Center For Software Engineering

C e B A S E

Fraunhofer USA

Center for
Experimental Software Engineering
Maryland

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

Barry Boehm, USC

Vic Basili, Fraunhofer Maryland

SIS Acquisition Conference

January 28, 2003

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)

• 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 * ^{3} 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

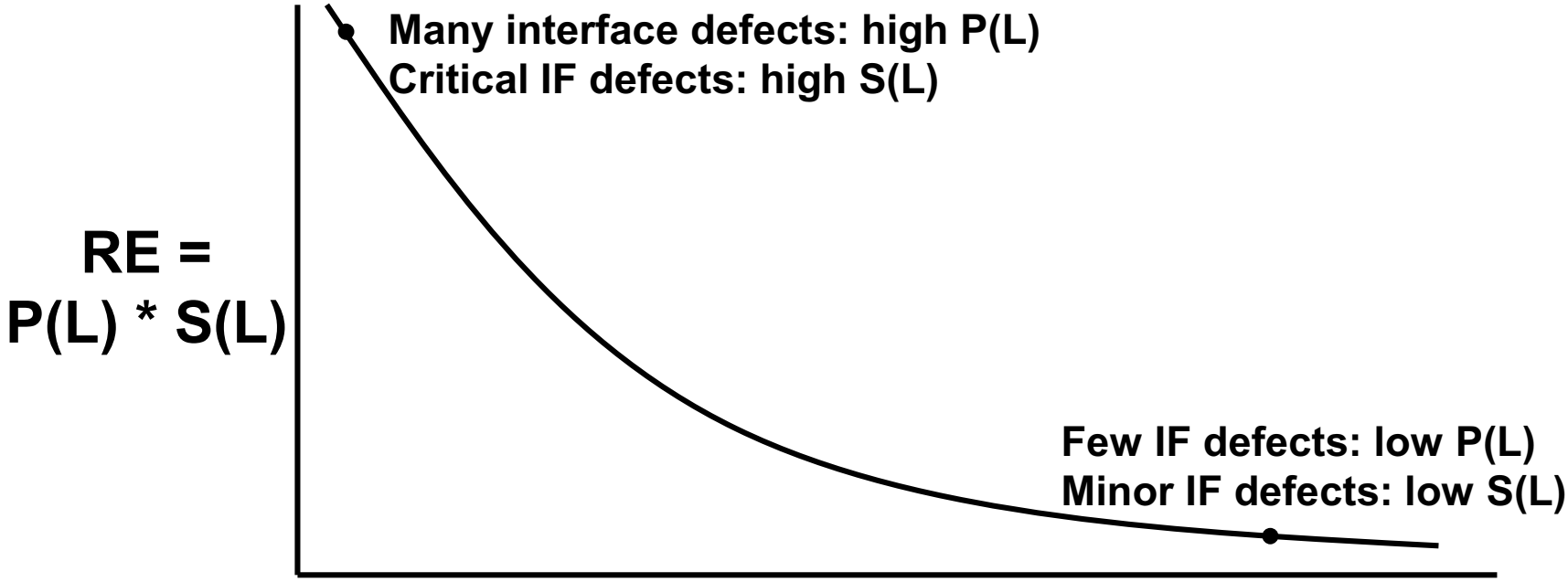
- **Risk #2. Critical dimensions of software integration may begin late and cause overruns. CSOS software needs to be integrated:**
  - **by CSC/CSCI hierarchy**
  - **by cross-IPT software exercises which incrementally ingrate the software**
  - **by critical threads and scenarios (nominal and off-nominal);**
  - **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.

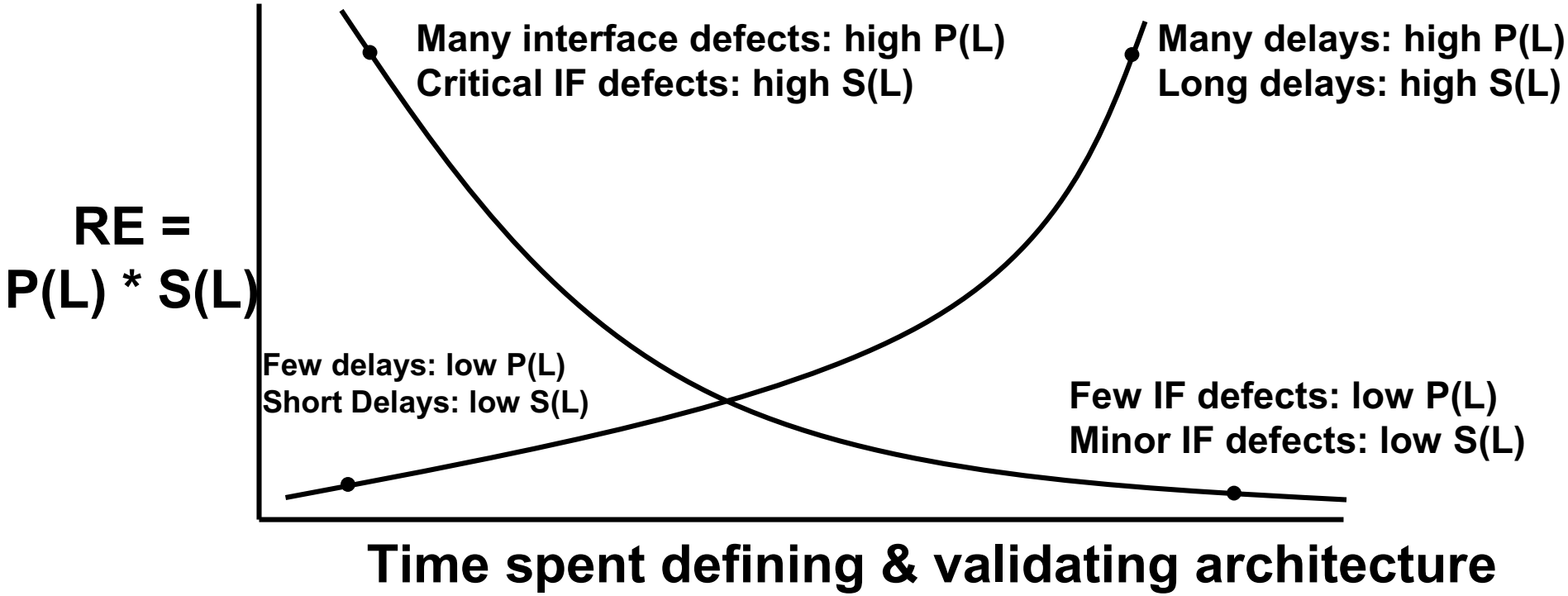
How Soon to Define Subcontractor Interfaces?

Risk exposure RE = Prob(Loss) * Size(Loss)
-Loss due to rework delays

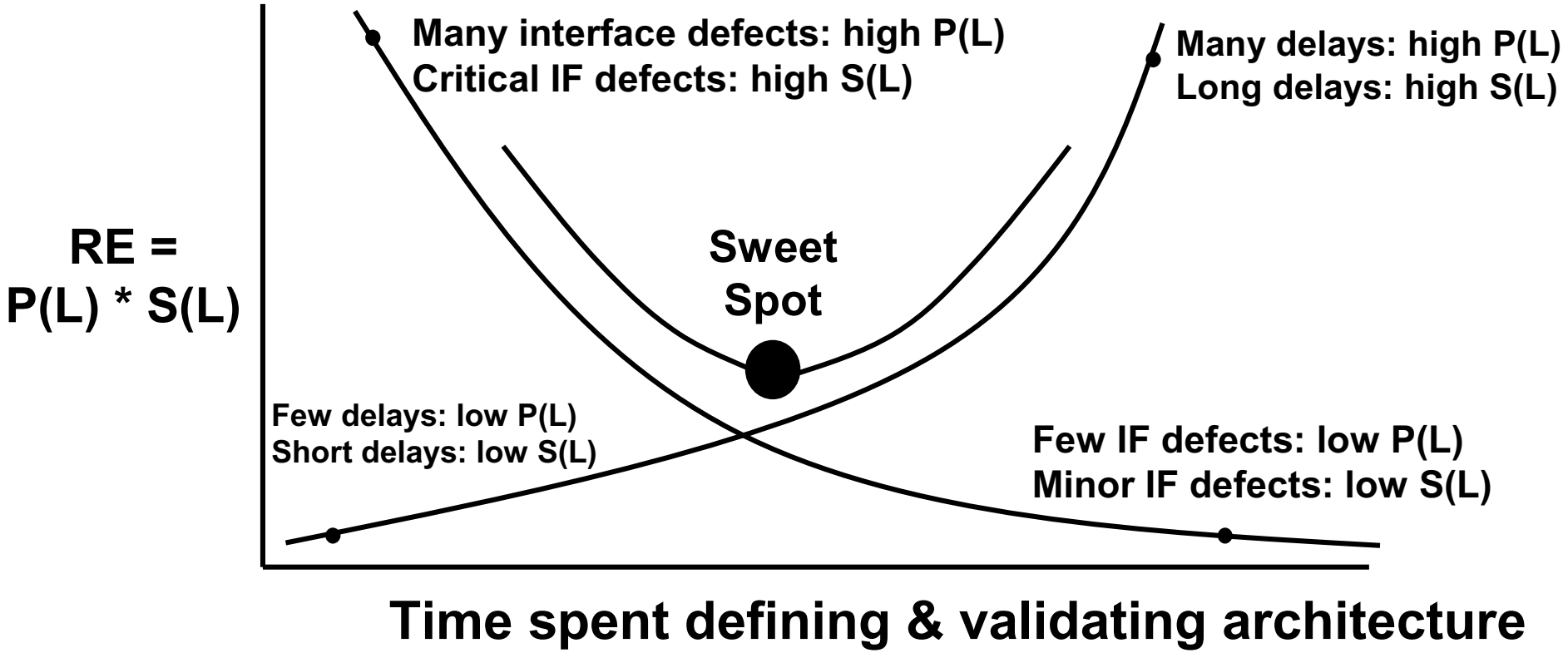


How Soon to Define Subcontractor Interfaces?

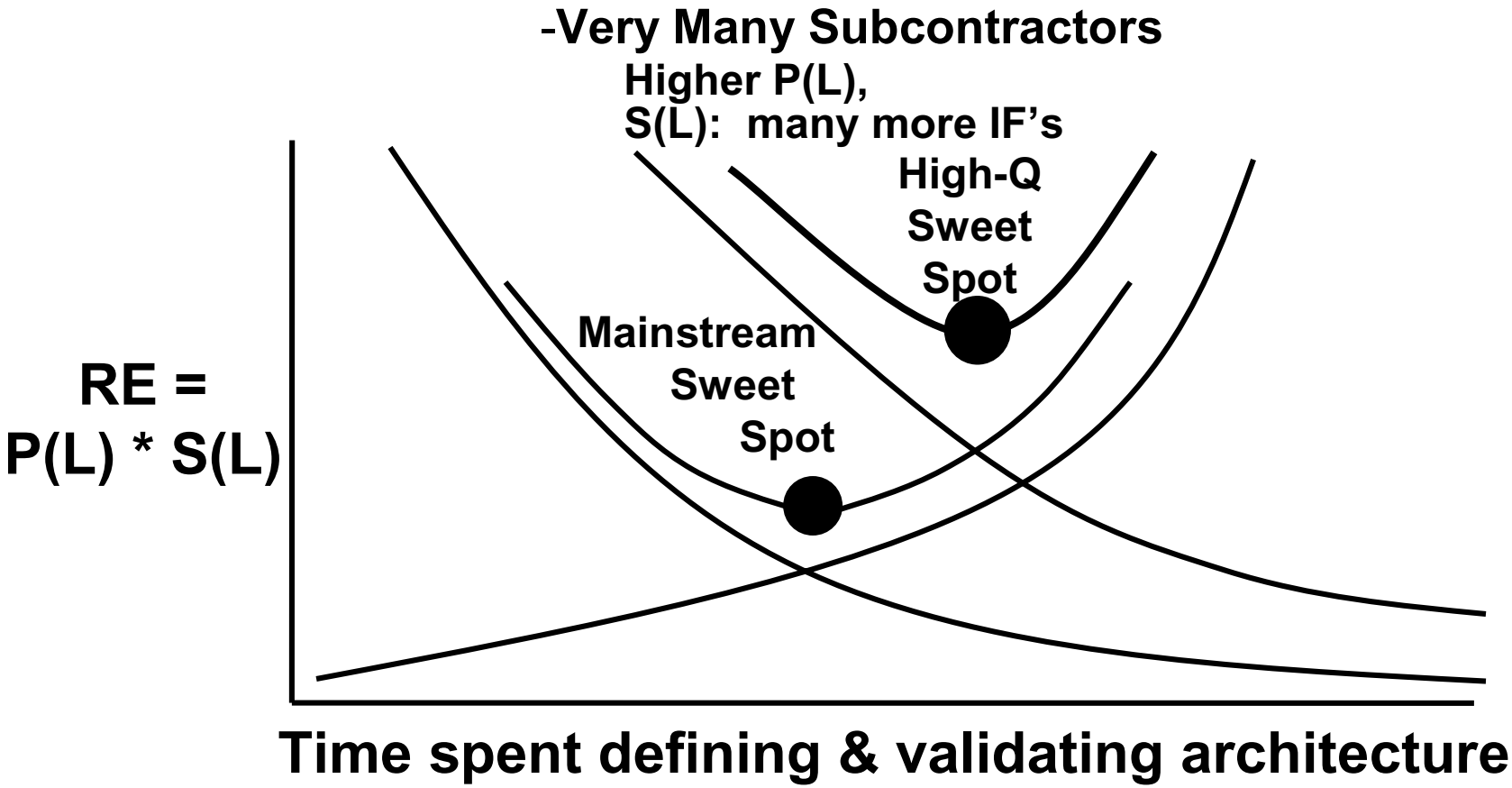
- Loss due to rework delays
- Loss due to late subcontact startups



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



How Soon to Define Subcontractor Interfaces?



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