Systems Qualities Ontology, Tradespace and Affordability

Sponsor: OUSD(R&E) | CCDC

By
Prof. Barry Boehm, USC
11th Annual SERC Sponsor Research Review
November 19, 2019
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW, 8th Floor
Washington, DC 20009

www.sercuarc.org
Outline

Project Origins
- Support of DoD Engineered Resilient Systems Initiative
- Need for and development of Systems Quality Ontology

• Project Teammate Contributions
  - AFIT/NPS
  - GaTech
  - MIT
  - Penn State
  - Wayne State
  - USC/UVa/NPS
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

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
• Oversimplified one-size-fits all definitions

— 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.

  o Satisfying just these will pass “ISO/IEC Reliability,” even if the system fails on rainy-day user stories

  o Surprisingly for a quality standard, it will pass “ISO/IEC Reliability,” even if system fails on satisfying quality requirements

  o Resilience not mentioned

— Need to reflect that different stakeholders rely on different capabilities (functions, performance, flexibility, etc.) at different times and in different environments

— Weak understanding of inter-SQ relationships, e.g. Security
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?
Need for and Nature of SQs Ontology

- Nature of an ontology; choice of IDEF5 structure
- Stakeholder value-based, means-ends hierarchy
- Key role of Maintainability
- Means of clarifying types of Resilience
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, Life Cycle Efficiency, 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
  o Referents: Stakeholder-SQ value-variation (gas mileage vs. size, safety)
  o States: Internal (miles driven); External (off-road, bad weather)
  o Processes: Internal (cost vs. quality); External (haulage, wild driver)
  o Relations: Impact of other SQs (cost vs. weight vs. safety)
Stakeholder value-based, means-ends hierarchy

• Mission operators and managers want improved Mission Effectiveness
  — Involves Physical Capability, Cyber Capability, Human Usability, Speed, Accuracy, Impact, Endurability, Maneuverability, Scalability, Versatility, Interoperability

• Mission investors and system owners want Life Cycle Efficiency
  — Involves Cost, Duration, Personnel, Scarce Quantities (capacity, weight, energy, ...); Manufacturability, Maintainability

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

• In an increasingly dynamic world, all want system Changeability: to be rapidly and cost-effectively changeable
  — Involves Maintainability (Modifiability, Repairability), Adaptability
Dependability, Changeability, and Resilience

- **Reliability**
  - Defect Freedom
  - Survivability
  - Fault Tolerance
  - Complete
  - Partial
    - Robustness
    - Self-Repairability
    - Graceful Degradation
    - Testability, Diagnosability, etc.

- **Availability**
  - Mean Time To Failure
  - Mean Time Between Failures

- **Repairability**
  - Mean Repair Time
  - Mean Time To Repair

- **Maintainability**
  - Test Plans, Coverage
  - Test Scenarios, Data
  - Test Drivers, Oracles
  - Test Software Qualities
  - Testability

- **Resilience**
  - Dependability, Changeability, and Resilience
  - Changeability
  - Adaptability
  - Modifiability
  - Adaptable

- **Means to End (and) Subclass of (or)**
  - A
  - B
  - C, D
Outline

• Project Origins
  — Support of DoD Engineered Resilient Systems Initiative
  — Need for and development of Systems Quality Ontology

→ Project Teammate Contributions
  — AFIT/NPS
  — GaTech
  — MIT
  — Penn State
  — Wayne State
  — USC/UVa/NPS
AFIT-NPS Contributions

• Design of Vehicle Families
  — Associated reuse cost/schedule savings models

Control of swarms of autonomous vehicles; associated flight test at USAF facility

Associated cost tradespace models
• Monterey Phoenix was used to model software system and user behaviors
• The methodology extracts an unadjusted function point (UFP) count from Monterey Phoenix’s executable architecture models for use in software cost estimation
• The COCOMO II model is used to input the UFP count to determine cost estimates
• Allows the assessment of architecture design decisions and their cost impacts
Georgia Tech Contributions

- The Georgia Tech Research Institute (GTRI) began with an existing US Marine Corps design and cost model: the Framework for Assessing Cost and Technology (FACT). They extended the model into a toolset including SysML, the NASA MDAO Framework, Open-source web frameworks, and the MIT Epoch-Era analysis framework, along with collaborating with USC in developing a SysML-based extension of the COSYSMO SysE cost model.

- Identify, develop, and integrate a design space environment and integrated workflow
- Investigate how to operationalize formalisms to measureable and executable constructs
- ... to support Pre-Milestone A Tradespace Analysis

To support investigation and comparison of new methods and constructs for design exploration

To create flexible, rationally guided analyses in a way that allows the customer to quantify and visualize the impact of various requirements.
MIT Contributions

- Extended Ross-Rhodes Changeability Semantic Framework
- Integrated with extended MIT Epoch-Era Analysis Framework
  - Applied to several USAF systems analyses
- Compared with USC ontology framework
Penn State Contributions

- Extended existing design and analysis toolset developed for and used by the Navy to enable addressal of set-based design

![Diagram of set-based design process]

- Revisit a previous model and trade space
- Modify model, input bounds, analyses, etc.
- Develop models
- Generate data
- Visualize data
- Explore sensitivities, find efficient designs, eliminate sets, explore limits, highlight preferences, etc.
Wayne State Contributions

• System Engineering (SE) for Acquisition Qualities and Tradeoffs in Autonomy Enabled Military Systems (AEMS) with Machine Learning (ML) Applications for Manned-Unmanned Teaming (MUM-T)

Set-Based Design in Action 1

• In common practice, ML applications are designed to provide the “maximum likelihood” prediction. A single “best” point solution is selected, and propagated in the chain of ML modules.

• In both classification and regression problems, ML can provide the probability distribution of alternative results. In systems of cascading or composed ML modules, assessment of the distribution of answers may need to be propagated. Some answers may have low probability, but high cost of consequences. In real-time applications, at some point in time an option must be chosen or rejected because it will be too late to execute the action, a fork in the road.

• When one action is chosen, alternatives are eliminated. When time passes and it is too late for an action, that alternative is eliminated. All interpretations are maintained as long as they are possible. At any point in time, the is a best choice or maximum likelihood answer. If external events or new information require immediate action, this will be the choice. Otherwise, the alternative, conflicting, plans or interpretations can be maintained in parallel for multiple hypothesis tracking and delayed differentiation. This is Set-Based Design in action.
Set-Based Design in Action – 2

• (“Multiple hypothesis tracking” and “delayed differentiation” and “Set-Based Design” are essentially synonyms from different academic domains.) The challenge in applying SBD is determining when to eliminate alternative interpretations.

• AEMS can be set up to mechanically execute the actions of a plan. These are simplistic systems. In military applications there are significant unknowns and uncertainties. The AEMS has the alternatives of

• Committing to a course of action based on the maximum likelihood interpretation, more-or-less automatic, mechanical execution of a prior plan with local adaptation to the situation

• Taking actions to obtain information to resolve ambiguity & uncertainty before committing to a course of action (i.e., experimentation or probing)

• Taking the immediate action or interpretation that delays commitment and maximizes the range of future choices, or Deciding that the AEMS is unable to resolve the ambiguity, and asks the handler/commander to decide

• These are the problems for the executive decision-making function, requiring input from the planning function.
Continuing USC/UVa/NPS Research

• Maintainability and Technical Debt
  — Big-Data Analysis: Software Quality Understanding by Analysis of Abundant Data (SQUAAD)
  — Successful Navy and NASA applications
  — Over 1.5 billion lines of open-source code
  — Developing private-cloud version for DoD applications (see poster)

• Velocity
  — Parallel Agile process, code generation

• Cost estimation
  — Working, calibrated COSYSMO 3.0
  — Gathering data to calibrate COCOMO III
  — Extensive survey and insights on cost of improving security (see poster)
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.
  ○ Technical debt, 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

<table>
<thead>
<tr>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Basic</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Code</td>
</tr>
<tr>
<td>Quality</td>
</tr>
<tr>
<td>Security</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></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
  ○ Size (top)
  ○ Code Smells (bottom)
➢ One project
➢ 9 years
Get to market faster without sacrificing quality

- 3 phases: Proof of concept, MVP, Initial Release
  - Each phase approximately a month long
  - Proof of concept uses prototyping to discover requirements, reduce risk
  - MVP uses UML modeling, details sunny/rainy day scenarios, reduce technical debt
  - Initial Release focuses on acceptance testing, performance tuning, optimization, reduce hotfixes
Current status

- 2014-2015 Location Based Advertising (75 students)
  - Implemented commercially; discontinued due to low sales

- 2015 Picture Sharing (12 students)
  - Experiment comparison with Architected Agile project
  - PA project faster, less effort; comparable performance

- 2016-2018 CarmaCam (75 students)
  - In LA-Metro experimental use for bus-lane monitoring
  - Several additional organizations, applications interested

- 2017-2018 TikiMan Go Game project (25 students)
  - Being prepared for commercial application
Database access code doesn’t get written manually

In round numbers this might be 20-40% of your code