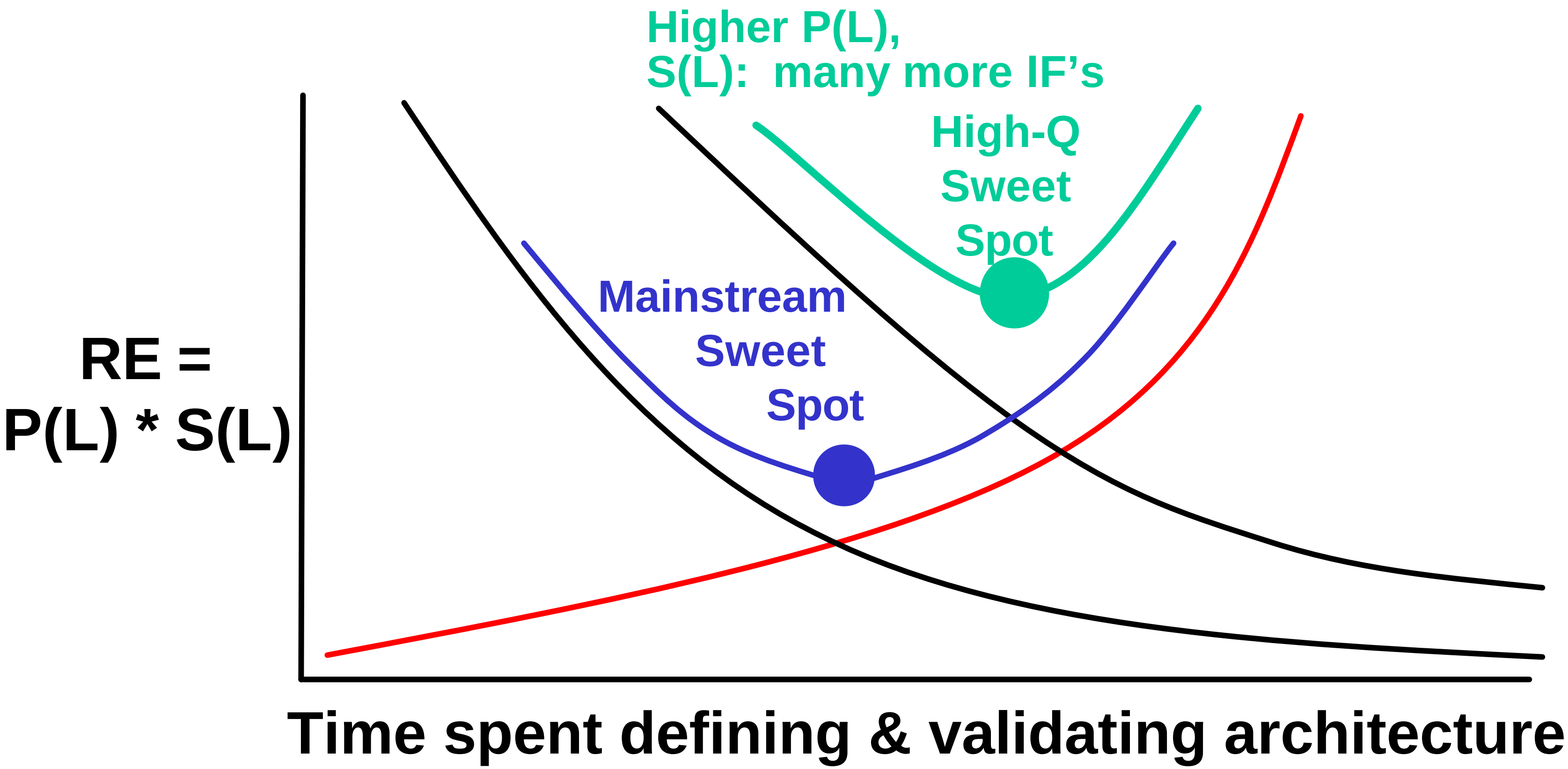


How Soon to Define Subcontractor Interfaces?
- Sum of Risk Exposures



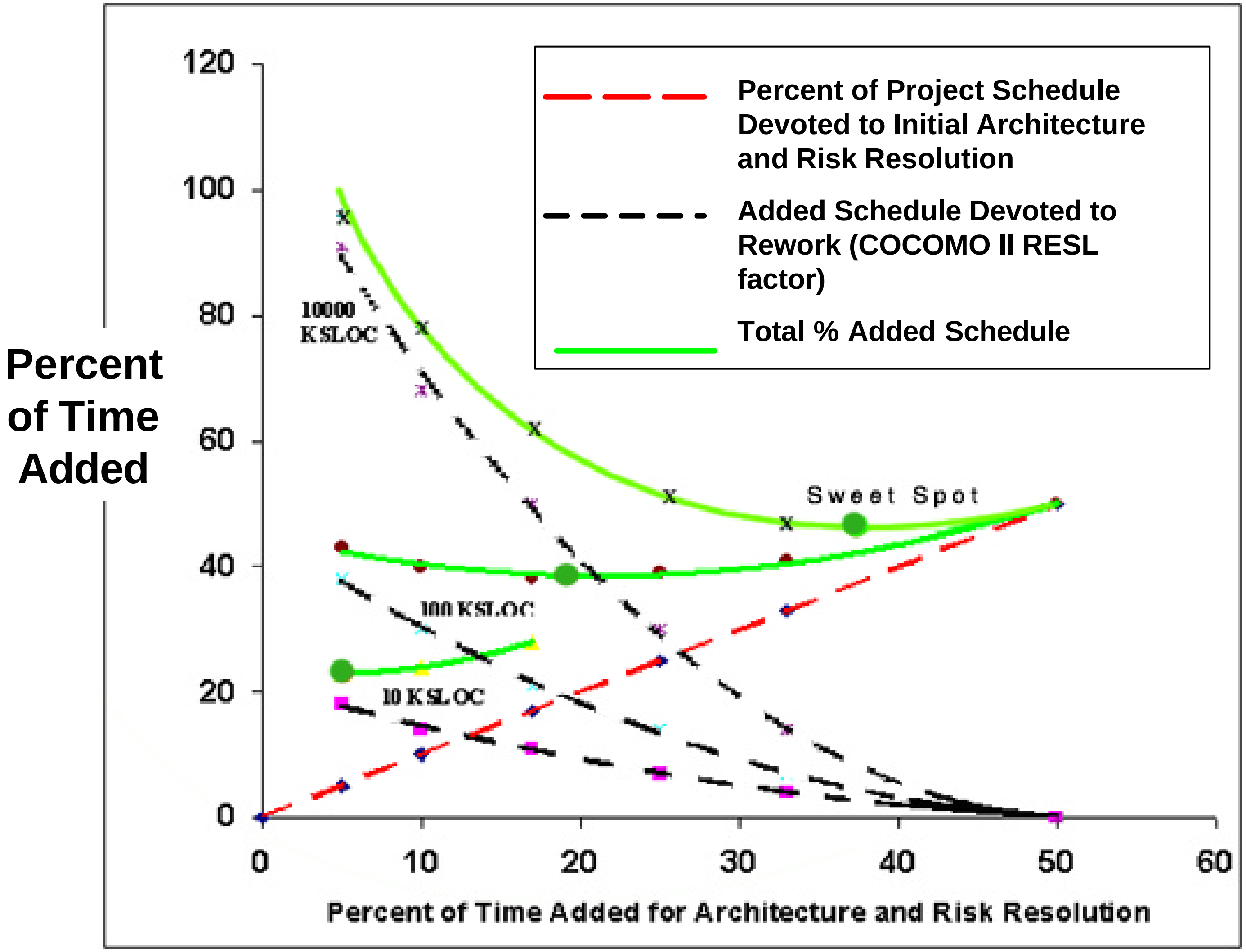
How Soon to Define Subcontractor Interfaces?

-Very Many Subcontractors



How Much Architecting is Enough: A COCOMO II Analysis

-KSLOC: Thousands of Source Lines of Code.



Conclusions

- Future systems need to balance discipline and agility
  - Need both high dependability and high adaptability
- Can interpret CMMI in two ways
  - Rigorous, heavyweight, one-size fits-all standard
  - Risk-driven combination of discipline and agility
- Need ability to accommodate agile methods
  - Liberal as well as literal compliance interpretations
  - Risk-driven content of processes and artifacts
  - Outreach to commercial sector