Avoiding Overruns in the Specification of Non-Functional Requirements

Barry Boehm, USC

NDIA SE Conference 2015
October 29, 2015

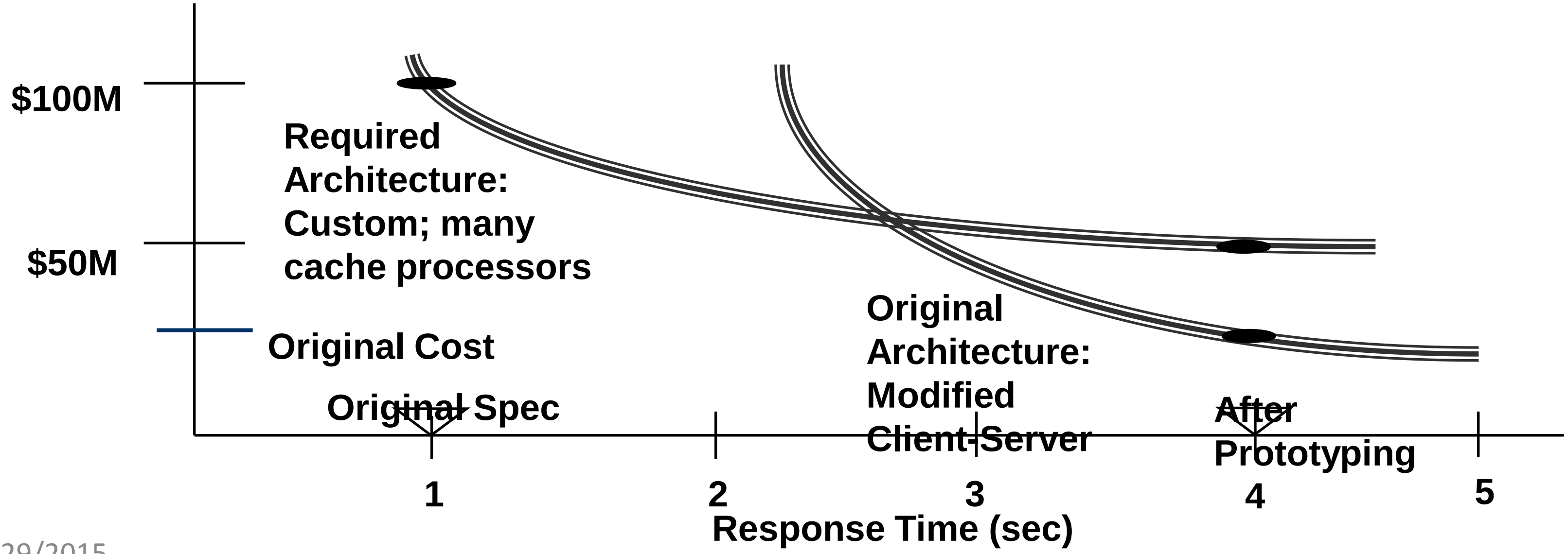
Outline

- ➡ Critical nature of non-functional requirements (NFRs)
  - Also called System Qualities, ilities
  - Major source of project overruns, failures
  - Poorly defined, understood
  - Underemphasized in project management
- Example sources of project overruns
  - The conspiracy of optimism and its effect on SE
  - Inflexible requirements
  - Overagile and Underagile Methods
  - Optimizing some NFRs at the expense of others: Security
  - Chaotic nature of NFR definition, understanding
    - DoD-SERC NFR definition, practices efforts
- Principles for avoiding the sources of project overruns

Critical Nature of NFRs

Major source of DoD, other system overruns

- NFRs have systemwide impact
  - System elements generally just have local impact
- NFRs often exhibit asymptotic behavior
  - Watch out for the knee of the curve
- Best architecture is a discontinuous function of NFR level
  - "Build it quickly, tune or fix it later" highly risky
  - Large system example below



Example of Current Practice

- “The system shall have a Mean Time Between Failures of 10,000 hours”
- What is a “failure?”
  - 10,000 hours on liveness
  - But several dropped or garbled messages per hour?
- What is the operational context?
  - Base operations? Field operations? Conflict operations?
- Most management practices focused on functions
  - Requirements, design reviews; traceability matrices; work breakdown structures; data item descriptions; earned value management
- What are the effects of or on other SQs?
  - Cost, schedule, performance, maintainability?

Proliferation of Definitions: Resilience

- Wikipedia Resilience variants: Climate, Ecology, Energy Development, Engineering and Construction, Network, Organizational, Psychological, Soil

- Ecology and Society Organization Resilience variants: Original-ecological, Extended-ecological, Walker et al. list, Folke et al. list; Systemic-heuristic, Operational, Sociological, Ecological-economic, Social-ecological system, Metaphoric, Sustainabilty-related

- Variants in resilience outcomes
  - Returning to original state; Restoring or improving original state; Maintaining same relationships among state variables; Maintaining desired services; Maintaining an acceptable level of service; Retaining essentially the same function, structure, and feedbacks; Absorbing disturbances; Coping with disturbances; Self-organizing; Learning and adaptation; Creating lasting value
  - Source of serious cross-discipline collaboration problems

Outline

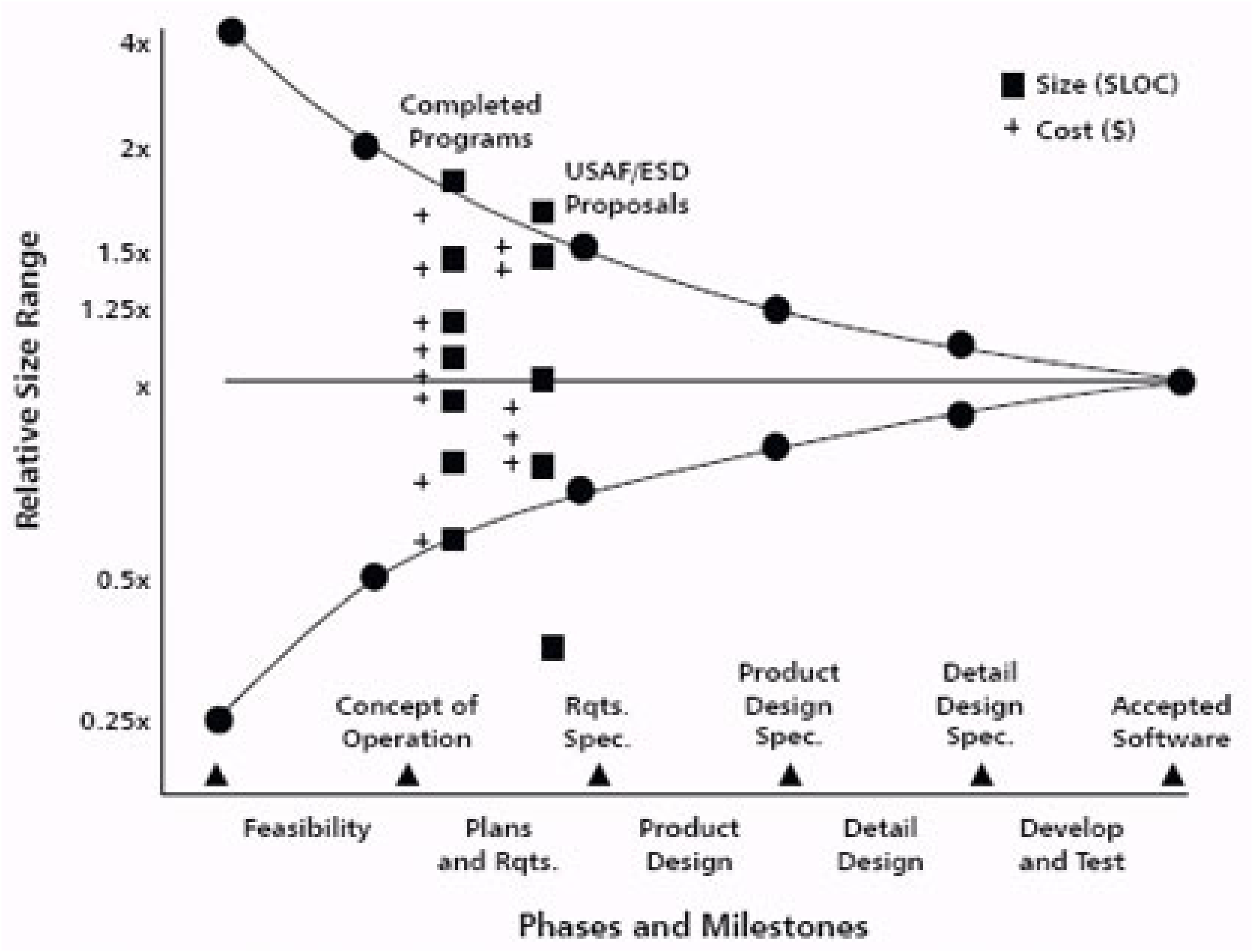
- **Critical nature of non-functional requirements (NFRs)**
  - **Also called System Qualities, ilities**
  - **Major source of project overruns, failures**
  - **Poorly defined, understood**
  - **Underemphasized in project management**

➡ **Example sources of project overruns**
  - **The conspiracy of optimism and its effect on SE**
  - **Inflexible requirements**
  - **Overagile and Underagile Methods**
  - **Optimizing some NFRs at the expense of others: Security**
  - **Chaotic nature of NFR definition, understanding**
    - **DoD-SERC NFR definition, practices efforts**

- **Principles for avoiding the sources of project overruns**

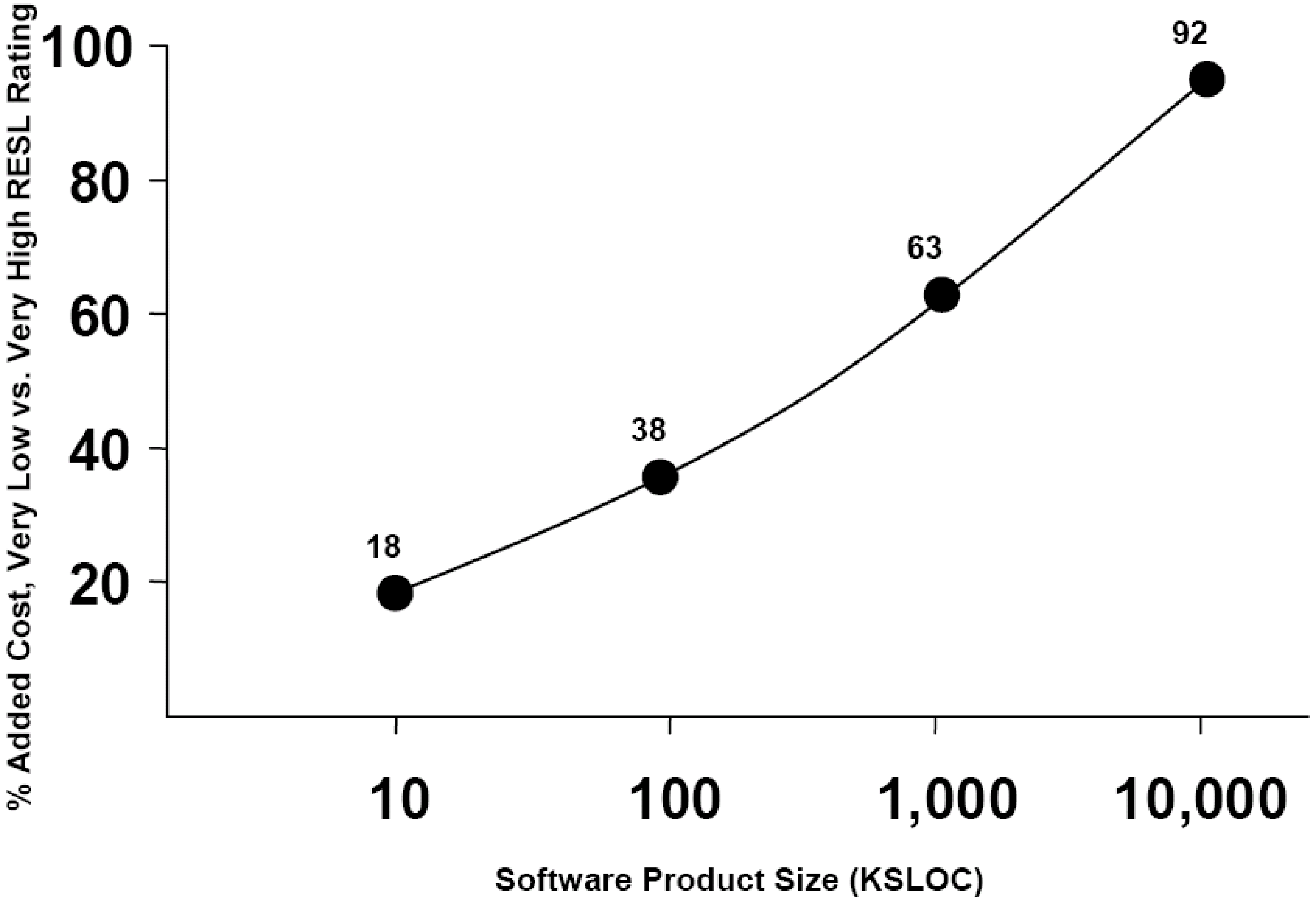
The Conspiracy of Optimism

Take the lower branch of the Cone of Uncertainty

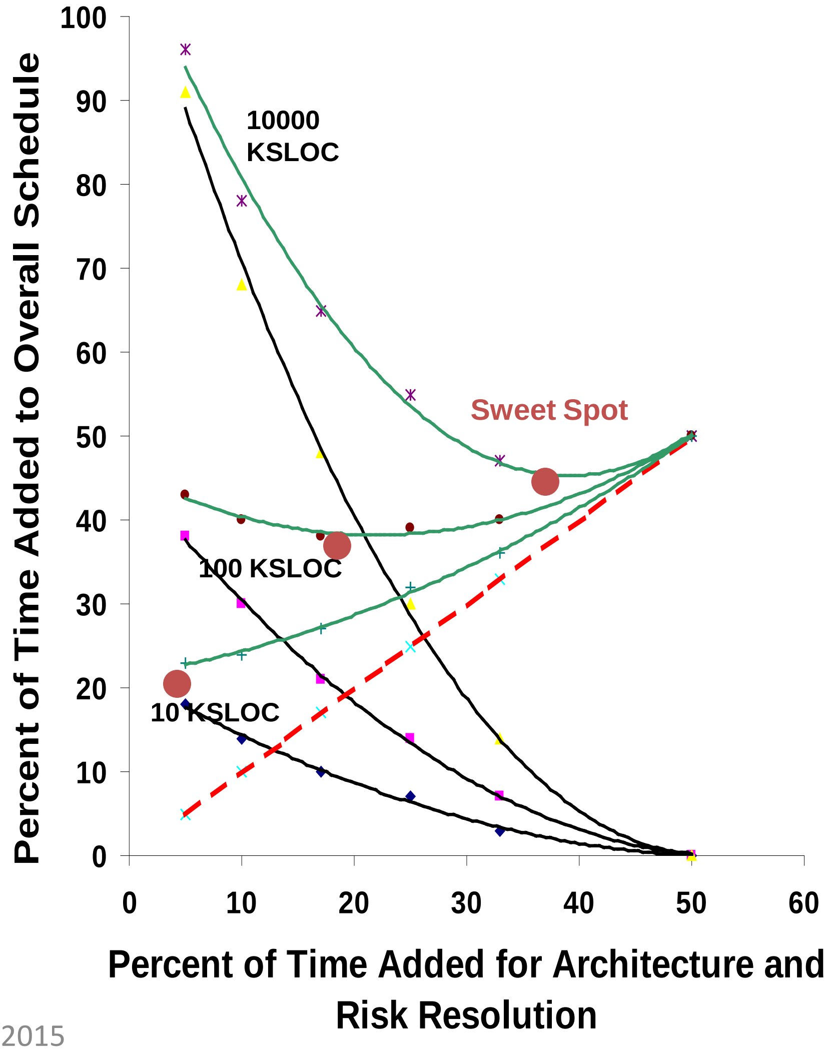


Added Cost of Minimal Software SysE

Based on COCOMO II calibration data



How Much Architecting is Enough?

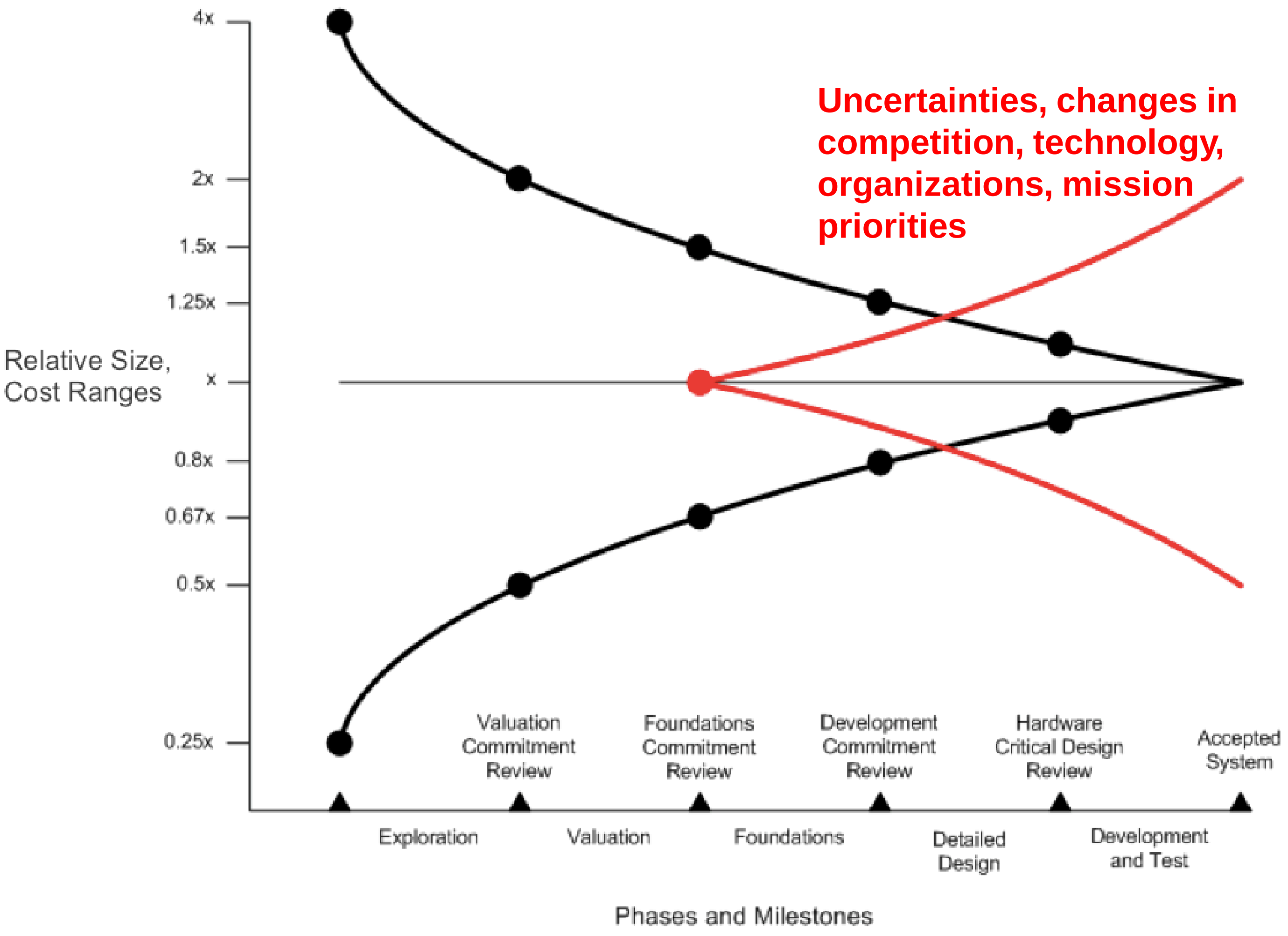


Sweet Spot Drivers:

Rapid Change: leftward

High Assurance: rightward

Inflexible Requirements: Dual Cones of Uncertainty – Obsolete large command and control system



Uncertainties, changes in competition, technology, organizations, mission priorities

Overagile and Underagile Methods

- Overagile Methods: Easiest First
  - Treat security, safety, scalability as user stories
  - Defer their development to late releases
  - Doing the easy parts will make the hard parts easier
    - Maybe for puzzles, but not for complex software-intensive systems

- Underagile Methods: Apply rigorous methods to all system parts
  - May need for some parts: security-critical, safety-critical
  - But not for others: user, evolving external-system interfaces
    - Particularly hard to change if included in contracts
      - Two systems of systems: 141 average workdays vs. 48
    - Important to modularize around sources of change
      - Avoids ripple effects on other system parts

Example of NFR Value Conflicts: Security IPT

- **Single-agent key distribution; single data copy**
  - **Reliability: single points of failure**
- **Elaborate multilayer defense**
  - **Performance: 50% overhead; real-time deadline problems**
- **Elaborate authentication**
  - **Usability: delays, delegation problems; GUI complexity**
- **Everything at highest level**
  - **Modifiability: overly complex changes, recertification**

Outline

- Critical nature of non-functional requirements (NFRs)
  - Also called System Qualities, ilities
  - Major source of project overruns, failures
  - Poorly defined, understood
  - Underemphasized in project management
- Example sources of project overruns
  - The conspiracy of optimism and its effect on SE
  - Inflexible requirements
  - Overagile and Underagile Methods
  - Optimizing some NFRs at the expense of others: Security
  - ➡ Chaotic nature of NFR definition, understanding
    - DoD-SERC NFR definition, practices efforts
- Principles for avoiding the sources of project overruns

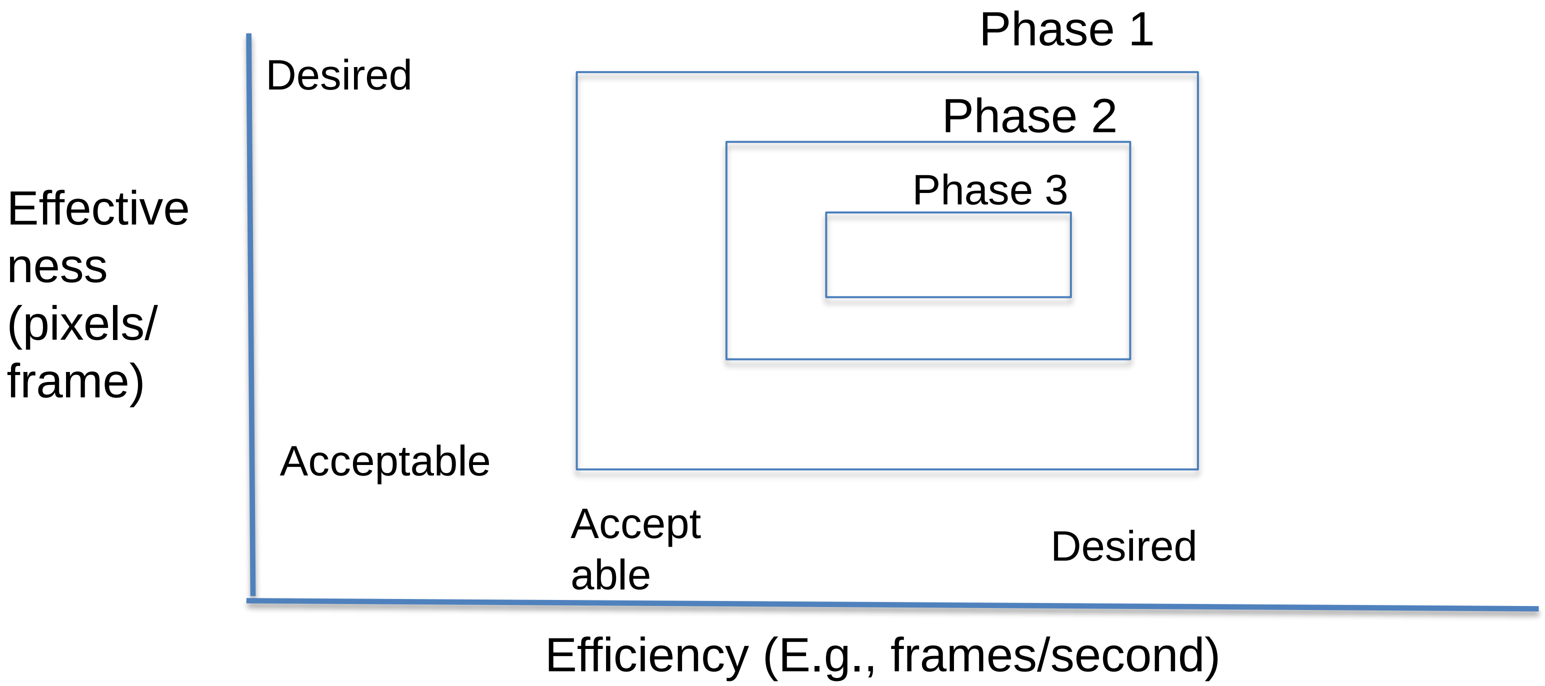
DoD-SERC NFR Definition, Practices Efforts:

NFR Ontology, Set-based requirements and design

- Modified version of IDEF5 ontology framework
  - Classes, Subclasses, and Individuals
  - Referents, States, Processes, and Relations
- Top classes cover stakeholder value propositions
  - Mission Effectiveness, Resource Utilization, Dependability, Changeabiity
- Subclasses identify means for achieving higher-class ends
  - Means-ends one-to-many for top classes
  - Ideally mutually exclusive and exhaustive, but some exceptions
  - Many-to-many for lower-level subclasses
- Referents, States, Processes, Relations cover NFR variation
  - Referents: Sources of variation by stakeholder value context:
  - States: Internal (beta-test); External (rural, temperate, sunny)
  - Processes: Operational scenarios (normal vs. crisis; experts vs. novices)
  - Relations: Impact of other SQs (security as above, synergies & conflicts)

Set-Based NFRs Definition Convergence

RPV Surveillance Example


Phase 1
Phase 2
Phase 3
Desired
Effectiveness (pixels/frame)
Acceptable
Acceptable
Desired
Efficiency (E.g., frames/second)

Phase 1. Rough ConOps, Rqts, Solution Understanding
Phase 2. Improved ConOps, Rqts, Solution Understanding
Phase 3. Good ConOps, Rqts, Solution Understanding

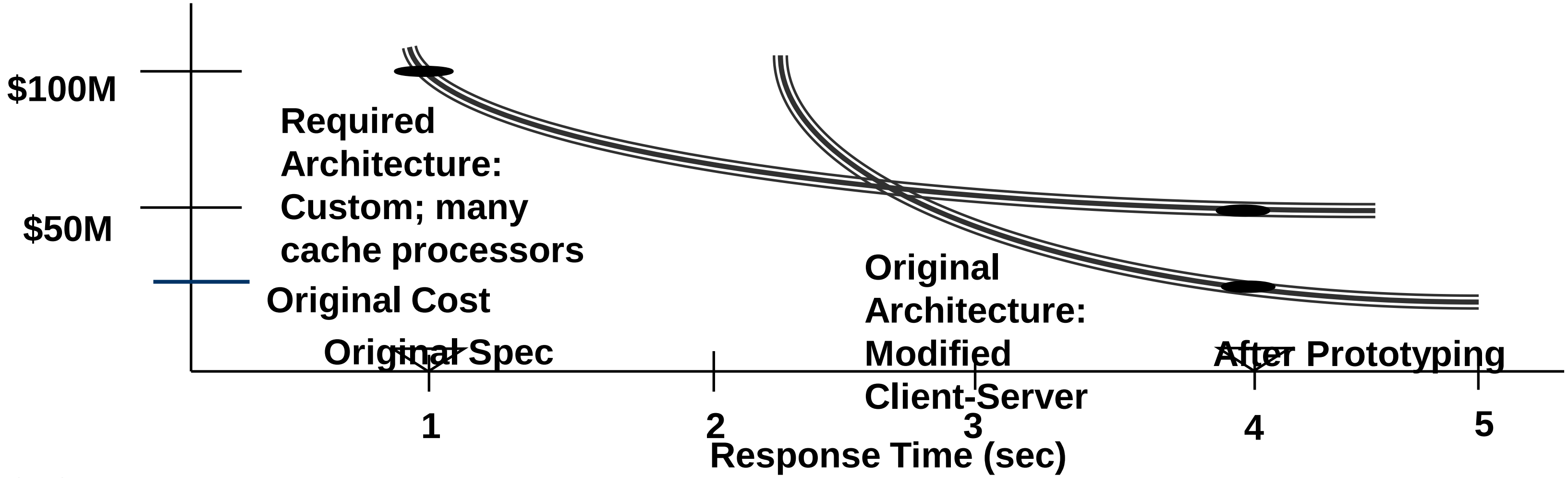
Outline

- Critical nature of non-functional requirements (NFRs)
  - Also called System Qualities, ilities
  - Major source of project overruns, failures
  - Poorly defined, understood
  - Underemphasized in project management
- Example sources of project overruns
  - The conspiracy of optimism and its effect on SE
  - Inflexible requirements
  - Overagile and Underagile Methods
  - Optimizing some NFRs at the expense of others: Security
  - Chaotic nature of NFR definition, understanding
    - DoD-SERC NFR definition, practices efforts

➡ Principles for avoiding the sources of project overruns

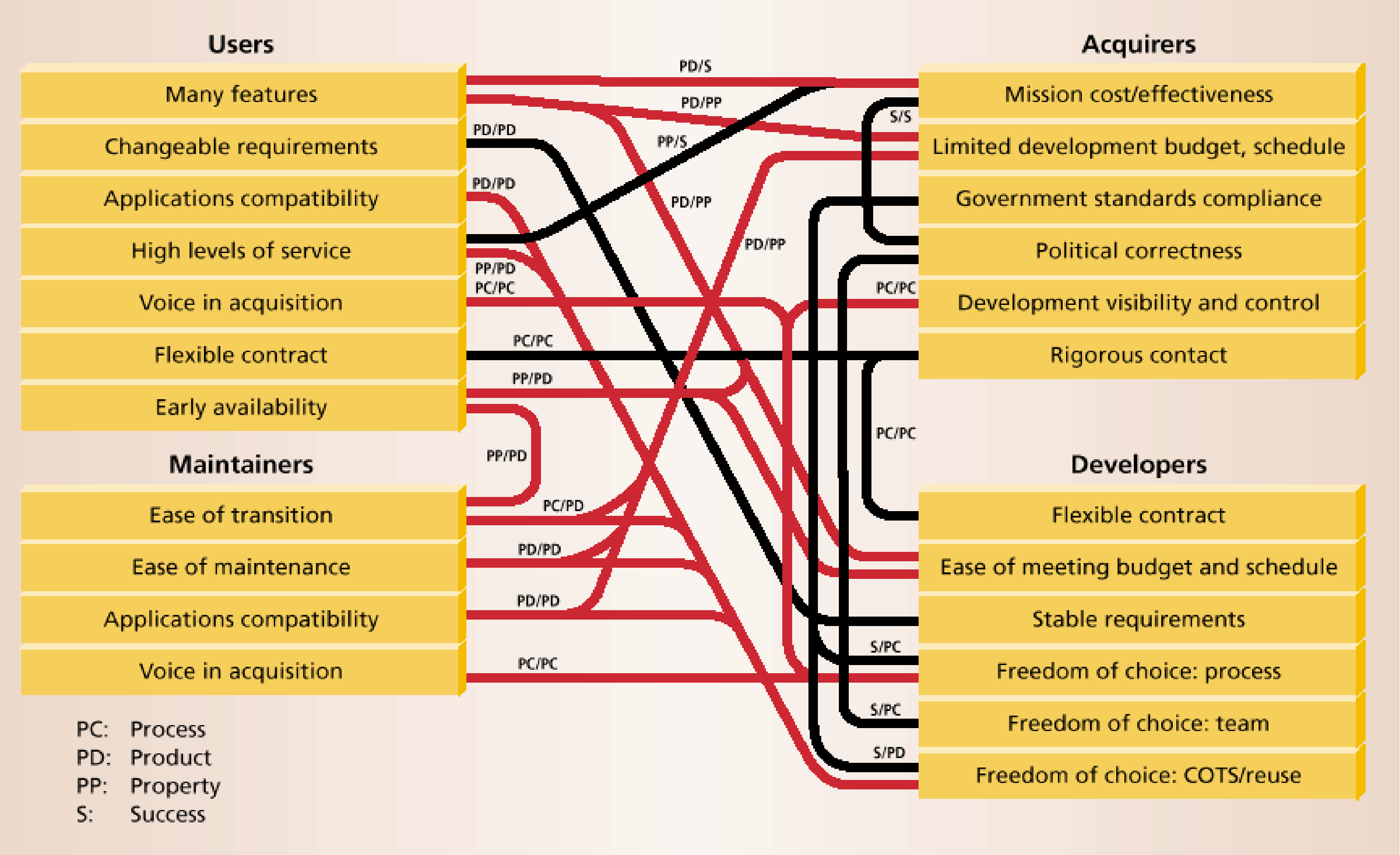
Principles for Avoiding Project Overrun Sources

- Stakeholder Value-Based Guidance
  - Identify, involve missing success-critical stakeholders
  - Bank of America Master Net example
- Incremental Commitment and Accountability
  - Set-Based Requirements and Design: Build in Tradespace
- Concurrent Multi-Discipline Engineering
  - Complement NFR IPTs with NFRs IPT
- Evidence and Risk-Based Decisions



ICSM Principles Counterexample:

Bank of America Master Net



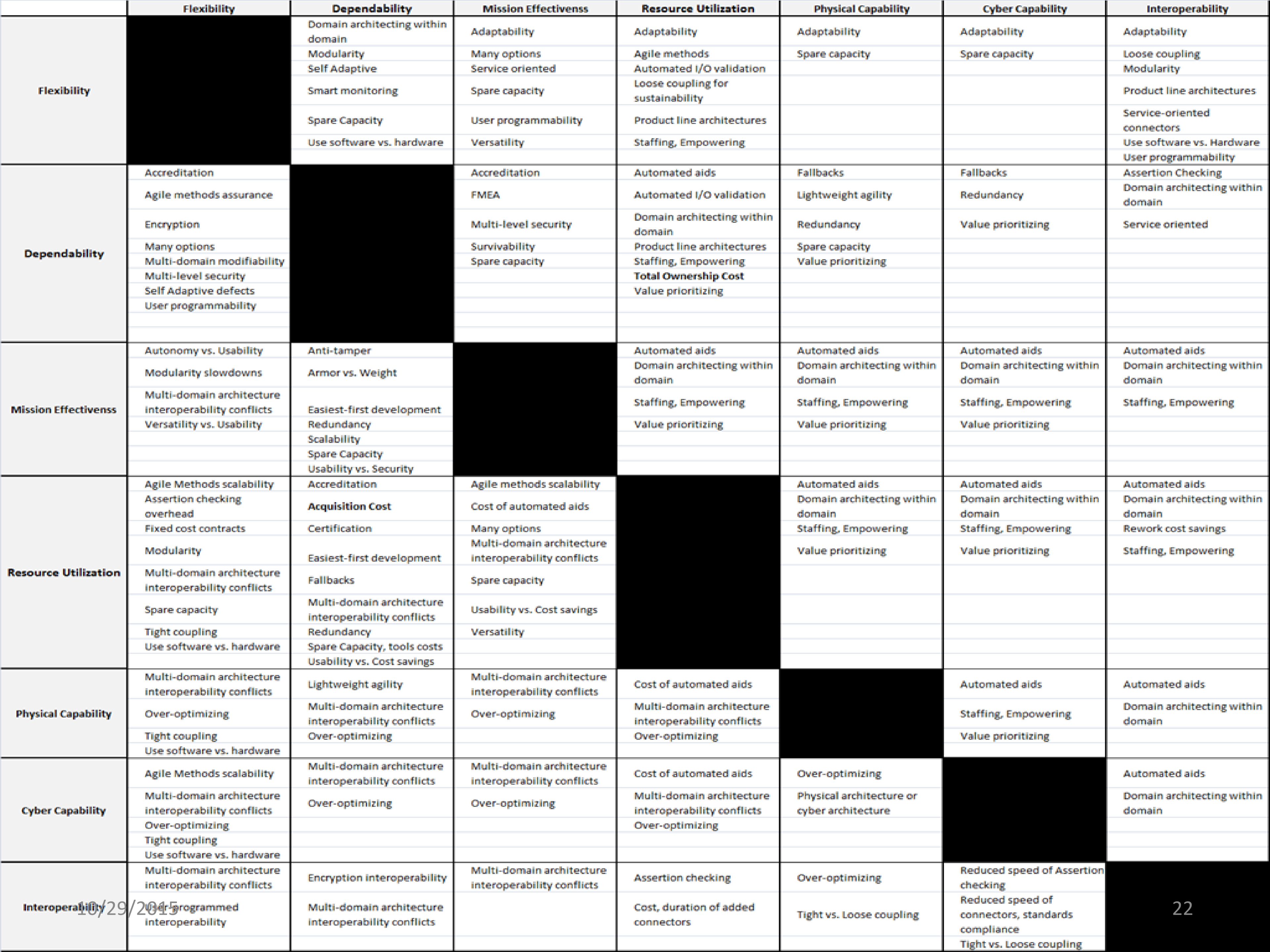
Types of Decision Reviews

- **Schedule-based commitment reviews (plan-driven)**
  - **We'll release the RFP on April 1 based on the schedule in the plan**
  - **$70M overrun to produce overperforming system**
- **Event-based commitment reviews (artifact-driven)**
  - **The design will be done in 15 months, so we'll have the review then**
  - **Responsive design found to be unaffordable 15 months later**
- **Evidence-based commitment reviews (risk-driven)**
  - **Evidence: affordable COTS-based system can't satisfy 1-second requirement**
    - **Custom solution roughly 3x more expensive**
  - **Need to reconsider 1-second requirement**

Evidence-Based Decision Result

- Attempt to validate 1-second response time
  - Commercial system benchmarking and architecture analysis: needs expensive custom solution
  - Prototype: 4-second response time OK 90% of the time
- Negotiate response time ranges
  - 2 seconds desirable
  - 4 seconds acceptable with some 2-second special cases
- Benchmark commercial system add-ons to validate their feasibility
- Present solution and feasibility evidence at evidence-based decision review
  - Result: Acceptable solution with minimal delay

Backup charts



	Flexibility	Dependability	Mission Effectivenss	Resource Utilization	Physical Capability	Cyber Capability	Interoperability
Flexibility		Domain architecting within domain Modularity Self Adaptive Smart monitoring Spare Capacity Use software vs. hardware	Adaptability Many options Service oriented Spare capacity User programmability Versatility	Adaptability Agile methods Automated I/O validation Loose coupling for sustainability Product line architectures Staffing, Empowering	Adaptability Spare capacity	Adaptability Spare capacity	Adaptability Loose coupling Modularity Product line architectures Service-oriented connectors Use software vs. Hardware User programmability
Dependability	Accreditation Agile methods assurance Encryption Many options Multi-domain modifiability Multi-level security Self Adaptive defects User programmability		Accreditation FMEA Multi-level security Survivability Spare capacity	Automated aids Automated I/O validation Domain architecting within domain Product line architectures Staffing, Empowering **Total Ownership Cost** Value prioritizing	Fallbacks Lightweight agility Redundancy Spare capacity Value prioritizing	Fallbacks Redundancy Value prioritizing	Assertion Checking Domain architecting within domain Service oriented
Mission Effectivenss	Autonomy vs. Usability Modularity slowdowns Multi-domain architecture interoperability conflicts Versatility vs. Usability	Anti-tamper Armor vs. Weight Easiest-first development Redundancy Scalability Spare Capacity Usability vs. Security		Automated aids Domain architecting within domain Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain Staffing, Empowering
Resource Utilization	Agile Methods scalability Assertion checking overhead Fixed cost contracts Modularity Multi-domain architecture interoperability conflicts Spare capacity Tight coupling Use software vs. hardware	Accreditation **Acquisition Cost** Certification Easiest-first development Fallbacks Multi-domain architecture interoperability conflicts Redundancy Spare Capacity, tools costs Usability vs. Cost savings	Agile methods scalability Cost of automated aids Many options Multi-domain architecture interoperability conflicts Spare capacity Usability vs. Cost savings Versatility		Automated aids Domain architecting within domain Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain Rework cost savings Staffing, Empowering
Physical Capability	Multi-domain architecture interoperability conflicts Over-optimizing Tight coupling Use software vs. hardware	Lightweight agility Multi-domain architecture interoperability conflicts Over-optimizing	Multi-domain architecture interoperability conflicts Over-optimizing	Cost of automated aids Multi-domain architecture interoperability conflicts Over-optimizing		Automated aids Staffing, Empowering Value prioritizing	Automated aids Domain architecting within domain
Cyber Capability	Agile Methods scalability Multi-domain architecture interoperability conflicts Over-optimizing Tight coupling Use software vs. hardware	Multi-domain architecture interoperability conflicts Over-optimizing	Multi-domain architecture interoperability conflicts Over-optimizing	Cost of automated aids Multi-domain architecture interoperability conflicts Over-optimizing	Over-optimizing Physical architecture or cyber architecture		Automated aids Domain architecting within domain
Interoperability	Multi-domain architecture interoperability conflicts User programmed interoperability	Encryption interoperability Multi-domain architecture interoperability conflicts	Multi-domain architecture interoperability conflicts	Assertion checking Cost, duration of added connectors	Over-optimizing Tight vs. Loose coupling	Reduced speed of Assertion checking Reduced speed of connectors, standards compliance Tight vs. Loose coupling	

	Security	Reliability	Maintainability
Security		Confidentiality, Integrity, Avalability Assurance Cases Certification Failure Modes and Effects Analysis Fault Tree Analysis Recertification	Certification Diagnosability Integrity, Avalability Repairability Smart Monitoring Spare Capacity
Reliability	Non-redundancy (For Security) Redundancy (For Reliability)		Accessibility Certification Diagnosability Repairability Smart Monitoring Spare Capacity
Maintainability	Accessibility Compartmentalization Encryption Recertification	Armor Recertification	