System Qualities (SQs) Tradespace and Affordability

Barry Boehm, USC

SERC SSRR 2016
November 17, 2016
PROJECT DESCRIPTION:

• Develop and build upon firm scientific foundations for reasoning about tradespaces among System Qualities, particularly for Life Cycle Affordability, using its MORS and INCOSE definition as Cost-Effectiveness. Develop, pilot, refine, and transition improved SQ tradespace methods, processes, and tools (MPTs), using set-based design tradespaces, versus current point-solution designs.

VALUE:

• Being able to quickly and rigorously analyze the tradespace of complex systems, especially with regard to DoD-critical SQs such as security, resilience, adaptability, usability, interoperability, and affordability, will aid decision makers early in the life cycle in a project when alternative solutions are all under consideration.

CONTRIBUTOR(S):

• University of Southern California, Georgia Institute of Technology, Massachusetts Institute of Technology, Stevens Institute of Technology, University of Virginia, Wayne State University, Air Force Institute of Technology, Naval Post-graduate School, Pennsylvania State University
RESEARCH AREAS:
Research divided into three areas

• Foundations and Frameworks: Developing a formally-based, systems engineering and management-based ontology for the SQs, their sources of variation, and their tradespace interactions
• Full-Lifecycle, Set-Based SQ Tradespace Methods, Processes, and Tools Extension and Demonstration
• Next Generation, Full Life Cycle Cost, Schedule, and Quality Estimation Models for System Engineering and Software-Intensive Systems

Non-ASD(R&E) Sponsors

• USAF ASC, SMC, AFCAA; USA ERDC, TARDEC; USMC; USN NAVSEA, NSWC, NCCA
Critical nature of system qualities (SQs)
- Or non-functional requirements (NFRs); ilities
- Major source of project overruns, failures
- Underemphasized in project management
- Poorly defined, understood

Foundations: An initial SQs ontology
- Nature of an ontology; choice of IDEF5 structure
- Stakeholder value-based, means-ends hierarchy
- Synergies and Conflicts matrix and expansions
- Maintainability deep dive results to date

Future plans
Importance of SQ Tradeoffs

Major source of system overruns

- SQs have systemwide impact
  - System elements generally just have local impact
- SQs often exhibit asymptotic behavior
  - Watch out for the knee of the curve
- Best architecture is a discontinuous function of SQ level
  - “Build it quickly, tune or fix it later” highly risky
- Large system example
  - COTS-based Dept-level system successful with 1-second response
  - Contract to extend to full organization with 1-second response
  - COTS-based system unscalable; custom solution developed
    - Cost: $100 million vs. $30 million budget
    - Prototyping found 4-second response OK 90% of time
    - COTS-based system workable; changing 1 character in SQ requirement changed cost from $100M to $30M
Example of SQ 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
Set-Based SQs Definition Convergence Enables Systems Engineering Tradespace

Phase 1. Rough ConOps, Rqts, Solution Understanding
Phase 2. Improved ConOps, Rqts, Solution Understanding
Phase 3. Good ConOps, Rqts, Solution Understanding
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, Sustainability-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
• Example: Definition of Reliability:
  – The degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time
  – OK if specifications are precise, but increasingly “specified conditions” are informal, sunny-day user stories.
    • Satisfying just these will pass “ISO/IEC Reliability,” even if system fails on rainy-day user stories (bad data, communications, users)
    • Similarly for unspecified quality requirements, e.g., security

• Need to reflect diversity
  – Different stakeholders rely on different capabilities (functions, performance, flexibility, etc.) at different times and in different environments

• Quality definitions need a more precise ontology
Outline

• Critical nature of system qualities (SQs)
  – Or non-functional requirements (NFRs); ilities
  – Major source of project overruns, failures
  – Underemphasized in project management
  – Poorly defined, understood

  Foundations: An initial SQs ontology
  – Nature of an ontology; choice of IDEF5 structure
  – Stakeholder value-based, means-ends hierarchy
  – Synergies and Conflicts matrix and expansions
  – Maintainability deep dive results to date

• Future plans
Nature of an ontology; choice of IDEF5 structure

• An ontology for a collection of elements is a definition of what it means to be a member of the collection

• For “system qualities,” this means that an SQ identifies an aspect of “how well” the system performs
  – The ontology also identifies the sources of variability in the value of “how well” the system performs
    • Functional requirements specify “what;” NFRs specify “how well”

• After investigating several ontology frameworks, the IDEF5 framework appeared to best address the nature and sources of variability of system SQs
  – Good fit so far
Current SERC SQs Ontology

• 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, Changeability

• 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 SQ variation
  • Referents: Sources of variation by stakeholder value context
  • States: Internal (beta-test); External (infrastructure, interoperators)
  • Processes: Operational scenarios (normal vs. crisis; experts vs. novices)
  • Relations: Impact of other SQs (security as above, synergies & conflicts)
Mission operators and managers want improved Mission Effectiveness
  - Involves Physical Capability, Cyber Capability, Human Usability, Speed, Endurabilty, Maneuverability, Accuracy, Impact, Scalability, Versatility, Interoperability

Mission investors and system owners want Life Cycle Efficiency
  - Involves Development and Maintenance Cost, Duration, Key Personnel, Scarce Quantities (capacity, weight, energy, ...); Manufacturability, Sustainability

All want system Dependability: cost-effective defect-freedom, availability, and safety and security for the communities that they serve
  - Involves Reliability, Maintainability, Availability, Survivability, Robustness, Graceful Degredation, Safety, Security

In an increasingly dynamic world, all want system Changeability: to be rapidly and cost-effectively evolvable
  - Involves Maintainability, Modifiability, Repairability, Adaptability
Reliability is the probability that the system will deliver stakeholder-satisfactory results for a given time period (generally an hour), given specified ranges of:

- Stakeholder value propositions: desired and acceptable ranges of liveness, accuracy, response time, speed, capabilities, etc.
- System internal and external states: integration test, acceptance test, field test, etc.; weather, terrain, DEFCON, takeoff/flight/landing, etc.
- System internal and external processes: status checking frequency; types of missions supported; workload volume, interoperability with independently evolving external systems
- Effects via relations with other SQs: synergies improving other SQs; conflicts degrading other SQs
Current SERC SQs Ontology

• 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, Changeability

• 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 SQ variation

• Referents: Sources of variation by stakeholder value context
• States: Internal (test vs. use); External (infrastructure, interoperators)
• Processes: Operational scenarios (normal vs. crisis workload)
• Relations: Impact of other SQs (security as above, synergies & conflicts)
• Mission Effectiveness expanded to 4 elements
  – Physical Capability, Cyber Capability, Interoperability, Other
    Mission Effectiveness (including Usability as Human Capability)

• Synergies and Conflicts among the 7 resulting elements
  identified in 7x7 matrix
  – Synergies above main diagonal, Conflicts below
  – Ideally quantitative; example next

• Still need synergies and conflicts within elements
  – Example 3x3 Dependability subset provided
<table>
<thead>
<tr>
<th>Flexibility</th>
<th>Dependability</th>
<th>Mission Effectiveness</th>
<th>Resource Utilization</th>
<th>Physical Capability</th>
<th>Cyber Capability</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain architecting within domain</td>
<td>Modularity</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
</tr>
<tr>
<td>Modularity</td>
<td>Self Adaptive</td>
<td>Service oriented</td>
<td>Loose coupling</td>
<td>Spare capacity</td>
<td>Loose coupling</td>
<td>Loose coupling</td>
</tr>
<tr>
<td>Smart monitoring</td>
<td>Spare Capacity</td>
<td>Product line architectures</td>
<td>Versatility</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accreditation</td>
<td></td>
<td>FMEA</td>
<td>Domain architecting within domain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile methods assurance</td>
<td></td>
<td>Automated I/O validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
<td>Multi-level security</td>
<td>Staffing, Empowering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Many options</td>
<td></td>
<td>Survivability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain modifiability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-level security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self Adaptive defects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User programmability</td>
<td></td>
<td>Total Ownership Cost</td>
<td>Value prioritizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value prioritizing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autonomy vs. Usability</td>
<td>Anti-tamper</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Automated aids</td>
</tr>
<tr>
<td>Modularity slowdowns</td>
<td>Armor vs. Weight</td>
<td>Domain architecting within domain</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Easiest-first development</td>
<td>Staffing, Empowering</td>
<td></td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
</tr>
<tr>
<td>Versatility vs. Usability</td>
<td>Redundancy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>Spare Capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Sparsity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile Methods scalability</td>
<td>Acquisition Cost</td>
<td>Agile methods scalability</td>
<td>Cost of automated aids</td>
<td></td>
<td>Automated aids</td>
<td>Automated aids</td>
</tr>
<tr>
<td>Assertion checking overhead</td>
<td>Certification</td>
<td>Cost of automated aids</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed cost contracts</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modularity</td>
<td>Easiest-first development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Fallbacks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spare capacity</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Usability vs. Security</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Spare Capacity, tools costs</td>
<td>Versatility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Lightweight agility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Over-optimizing</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile Methods scalability</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td>Automated aids</td>
<td>Automated aids</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption Interoperability</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td>Over-optimizing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical architecture or cyber architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyber Security</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

18
• Critical nature of system qualities (SQs)
  – Or non-functional requirements (NFRs); ilities
  – Major source of project overruns, failures
  – Underemphasized in project management
  – Poorly defined, understood

• Foundations: An initial SQs ontology
  – Nature of an ontology; choice of IDEF5 structure
  – Stakeholder value-based, means-ends hierarchy
  – Synergies and Conflicts matrix and expansions

  Maintainability deep dive results to date

• Future plans
Product Quality View of Changeability

MIT Quality in Use View also valuable

- **Changeability (PQ):** Ability to become different product
  - Swiss Army Knife: Versatile but not Changeable

- **Changeability (Q in Use):** Ability to accommodate changes in use
  - Swiss Army Knife doesn’t change as a product but is Changeable in use
### Prescriptive Semantic Basis for Change-type Illities

In response to "cause" in "context", desire "agent" to make some "change" in "system" that is "valuable".

In response to "perturbation" in "context" during "phase" desire "agent" to make some "nature" impetus to the design "parameter" with "destination(s)" in the "aspect" to have an "effect" to the outcome "parameter" with "destination(s)" in the "aspect" of the "abstraction" that are valuable with respect to thresholds in "reaction", "span", "cost" and "benefit".

<table>
<thead>
<tr>
<th>Perturbation</th>
<th>Context</th>
<th>Phase</th>
<th>Agent</th>
<th>Impetus</th>
<th>Nature</th>
<th>Parameter</th>
<th>Destination</th>
<th>Aspect</th>
<th>Effect</th>
<th>Parameter</th>
<th>Destination</th>
<th>Aspect</th>
<th>Abstraction</th>
<th>Reaction</th>
<th>Span</th>
<th>Cost</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>disturbance</td>
<td>general</td>
<td>pre-ops</td>
<td>ops</td>
<td>internal</td>
<td>increase</td>
<td>level</td>
<td>one</td>
<td>form</td>
<td>increase</td>
<td>level</td>
<td>one</td>
<td>form</td>
<td>architecture</td>
<td>sooner</td>
<td>shorter</td>
<td>less</td>
<td>more</td>
</tr>
<tr>
<td>shift</td>
<td>general</td>
<td>ops</td>
<td>not-same</td>
<td>ops</td>
<td>not-same</td>
<td>any</td>
<td>many</td>
<td>operations</td>
<td>not-same</td>
<td>any</td>
<td>many</td>
<td>operations</td>
<td>system</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>none</td>
<td>any</td>
<td>inter-LC</td>
<td>ether</td>
<td>none</td>
<td>same</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>shift</td>
<td>general</td>
<td>ops</td>
<td>none</td>
<td>ops</td>
<td>not-same</td>
<td>same</td>
<td>level</td>
<td>few</td>
<td>form</td>
<td>system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>disturbance</td>
<td>general</td>
<td>inter-LC</td>
<td>ops</td>
<td>ops</td>
<td>not-same</td>
<td>one</td>
<td>form</td>
<td>not-same</td>
<td>&quot;Element set&quot;</td>
<td>one</td>
<td>form</td>
<td>not-same</td>
<td>&quot;Link set&quot;</td>
<td>form</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ops</td>
<td>general</td>
<td>inter-LC</td>
<td>ops</td>
<td>ops</td>
<td>not-same</td>
<td>one</td>
<td>form</td>
<td>not-same</td>
<td>&quot;Order set&quot;</td>
<td>operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ops</td>
<td>general</td>
<td>inter-LC</td>
<td>ops</td>
<td>ops</td>
<td>not-same</td>
<td>one</td>
<td>form</td>
<td>not-same</td>
<td>&quot;Function set&quot;</td>
<td>few</td>
<td>many</td>
<td>function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ops</td>
<td>general</td>
<td>inter-LC</td>
<td>ops</td>
<td>ops</td>
<td>not-same</td>
<td>one</td>
<td>form</td>
<td>not-same</td>
<td>&quot;Operation set&quot;</td>
<td>few</td>
<td>many</td>
<td>operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Illity Label
- Value Robustness
- Value Survivability
- Robustness
- Active Robustness
- Passive Robustness
- Classical Passive Robustness
- Survivability
- Changeability
- Evolvability
- Adaptability
- Flexibility
- Scalability
- Modifiability
- Extensibility
- Agility
- Reactivity
- Form Reconfigurability
- Operational Reconfigurability
- Versatility
- Functional Versatility
- Operational Versatility
- Substitutability
Initial Empirical Study: Evaluate SW Maintainability Index on Open Source Projects

- Evaluate MI across 97 open source projects
  - 3 programming languages: Java, PHP, Python
  - 5 domains: Web development framework, System administration, Test tools, Security/Encryption, Audio-Video
- Test MI invariance across languages, domains
- Evaluate completeness of MI vs. other sources
  - COCOMO II Software Understandability factors
    - Structuredness (cohesion, coupling)
    - Self-descriptiveness (documentation quality)
    - Application clarity (software reflects application content)
  - Other maintainability enablers (architecture, V&V support)
    - Repairability: Diagnosability, Accessibility, Testability, Tool support
    - Search for similar defects; root cause analysis
MI Variation among domains

- Web Development Framework has shown the highest medians and the highest maximum value.
- Audio and Video has both the lowest maximum value and the lowest median value
- **PHP** may be a good option for projects that desires higher maintainability within Web Development Framework, Security/Cryptography and Audio and Video domain,
- **Python** may be a good option for System Administrative Software
- **Java** may be a good option for Software Testing Tools.
What is Technical Debt (TD)?

• TD: Delayed technical work or rework that is incurred when short-cuts are taken or short-term needs are addressed first
  – The later you pay for it, the more it costs (interest on debt)

• Global Information Technology Technical Debt [Gartner 2010]
  – 2010: Over $500 Billion; By 2015: Over $1 Trillion

• TD as Investment
  – Competing for first-to-market
  – Risk assessment: Build-upon prototype of key elements
  – Rapid fielding of defenses from terrorist threats

• TD as Lack of Foresight
  – Overfocus on Development vs. Life Cycle
  – Skimping on Systems Engineering
  – Hyper-Agile Development: Easiest-First increments
    • Neglecting Rainy-Day Use Cases, Non-Functional Requirements
Top-10 Non-Technical Sources of TD

1. Separate organizations and budgets for systems and software acquisition and maintenance
2. Overconcern with the Voice of the Customer
3. The Conspiracy of Optimism
4. Inadequate system engineering resources
5. Hasty contracting that focuses on fixed operational requirements
6. CAIV-limited system requirements
7. Brittle, point-solution architectures
8. The Vicious Circle
9. Stovepipe systems
10. Over-extreme forms of agile development
# SIS Maintainability Readiness Framework (SMRF)

## Software-Intensive Systems Maintainability Readiness Levels

<table>
<thead>
<tr>
<th>SMR Level</th>
<th>OpCon, Contracting: Missions, Scenarios, Resources, Incentives</th>
<th>Personnel Capabilities and Participation</th>
<th>Enabling Methods, Processes, and Tools (MPTs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>5 years of successful maintenance operations, including outcome-based incentives, adaptation to new technologies, missions, and stakeholders</td>
<td>In addition, creating incentives for continuing effective maintainability, performance on long-duration projects</td>
<td>Evidence of improvements in innovative O&amp;M MPTs based on ongoing O&amp;M experience</td>
</tr>
<tr>
<td>8</td>
<td>One year of successful maintenance operations, including outcome-based incentives, refinements of OpCon.</td>
<td>Stimulating and applying People CMM Level 5 maintainability practices in continuous improvement and innovation in such technology areas as smart systems, use of multicore processors, and 3-D printing</td>
<td>Evidence of MPT improvements based on ongoing refinement, and extensions of ongoing evaluation, initial O&amp;M MPTs.</td>
</tr>
<tr>
<td>7</td>
<td>System passes Maintainability Readiness Review with evidence of viable OpCon, Contracting, Logistics, Resources, Incentives, personnel capabilities, enabling MPTs</td>
<td>Achieving advanced People CMM Level 4 maintainability capabilities such as empowered work groups, mentoring, quantitative performance management and competency-based assets, particularly across key domains.</td>
<td>Advanced, integrated, tested, and exercised full-LC MBS&amp;SE MPTs and Maintainability-other-SQ tradespace analysis</td>
</tr>
<tr>
<td>6</td>
<td>Mostly-elaborated maintainability OpCon. with roles, responsibilities, workflows, logistics management plans with budgets, schedules, resources, staffing, infrastructure and enabling MPT choices, V&amp;V and review procedures.</td>
<td>Achieving basic People CMM levels 2 and 3 maintainability practices such as maintainability work environment, competency and career development, and performance management especially in such key areas such as V&amp;V, identification &amp; reduction of technical debt.</td>
<td>Advanced, integrated, tested full-LC Model-Based Software &amp; Systems (MBS&amp;SE) MPTs and Maintainability-other-SQ tradespace analysis tools identified for use, and being individually used and integrated.</td>
</tr>
<tr>
<td>5</td>
<td>Convergence, involvement of main maintainability success-critical stakeholders. Some maintainability use cases defined. Rough maintainability OpCon, other success-critical stakeholders, staffing, resource estimates. Preparation for NDI and outsource selections.</td>
<td>In addition, independent maintainability experts participate in project evidence-based decision reviews, identify potential maintainability conflicts with other SQs</td>
<td>Advanced full-lifecycle (full-LC) O&amp;M MPTs and SW/SE MPTs identified for use. Basic MPTs for tradespace analysis among maintainability &amp; other SQs, including TCO being used.</td>
</tr>
<tr>
<td>2</td>
<td>Mission evolution directions and maintainability implications explored. Some mission use cases defined, some O&amp;M options explored.</td>
<td>Highly maintainability-capable SysEs included in Early SysE team.</td>
<td>Initial exploration of O&amp;M MPT options</td>
</tr>
<tr>
<td>1</td>
<td>Focus on mission opportunities, needs. Maintainability not yet considered</td>
<td>Awareness of needs for early expertise for maintainability. concurrent engr’g, O&amp;M integration, Life Cycle cost estimation</td>
<td>Focus on O&amp;M MPT options considered</td>
</tr>
</tbody>
</table>
# SIS Maintainability Readiness Levels 3-5

## Software-Intensive Systems Maintainability Readiness Framework (SMRF)

<table>
<thead>
<tr>
<th>SMR Level</th>
<th>OpCon, Contracting: Missions, Scenarios, Resources, Incentives</th>
<th>Personnel Capabilities and Participation</th>
<th>Enabling Methods, Processes, and Tools (MPTs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5</strong></td>
<td>Convergence, involvement of main maintainability success-critical stakeholders. Some maintainability use cases defined. Rough maintainability OpCon, other success-critical stakeholders, staffing, resource estimates. Preparation for NDI and outsource selections.</td>
<td>In addition, independent maintainability experts participate in project evidence-based decision reviews, identify potential maintainability conflicts with other SQs</td>
<td>Advanced full-lifecycle (full-LC) O&amp;M MPTs and SW/SE MPTs identified for use. Basic MPTs for tradespace analysis among maintainability &amp; other SQs, including TCO being used.</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Artifacts focused on missions. Primary maintenance options determined, Early involvement of maintainability success-critical stakeholders in elaborating and evaluating maintenance options.</td>
<td>Critical mass of maintainability SysEs with mission SysE capability, coverage of full M-SysE.skills areas, representation of maintainability success-critical-stakeholder organizations.</td>
<td>Advanced O&amp;M MPT capabilities identified for use: Model-Based SW/SE, TCO analysis support. Basic O&amp;M MPT capabilities for modification, repair and V&amp;V: some initial use.</td>
</tr>
</tbody>
</table>
Future Plans

• Explore applications of SQ tradespace analysis to help ensure balanced solutions to new initiatives in cyber security, autonomy, modular open-systems acquisition, internets of things, learning systems, other Third Offset initiatives

• Develop full-lifecycle set-based design MPTs, analyze areas of requirement uncertainty and evolution; life cycle readiness MPTs; extension of Maintainability data analytics

• Continue satellite cost modeling efforts with DoD and Services centers for cost analysis; industry via INCOSE, MORS, and NDIA

• Continue trial application of MPTs with NSWC Carderock, Army ERDC and TARDEC, USAF ASC and SMC, USMC

Extend transition collaborations with FFRDCs Aerospace, Mitre, SEI
Referents: Stakeholder Priorities
Cost, Schedule, Reliability, Functionality
COCOMO II Model Results

Cost/Schedule/RELY: “pick any two” points

- For 100-KSLOC set of features
- Can “pick all three” with 77-KSLOC set of features
States: Variation by Life Cycle Stage
TRW project defect estimates

Product Integrity
Estimated Defects Remaining

Time, Life Cycle Stage

Unit Test | Integration Test | Accept. Test | Field Test | Actual Usage
Processes: Cost, Speed Variation by Workload Level

Cost to Process Enterprise Workload vs. Response Time

- **Required Architecture:** Custom; many cache processors
- **Original Architecture:** Modified Client-Server

<table>
<thead>
<tr>
<th>Budget</th>
<th>Response Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100M</td>
<td>1</td>
</tr>
<tr>
<td>$50M</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

- **Original Spec**
- **After Prototyping**

5-17-2016
Relations: SW Development Cost vs. Reliability

Quality is Free: Did Crosby Get it Wrong?

<table>
<thead>
<tr>
<th>MTBF (hours)</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.82</td>
<td>0.92</td>
<td>1.0</td>
<td>1.10</td>
<td>1.26</td>
</tr>
</tbody>
</table>
Software Ownership Cost vs. Reliability

Operational-defect cost at Nominal dependability = Software life cycle cost

VL = 2.55
L = 1.52

Operational - defect cost = 0

MTBF (hours) 1 10 300 10,000 300,000

COCOMO II RELY Rating

Relative Cost to Develop, Maintain, Own and Operate