Querying, Qualifying, and Quantifying the Qualities Quagmire

Barry Boehm, USC
Cyber Resilience Summit 7
October 16, 2019
The System Qualities (SQs) quagmire

- Or non-functional requirements; ilities
- Poorly defined, understood, e.g. standards
- Underemphasized in project management
- Major source of project overruns, failures

Key role of Maintainability

- Maintainability opportunities and challenges
- Tools for improving Maintainability

Conclusions
Importance of SQ Tradeoffs

Major source of system overruns, Life cycle costs

- 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 below

Original Spec
Original Architecture: Custom; many cache processors

After Prototyping
Original Cost
$50M
$100M

Response Time (sec)
1 2 3 4 5

Required Architecture: Modified Client-Server
• Engineered Resilient Systems a US DoD priority area in 2012
• Most DoD activity focused on physical systems
  – Field testing, supercomputer modeling, improved vehicle design and experimentation
• DoD SERC tasked to address resilience, tradespace with other SQs for cyber-physical-human systems
  – Vehicles: Robustness, Maneuverability, Speed, Range, Capacity, Usability, Modifiability, Reliability, Availability, Affordability
  – C3I: also Interoperability, Understanding, Agility, Relevance, Speed
• Resilience found to have numerous definitions
  – Wikipedia 2012 proliferation of definitions
  – Weak standards: ISO/IEC 25010: Systems and Software Quality
Proliferation of Definitions: Resilience


- 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
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
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?
Outline

• The System Qualities (SQs) quagmire
  – Or non-functional requirements; ilities
  – Poorly defined, understood, e.g. standards
  – Underemphasized in project management
  – Major source of project overruns, failures

→ Key role of Maintainability
  – Maintainability opportunities and challenges
  – Tools for improving Maintainability

• Conclusions
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
  - 2018: CISQ estimate: 2.8 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
  - Aging legacy systems
Persistence of Legacy Systems

- New life-cycle technology needs to address improvement of aging legacy systems

1939's Science Fiction World of 2000

Actual World of 2000
Software Quality Understanding by Analysis of Abundant Data (SQUAAD)

➢ An automated cloud-based infrastructure to
  ○ Retrieve a subject system’s information from various sources (e.g., commit history and issue repository).
  ○ Distribute hundreds of distinct revisions on multiple cloud instances, compile each revision, and run static/dynamic programming analysis techniques on it.
  ○ Collect and interpret the artifacts generated by programming analysis techniques to extract quality attributes or calculate change.

➢ A set of statistical analysis techniques tailored for understanding software quality evolution.
  ○ Simple statistics, such as frequency of code smell introduction or correlation between two quality attributes.
  ○ Machine learning techniques, such as clustering developers based on their impact.

➢ An extensible web interface to illustrate software evolution.
### A Recent Experiment

#### Metrics

<table>
<thead>
<tr>
<th>Group</th>
<th>Abbr.</th>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>LC</td>
<td>SonarQube</td>
<td>Physical Lines excl. Whitespaces/Comments</td>
</tr>
<tr>
<td></td>
<td>FN</td>
<td>SonarQube</td>
<td>Functions</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>FindBugs</td>
<td>Classes</td>
</tr>
<tr>
<td>Code</td>
<td>CX</td>
<td>SonarQube</td>
<td>Complexity (Number of Paths)</td>
</tr>
<tr>
<td>Quality</td>
<td>SM</td>
<td>SonarQube</td>
<td>Code Smells</td>
</tr>
<tr>
<td>Security</td>
<td>VL</td>
<td>SonarQube</td>
<td>Vulnerabilities</td>
</tr>
<tr>
<td></td>
<td>SG</td>
<td>PMD</td>
<td>Security Guidelines</td>
</tr>
<tr>
<td></td>
<td>FG</td>
<td>FindBugs</td>
<td>Malicious Code, Security</td>
</tr>
</tbody>
</table>

#### Scale

<table>
<thead>
<tr>
<th>Org.</th>
<th>Time Span</th>
<th>Sys.</th>
<th>Dev.</th>
<th>Rev.</th>
<th>MSLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix</td>
<td>09/12-12/17</td>
<td>12</td>
<td>251</td>
<td>3683</td>
<td>34</td>
</tr>
<tr>
<td>Apache</td>
<td>01/02-03/17</td>
<td>39</td>
<td>1102</td>
<td>20197</td>
<td>576</td>
</tr>
<tr>
<td>Google</td>
<td>08/08-01/18</td>
<td>17</td>
<td>402</td>
<td>11354</td>
<td>753</td>
</tr>
<tr>
<td>Total</td>
<td>01/02-01/18</td>
<td>68</td>
<td>1755</td>
<td>35234</td>
<td>1363</td>
</tr>
</tbody>
</table>
Evolution of a Single Quality Attribute

➢ How a single quality attribute evolves.

➢ Two metrics
  o Size (top)
  o Code Smells (bottom)

➢ One project

➢ 9 years
Top-10 Non-Technical Sources of Tech Debt
Based on Workshop participant vote totals

1. Separate organizations and budgets for systems and software acquisition and maintenance (34)
2. Overconcern with the Voice of the Customer (31)
3. The Conspiracy of Optimism (28)
4. Inadequate system engineering resources (21)
5. Hasty contracting focused on fixed operational requirements (21)
6. CAIV-limited system requirements (20)
7. Brittle, point-solution architectures (18)
8. The Vicious Circle (15)
9. Stovepipe systems (12)
10. Over-extreme forms of agile development (10)
2. Overconcern with the Voice of the Customer/User

Bank of America Master Net

- Users
  - Many features
  - Changeable requirements
  - Applications compatibility
  - High levels of service
  - Voice in acquisition
  - Flexible contract
  - Early availability

- Maintainers
  - Ease of transition
  - Ease of maintenance
  - Applications compatibility
  - Voice in acquisition

- Acquirers
  - Mission cost/effectiveness
  - Limited development budget, schedule
  - Government standards compliance
  - Political correctness
  - Development visibility and control
  - Rigorous contact

- Developers
  - Flexible contract
  - Ease of meeting budget and schedule
  - Stable requirements
  - Freedom of choice: process
  - Freedom of choice: team
  - Freedom of choice: COTS/reuse

PC: Process
PD: Product
PP: Property
S: Success
3. The Conspiracy of Optimism
Take the lower branch of the Cone of Uncertainty

![Graph showing the relationship between size (SLOC), cost (S), and relative size range across different phases and milestones.]

F-22
187 A/C
$79B
Aerospace America, 1/2016

F-22
750 A/C
$26B
Example: Reliability Revisited

• Reliability is the probability that the system will deliver stakeholder-satisfactory results for a given time period (generally an hour), given specified ranges of:
  
  – Stakeholders: 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: security thresholds, types of payload/cargo; workload volume, diversity
  
  – Effects of other SQs: synergies, conflicts
Problem and Opportunity (%O&M costs)
Remember Willie Sutton

- US Government IT: ~75%; $59 Billion [GAO 2015]
- Hardware [Redman 2008]
  - 12% -- Missiles (average)
  - 60% -- Ships (average)
  - 78% -- Aircraft (F-16)
  - 84% -- Ground vehicles (Bradley)
- Software [Koskinen 2010]
  - 75-90% -- Business, Command-Control
  - 50-80% -- Complex platforms as above
  - 10-30% -- Simple embedded software
- Primary current emphasis minimizes acquisition costs
  - DoD Better Buying Power memos: Should-Cost
Average Change Processing Time: Two Complex Systems of Systems

Average workdays to process changes

Incompatible with turning within adversary’s OODA loop

Observe, Orient, Decide, Act

10-16-2019 SERC; USC
Maintainability Opportunity Tree: Modifiability

Anticipate Modifiability Needs
- Evolution information
  - Trend analysis
- Hotspot (change source) analysis
- Modifier involvement
- Address Potential Conflicts

Design/Develop for Modifiability
- Modularize around hotspots
- Service-orientation; loose coupling
- Spare capacity; product line engineering
- Domain-specific architecture in domain
- In-flight diagnosis
- Move to Continuous Delivery

Improve Modification V&V
- Prioritize, Schedule Modifications, V&V
- Modification compatibility analysis
- Regression test capabilities
- Value-Based V&V
Investing in Reliability vs. Maintainability

• Baseline: System with 10,000 hours MTBF, 4 days MTTR
  – Availability = 10,000 / (10,000 + 96) = 0.9905

• A. Higher Reliability: 100,000 hour Mean Time Between Failures
  – 4 days Mean Time to Repair
• B. Higher Maintainability: 10,000 hour MTBF
  – 4 hours Mean Time to Repair
  – F-35 Autonomic Logistics information System (ALIS)

• Compare on Availability = MTBF / (MTBF + MTTR)
• A. Availability = 100,000 / (100,000 + 96) = 0.9990
• B. Availability = 10,000 / (10,000 + 4) = 0.9996
7x7 Synergies and Conflicts Matrix

• 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

• Work-in-progress tool will enable clicking on an entry and obtaining details about the synergy or conflict
  – Ideally quantitative; some examples next

• Still need synergies and conflicts within elements
  – Such as Security-Reliability synergies and conflicts
<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>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
<td>Adaptable</td>
</tr>
<tr>
<td>Modularity</td>
<td>Many options</td>
<td>Agile methods</td>
<td>Loose capacity</td>
<td>Spare capacity</td>
<td>Spare capacity</td>
<td>Loose coupling</td>
</tr>
<tr>
<td>Self Adaptive</td>
<td>Service oriented</td>
<td>Loose coupling for sustainability</td>
<td>Product line architectures</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Modularity</td>
</tr>
<tr>
<td>Smart monitoring</td>
<td>User programmability</td>
<td>Total Ownership Cost</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Product line architectures</td>
</tr>
<tr>
<td>Spare Capacity</td>
<td>Versatility</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Service oriented</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>FMEA</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Automated aids</td>
<td>Domain architecting within domain</td>
</tr>
<tr>
<td>Accreditation</td>
<td>Automated I/O validation</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
</tr>
<tr>
<td>Agile methods assurance</td>
<td>Multi-level security</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
</tr>
<tr>
<td>Encryption</td>
<td>Survivability</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
</tr>
<tr>
<td>Many options</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
</tr>
<tr>
<td>Multi-domain modifiability</td>
<td>Versatility</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Multi-level security</td>
<td>Redundancy</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Self Adaptive defects</td>
<td>Scalability</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>User programmability</td>
<td>Spare Capacity</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Usability vs. Security</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Autonomy vs. Usability</th>
<th>Anti-tamper</th>
<th>Automated aids</th>
<th>Automated aids</th>
<th>Automated aids</th>
<th>Automated aids</th>
<th>Automated aids</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity slowdowns</td>
<td>Armor vs. Weight</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
<td>Domain architecting within domain</td>
</tr>
<tr>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Easiest-first development</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
<td>Staffing, Empowering</td>
</tr>
<tr>
<td>Versatility vs. Usability</td>
<td>Redundancy</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Resource Utilization</td>
<td>Scalability</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Spare capacity</td>
<td>Multi-domain architecture interoperability conflicts</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Tight coupling</td>
<td>Usability vs. Cost savings</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
<tr>
<td>Use software vs. hardware</td>
<td>Versatility</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
<td>Value prioritizing</td>
</tr>
</tbody>
</table>

|Multi-domain architecture interoperability conflicts| Lightweight agility| Automated aids| Automated aids| Automated aids|
|Over-optimizing| Multi-domain architecture interoperability conflicts| Automated aids| Automated aids| Automated aids|
|Tight coupling| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|
|Use software vs. hardware| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|

|Agile Methods scalability| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|
|Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing| Over-optimizing|
|Over-optimizing| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|
|Tight coupling| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|
|Use software vs. hardware| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing|

|Interoperability| Encryption interoperability| Multi-domain architecture interoperability conflicts| Over-optimizing| Over-optimizing| Over-optimizing| Reduced speed of Assertion checking|
|Multi-domain architecture interoperability conflicts| Multi-domain architecture interoperability conflicts| Asserting integrity of added components| Over-optimizing| Over-optimizing| Over-optimizing| Reduced speed of connectivity, standards compliance|
|Tight vs. Loose coupling| Multi-domain architecture interoperability conflicts| Tight vs. Loose coupling| Tight vs. Loose coupling| Tight vs. Loose coupling| Tight vs. Loose coupling| Tight vs. Loose coupling|
Software Development Cost vs. Reliability

COCOMO II RELY Rating

<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>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10-16-2019 SERC; USC
Software Ownership Cost vs. Reliability

Relative Cost to Develop, Maintain, Own and Operate

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

COCOMO II RELY Rating

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

VL = 2.55
L = 1.52

Operational - defect cost = 0

70% Maint.

1.07
1.11
1.10
1.10
1.23
1.26
0.92
0.76
0.69
0.82
10-16-2019 SERC; USC
Conclusions

• System qualities (SQs) are success-critical
  – Major source of project overruns, failures
  – Significant source of stakeholder value conflicts
  – Poorly defined, understood
  – Underemphasized in project management

• Need more emphasis on preparing for Maintainability
  – Critical to Resilience and Total Ownership Cost
Backup Charts
## SIS 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>10</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>
<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>4</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>
<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>Mission evolution directions and maintainability implications explored. Some mission use cases defined, some O&amp;M options explored.</td>
<td>Identification of mission-specific maintenance needs, concurrent engrg, O&amp;M integration, Life Cycle cost estimation</td>
<td>Initial exploration of O&amp;M MPT options</td>
</tr>
</tbody>
</table>

10-16-2019 SERC; USC
### SIS Maintainability Readiness Levels 5-7

#### 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>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>
</tbody>
</table>
Agility, Assurance, and Continuous Delivery

Recent SERC Talks; available at https://sercuarc.org/serc-talks/

• Agile Methods for High-Criticality Systems Series
  • Feb. 7, 2018: Jan Bosch, Director Software Center, Chalmers U.
    – Speed, Data and Ecosystems: How to Excel in a Software-Driven World?
  • April 4, 2018: Robin Yeman, Lockheed Martin Fellow
    – How do Agile Methods Reduce Risk Exposure and Improve Security on Highly-Critical Systems?
  • June 6, 2018: Phyllis Marbach, Recent Boeing Agile Lead
    – How Do You Use Agile Methods on Highly-Critical Systems that Require Earned Value Management?

• Systems and Software Qualities Tradespace Analysis Series
  • August 8, 2018: Barry Boehm, USC Prof., SERC Chief Scientist
    – How to Query, Qualify and Quantify the Qualities Quagmire?
  • October 3, 2018: Bill Curtis, Senior VP, CAST; Executive Director, CISQ
    – How Can We Advance Structural Quality Analysis with Standards and Machine Learning?
  • December 11, 2018: Xavier Franch, U. Catalonia Poly, Co-Director, EC Q-Rapids
    – Why Are Ontologies and Languages for Software Quality Increasingly Important?