SYSTEMS ENGINEERING RESEARCH CENTER

SERC TALKS

WELCOME

How to Query, Qualify and Quantify the Qualities Quagmire?

Barry Boehm - Chief Scientist, SERC; TRW Professor of Software Engineering and Director, Center for Software Engineering, University of Southern California

August 8 | 1:00 PM ET

- Today's session will be recorded.
- An archive of today's talk will be available at: www.sercuarc.org/serc-talks/
- Use the Q&A box to queue questions, reserving the chat box for comments, and questions will be answered during the last 5-10 minutes of the session.
- If you are connected via the dial-in information only, please email questions or comments to SERCtalks@stevens.edu.
- Any issues? Use the chat feature for any technical difficulties or other comments, or email SERCtalks@stevens.edu.

SYSTEMS ENGINEERING RESEARCH CENTER

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense, ASD(R&E), nor the SERC.

No Warranty. This Stevens Institute of Technology Material is furnished on an “as-is” basis. Stevens Institute of Technology makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Stevens Institute of Technology does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

This material has been approved for public release and unlimited distribution.

SYSTEMS ENGINEERING RESEARCH CENTER

Querying, Qualifying, and Quantifying the Qualities Quagmire

Barry Boehm, USC

SERC Talks Presentation

August 8, 2016

SYSTEMS ENGINEERING RESEARCH CENTER

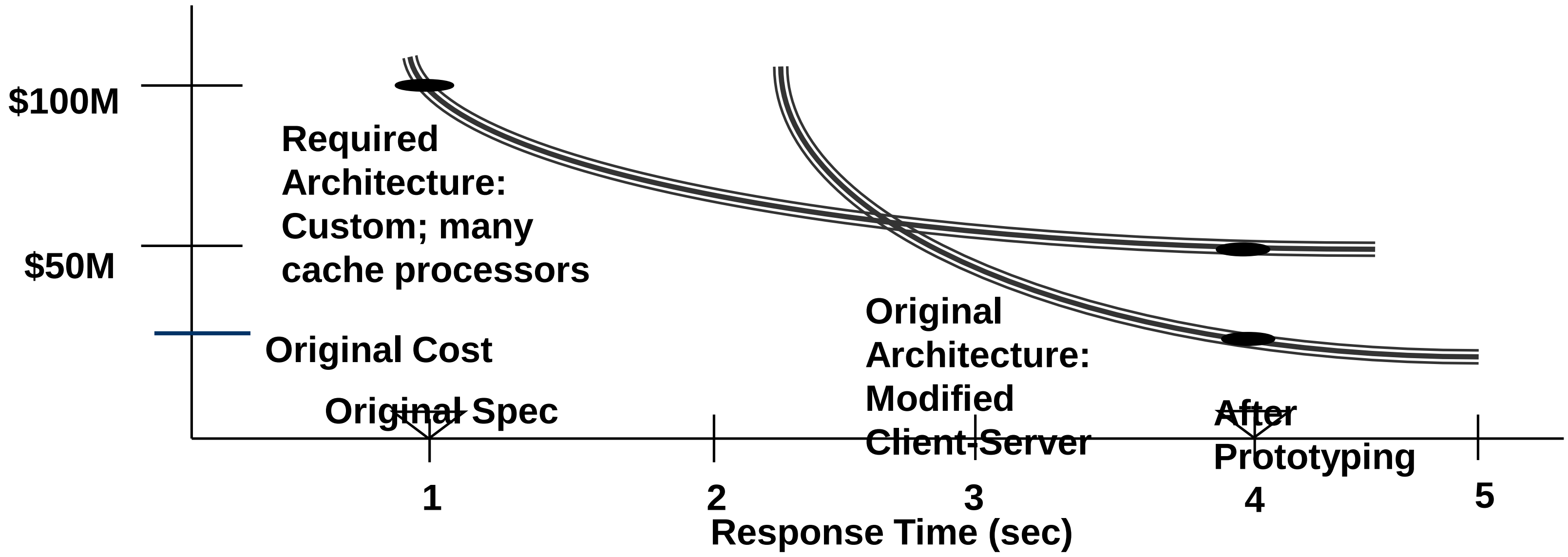
Outline

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



The Quagmire: SQs Ontology origins

- 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
  - DoD Systems Engineering Research Center (SERC) Ontology
    - Fits INCOSE Systems Engineering Handbook Resilience definition

SYSTEMS ENGINEERING RESEARCH CENTER

Proliferation of Definitions: Resilience

- **Wikipedia 2012 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**

Weak standards: ISO/IEC 25010: Systems and Software Quality

- 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.
    - Satisfying just these will pass "ISO/IEC Reliability," even if the system fails on rainy-day user stories
    - Surprisingly for a quality standard, it will pass "ISO/IEC Reliability," even if system fails on satisfying quality requirements
    - 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?

Outline

- The qualities quagmire
  - Or non-functional requirements; ilities
  - Poorly defined, understood, e.g. standards
  - Underemphasized in project management
  - Major source of project overruns, failures
- ➡ 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
- Maintainability opportunities and challenges
- Tools for improving Maintainability
- Conclusions

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
  - Referents: Stakeholder-SQ value-variation (gas mileage vs. size, safety)
  - States: Internal (miles driven); External (off-road, bad weather)
  - Processes: Internal (cost vs. quality); External (haulage, wild driver)
  - Relations: Impact of other SQs (cost vs. weight vs. safety)

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**

SYSTEMS ENGINEERING RESEARCH CENTER

Stakeholder value-based, means-ends hierarchy

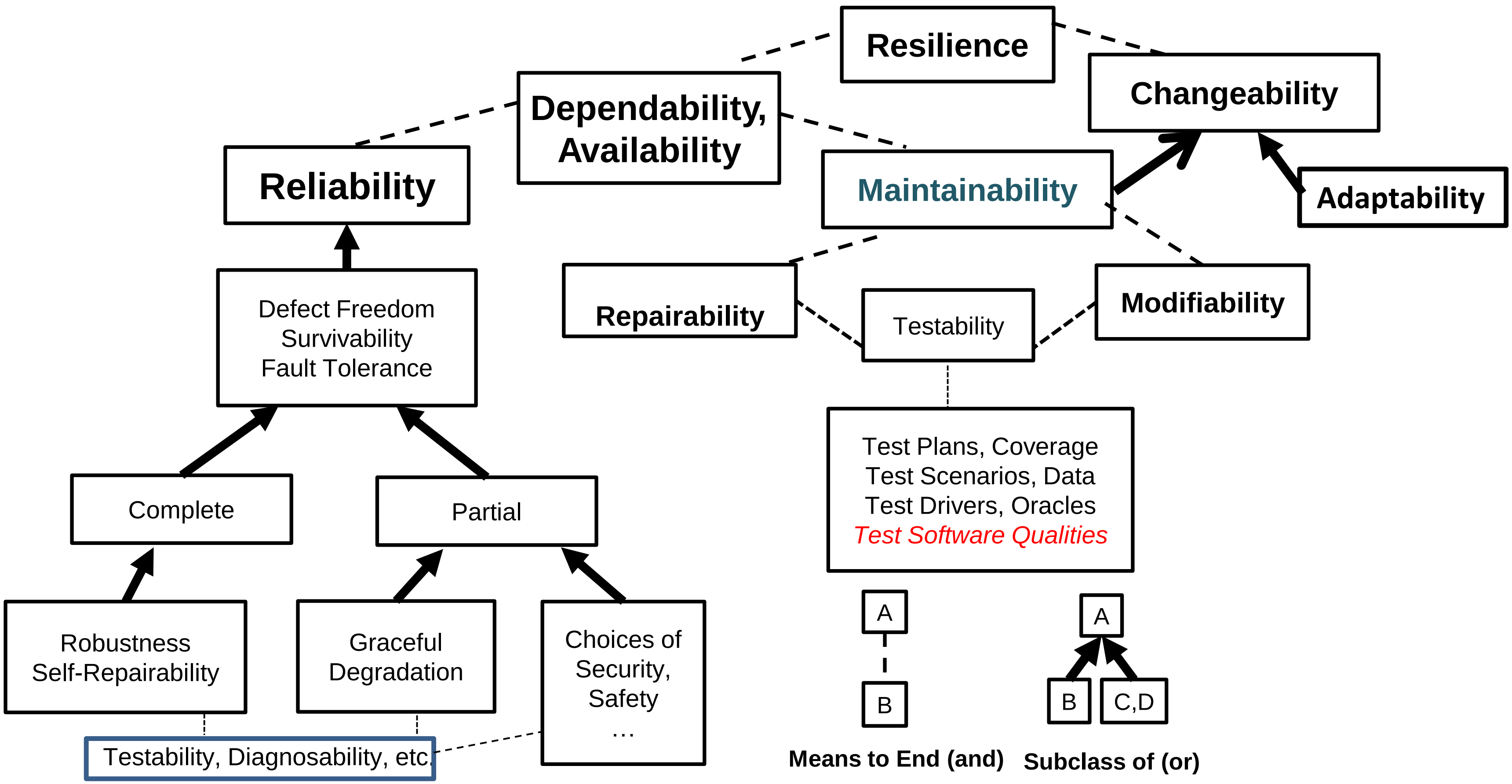
- Mission operators and managers want improved Mission Effectiveness
  - Involves Physical Capability, Cyber Capability, Human Usability, Speed, Accuracy, Impact, Endurability, Maneuverability, Scalability, Versatility, Interoperability

- 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, Availablilty, 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



SYSTEMS ENGINEERING RESEARCH CENTER

How does Resilience depend on Maintainability?

Resilience: INCOSE SysE Handbook

- Resilience is the ability to prepare and plan for, absorb or mitigate, recover from, or more successfully adapt to actual or potential adverse events.
  - Absorb: Robustness (e.g., via armor or redundancy)
  - Mitigate: Graceful Degradation
  - Recover from: Repairability
  - Adapt to actual or potential adverse events:
    - Internally: Self-modifiability
    - Externally: User-modifiability

- Activities in black are performed during Development. Subsequent upgrades are counted as Maintenance activity along with the activities in red.

SYSTEMS ENGINEERING RESEARCH CENTER

Outline

- The qualities quagmire
  - Or non-functional requirements; ilities
  - Poorly defined, understood, e.g. standards
  - Underemphasized in project management
  - Major source of project overruns, failures
- 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
- ➡ Maintainability opportunities and challenges
- Tools for improving Maintainability
- Conclusions

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

Surprise-Free Systems, Software Trends

- More, larger, more complex software and systems
  - Internets of things, more dynamic systems of systems
- Increasing speed of change
- Increasing need for software dependability
  - Safety, security of cyber-physical-human systems
- Increasing software autonomy
  - Principle of Human Primacy in microseconds?
- Increasing data capture, data analytics
- Increasing legacy software, evolution challenges
  - Mounting technical debt

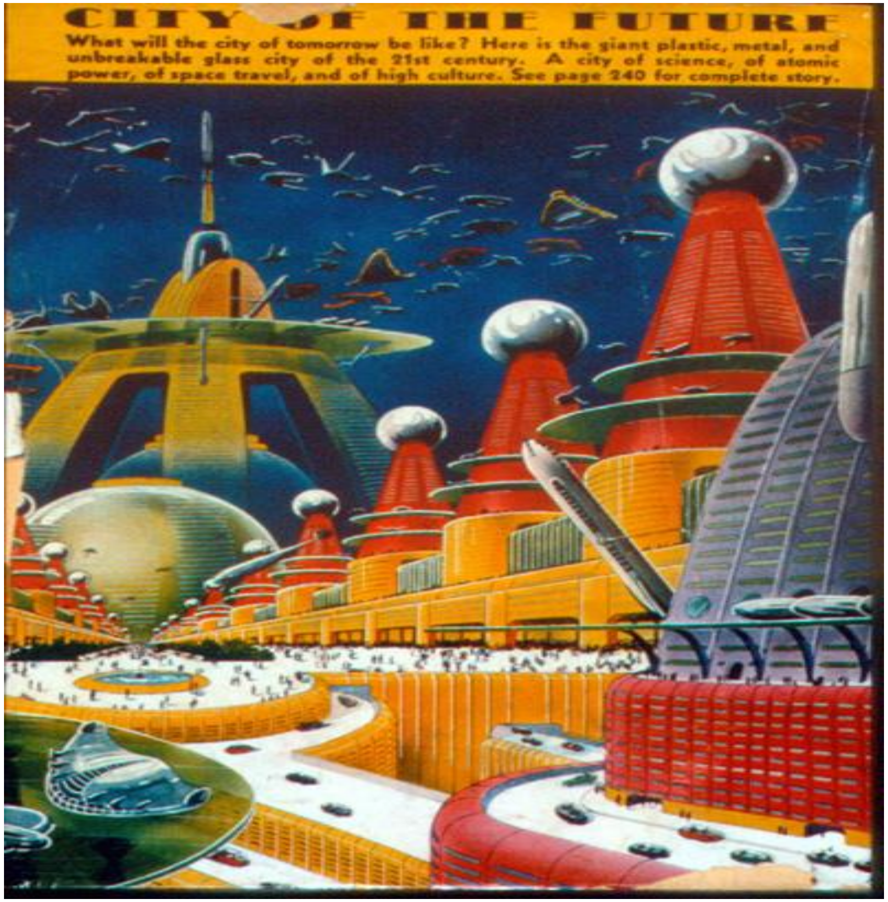
What is Technical Debt (TD)?

- 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
  - Aging legacy systems

Persistence of Legacy Systems

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

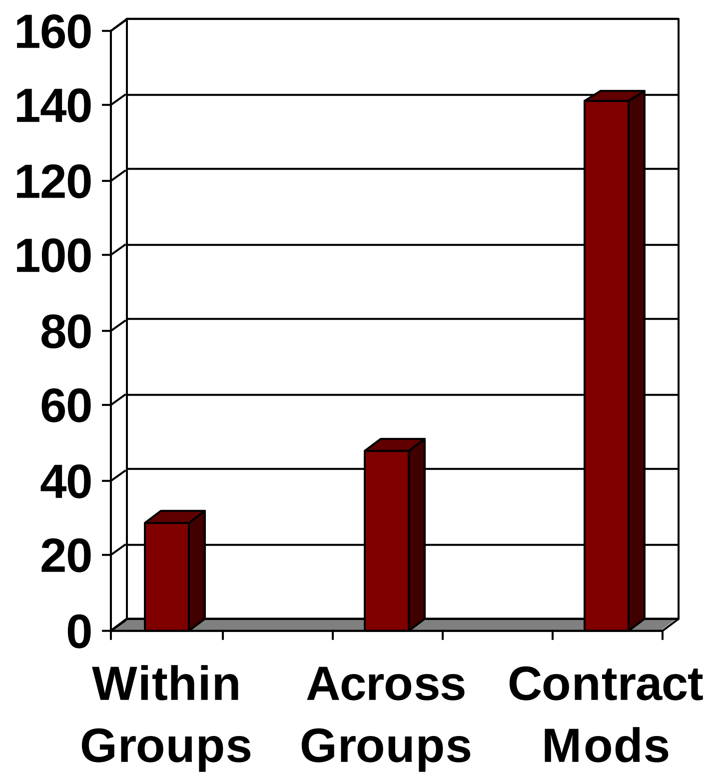
1939's Science Fiction World of 2000



Actual World of 2000

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

Outline

- The qualities quagmire
  - Or non-functional requirements; ilities
  - Poorly defined, understood, e.g. standards
  - Underemphasized in project management
  - Major source of project overruns, failures
- 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
- Maintainability opportunities and challenges
- ➡ Tools for improving Maintainability
- Conclusions

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)

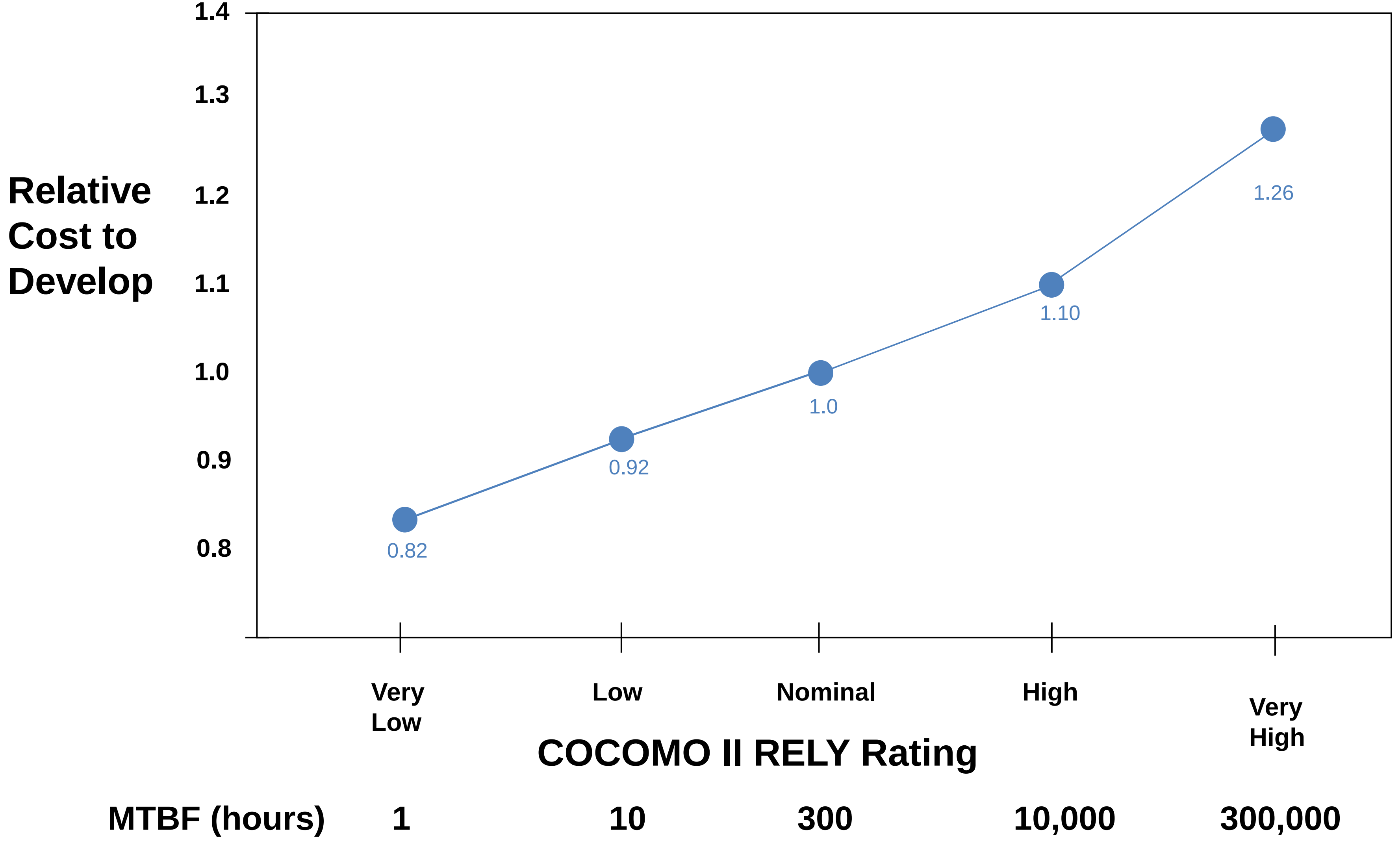
- Compare on Availability = MTBF / (MTBF + MTTR)
- A. Availability = 100,000 / (100,000 + 96) = 0.9990
- B. Availability = 10,000 / (10,000 + 4) = 0.9996

7x7 Synergies and Conflicts Matrix

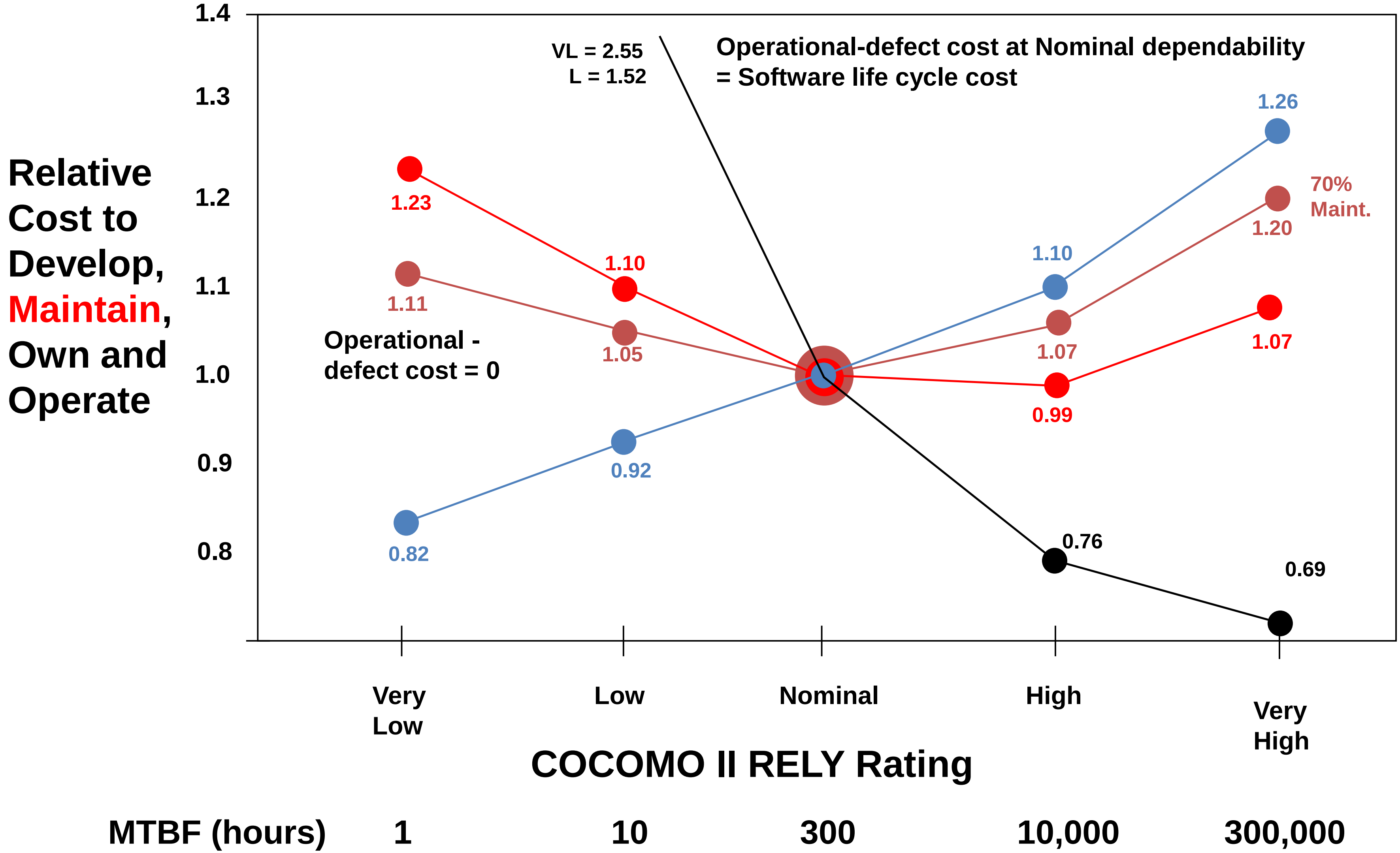
- Mission Effectiveness expanded to 4 elements
  - Physical Capability, Cyber Capability, Interoperability, Other Mission Effectiveness (including Usability as Human Capability)
- Synergies and Conflicts among the 7 resulting elements identified in 7x7 matrix
  - Synergies above main diagonal, Conflicts 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

	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	

Software Development Cost vs. Reliability



Software Ownership Cost vs. Reliability



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

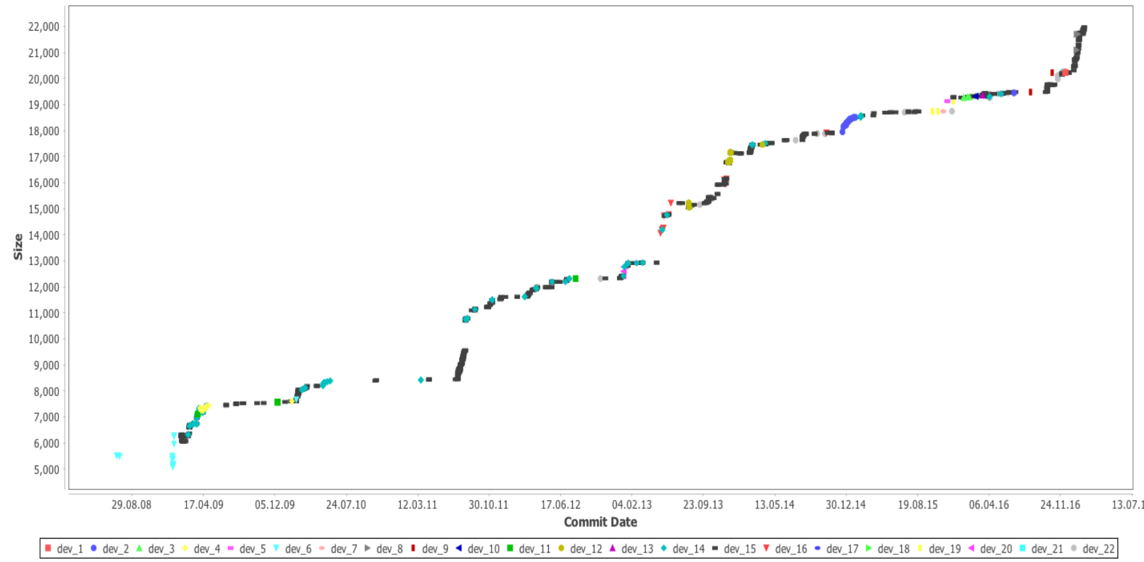
Group	Abbr.	Tool	Description
Basic	LC	SonarQube	Physical Lines excl. Whitespaces/Comments
	FN	SonarQube	Functions
	CS	FindBugs	Classes
Code Quality	CX	SonarQube	Complexity (Number of Paths)
	SM	SonarQube	Code Smells
	PD	PMD	Empty Code, Naming, Braces, Import Statements, Coupling, Unused Code, Unnecessary, Design, Optimization, String and StringBuffer, Code Size
Security	VL	SonarQube	Vulnerabilities
	SG	PMD	Security Guidelines
	FG	FindBugs	Malicious Code, Security

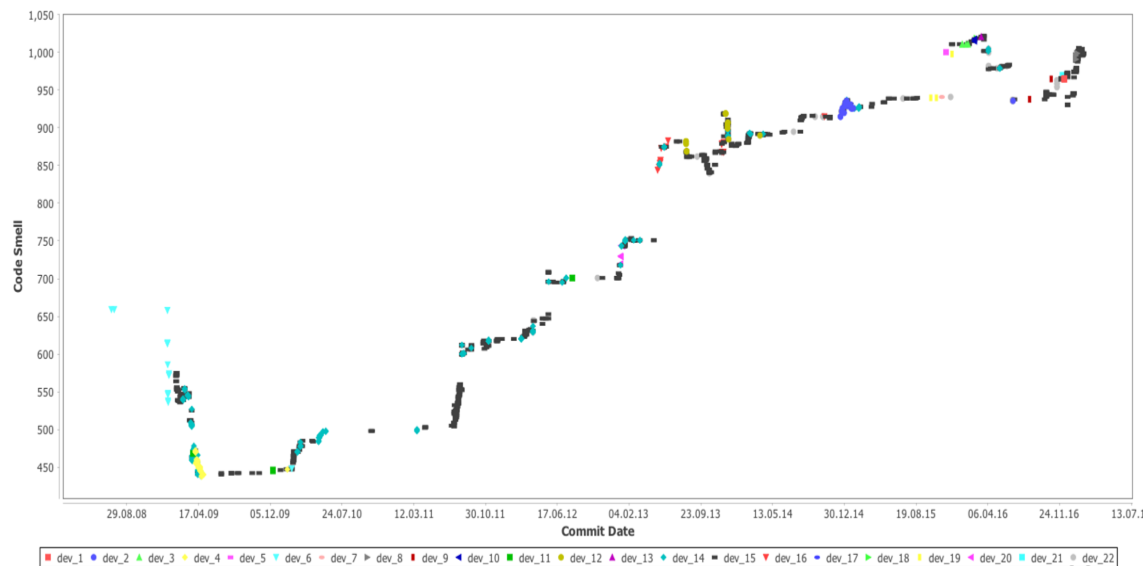
Scale

Org.	Time Span	Sys.	Dev.	Rev.	MSLOC
Netflix	09/12-12/17	12	251	3683	34
Apache	01/02-03/17	39	1102	20197	576
Google	08/08-01/18	17	402	11354	753
Total	01/02-01/18	68	1755	35234	1363

Evolution of a Single Quality Attribute

- How a single quality attribute evolves.
- Two metrics
  - Size (top)
  - Code Smells (bottom)
- One project
- 9 years



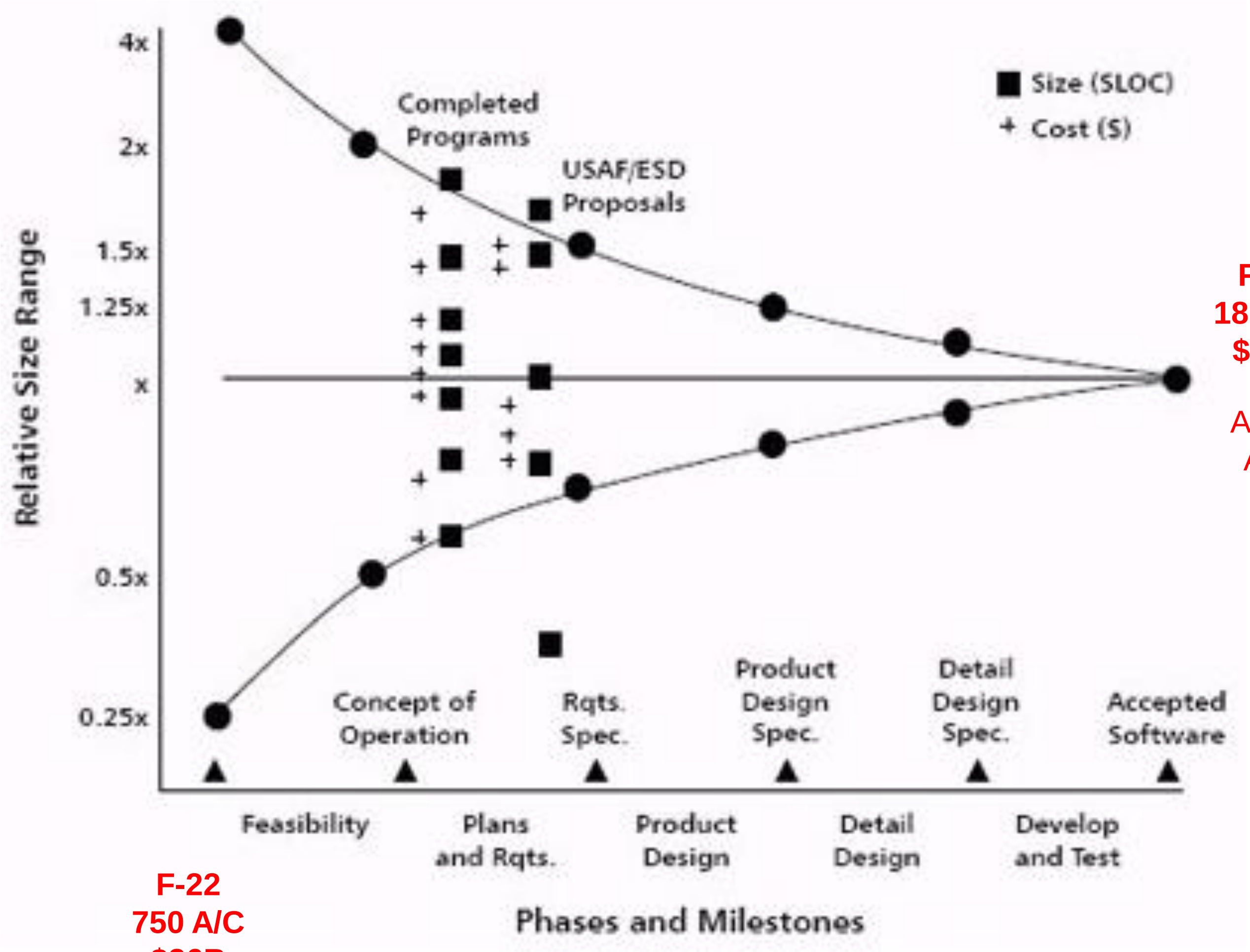


Top-10 Non-Technical Sources of Tech Debt

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

3. The Conspiracy of Optimism

Take the lower branch of the Cone of Uncertainty



F-22
187 A/C
$79B

Aerospace
America,
1/2016

F-22
750 A/C
$26B

SIS Maintainability Readiness Levels

Software-Intensive Systems Maintainability Readiness Levels			
SMR Level	**OpCon, Contracting: Missions, Scenarios, Resources, Incentives**	**Personnel Capabilities and Participation**	**Enabling Methods, Processes, and Tools (MPTs)**
9	5 years of successful maintenance operations,including outcome-based incentives, adaptation to new technologies, missions, and stakeholders	In addition, creating incentives for continuing effective maintainability. performance on long-duration projects	Evidence of improvements in innovative O&M MPTs based on ongoing O&M experience
8	One year of successful maintenance operations, including outcome-based incentives, refinements of OpCon.	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	Evidence of MPT improvements based on ongoing refinement, and extensions of ongoing evaluation, initial O&M MPTs.
7	System passes Maintainability Readiness Review with evidence of viable OpCon, Contracting, Logistics, Resources, Incentives, personnel capabilities, enabling MPTs	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.	Advanced, integrated, tested, and exercised full-LC MBS&SE MPTs and Maintainability-other-SQ tradespace analysis
6	Mostly-elaborated maintainability OpCon. with roles, responsibilities, workflows, logistics management plans with budgets, schedules, resources, staffing, infrastructure and enabling MPT choices, V&V and review procedures.	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&V, identification & reduction of technical debt.	Advanced, integrated, tested full-LC Model-Based Software & Systems (MBS&SE) MPTs and Maintainability-other-SQ tradespace analysis tools identified for use, and being individually used and integrated.
5	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.	In addition, independent maintainability experts participate in project evidence-based decision reviews, identify potential maintainability conflicts with other SQs	Advanced full-lifecycle (full-LC) O&M MPTs and SW/SE MPTs identified for use. Basic MPTs for tradespace analysis among maintainability & other SQs, including TCO being used.
4	Artifacts focused on missions. Primary maintenance options determined, Early involvement of maintainability success-critical stakeholders in elaborating and evaluating maintenance options.	Critical mass of maintainability SysEs with mission SysE capability, coverage of full M-SysE.skills areas, representation of maintainability success-critical-stakeholder organizations.	Advanced O&M MPT capabilities identified for use: Model-Based SW/SE, TCO analysis support. Basic O&M MPT capabilities for modification, repair and V&V: some initial use.
3	Elaboration of mission OpCon, Arch views, lifecycle cost estimation. Key mission, O&M, success-critical stakeholders (SCSHs) identified, some maintainability options explored.	O&M success-critical stakeholders's provide critical mass of maintainability-capable Sys. engrs. Identification of additional. M-critical success-critical stakeholders.	Basic O&M MPT capabilities identified for use, particularly for OpCon, Arch, and Total cost of ownership (TCO) analysis: some initial use.
2	Mission evolution directions and maintainability implications explored. Some mission use cases defined, some O&M options explored.	Highly maintainability-capable SysEs included in Early SysE team.	Initial exploration of O&M MPT options
1	Focus on mission opportunities, needs. Maintainability not yet considered	Awareness of needs for early expertise for maintainability. concurrent engr'g, O&M integration, Life Cycle cost estimation	Focus on O&M MPT options considered

SIS Maintainability Readiness Levels 5-7

Software-Intensive Systems Maintainability Readiness Levels			
SMR Level	OpCon, Contracting: Missions, Scenarios, Resources, Incentives	Personnel Capabilities and Participation	Enabling Methods, Processes, and Tools (MPTs)
7	System passes Maintainability Readiness Review with evidence of viable OpCon, Contracting, Logistics, Resources, Incentives, personnel capabilities, enabling MPTs	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.	Advanced, integrated, tested, and exercised full-LC MBS&SE MPTs and Maintainability-other-SQ tradespace analysis
6	Mostly-elaborated maintainability OpCon. with roles, responsibilities, workflows, logistics management plans with budgets, schedules, resources, staffing, infrastructure and enabling MPT choices, V&V and review procedures.	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&V, identification & reduction of technical debt.	Advanced, integrated, tested full-LC Model-Based Software & Systems (MBS&SE) MPTs and Maintainability-other-SQ tradespace analysis tools identified for use, and being individually used and integrated.
5	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.	In addition, independent maintainability experts participate in project evidence-based decision reviews, identify potential maintainability conflicts with other SQs	Advanced full-lifecycle (full-LC) O&M MPTs and SW/SE MPTs identified for use. Basic MPTs for tradespace analysis among maintainability & other SQs, including TCO being used.

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?

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

- SQs ontology clarifies nature of system qualities
  - Using value-based, means-ends hierarchy
  - Identifies variation types: referents, states, processes, relations
  - Relations enable SQ synergies and conflicts identification

- Need more emphasis on preparing for Maintainability
  - Critical to Dependability, Changeability, and Total Ownership Cost

References

- Alfayez, R., Chen, C., Behnamghader, P., Srisopha, K. and Boehm, B. “An Empirical Study of Technical Debt in Open-Source Software Systems”, CSER 2017
- Behnamghader, P. and Boehm, B., Towards Better Understanding of Software Quality Evolution Through Commit-Impact Analysis, Proceedings, IEEE Software Quality, Reliability and Security Conference (QRS), July 2017.
- Boehm, B.,Kukreja, N. “An Initial Ontology for System Quality Attributes.” In *Proceedings, INCOSE International Symposium,* July 2015.
- B.Boehm, C.Chen, L.Shi, K.Srisopha, “Maintainability Readiness Levels for Software-Intensive Systems,” CSER 2016, March 2016.
- B.Boehm, C.Chen, L.Shi, K.Srisopha, “The Key Roles of Maintainability in an Ontology for System Qualities,” INCOSE International Symposium, July 2016.
- Chen, C., Alfayez, R., Srisopha, K., Shi, L., and Boehm, B. “Evaluating Human-Assessed Software Maintainability Metrics.” In Proceedings, NASAC, Dec. 2016.
- Chen, C, Shi, L., Shoga, M, Wang, Q. and Boehm, B. , How Do Defects Hurt Qualities? An Empirical Study on Characterizing A Software Maintainability Ontology in Open Source Software, Proceedings, IEEE QRS, July 2018.

Upcoming Events

UPCOMING TALKS:

“Systems and Software Qualities Tradespace Analysis” Series

“How Can We Advance Structural Quality Analysis with Standards and Machine Learning?”

Bill Curtis, Senior VP & Chief Scientist, CAST Software; Head of CAST Research Labs, Executive Director, Consortium for IT Software Quality (CISQ)

October 3 | 1:00 PM ET

“Why Are Ontologies and Languages for Software Quality Increasingly Important?”

Xavier Franch, Full Professor, Polytechnic University of Catalonia (BarcelonaTech)

December 11 | 1:00 PM ET

Please visit the [SERC Talks page](#) for more information and updates.

SERC Annual Events | November 7-8

10 YEARS of SERC

Recognizing a decade of contributions to systems engineering research

WEDNESDAY
NOVEMBER
7 2018

SERC DOCTORAL STUDENTS FORUM

Reception to immediately follow at 5pm

Time: 12:00 - 5:00PM

THURSDAY
NOVEMBER
8 2018

SERC SPONSOR RESEARCH REVIEW

Time: 8:00AM - 5:00PM

LOCATION: FHI360 CONFERENCE CENTER
1825 CONNECTICUT AVE NW, 8TH FLOOR,
WASHINGTON, DC 20009

REGISTER TODAY

Events are free to Academia and Government participants; Industry attendees are charged a nominal fee.

sercuarc.org/event/sdsf-2018
sercuarc.org/event/ssrr-2018

Deadline to Register: October 26

We invite you to visit the collections of research reviews online: ***https://sercuarc.org/research-reviews/***

FOR MORE INFORMATION:
Ms. Monica Brito
mbrito@stevens.edu

Thank you for joining us!

Please check back on the SERC website for today's recording and future SERC Talks information!

Subscribe and follow SERC on these channels:

CONTACT

Editor-in-Chief: Dr. Barry Boehm, University of Southern California – boehm@usc.edu

Webinar Coordinator: Ms. Mimi Marcus, Stevens Institute of Technology – SERCtalks@stevens.edu