SYSTEMS ENGINEERING RESEARCH CENTER

USC CSSE

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

SYSTEMS ENGINEERING RESEARCH CENTER

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

System Quality 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**

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**

SYSTEMS ENGINEERING RESEARCH CENTER

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**

SYSTEMS ENGINEERING RESEARCH CENTER

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**

SYSTEMS ENGINEERING RESEARCH CENTER

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

SYSTEMS ENGINEERING RESEARCH CENTER

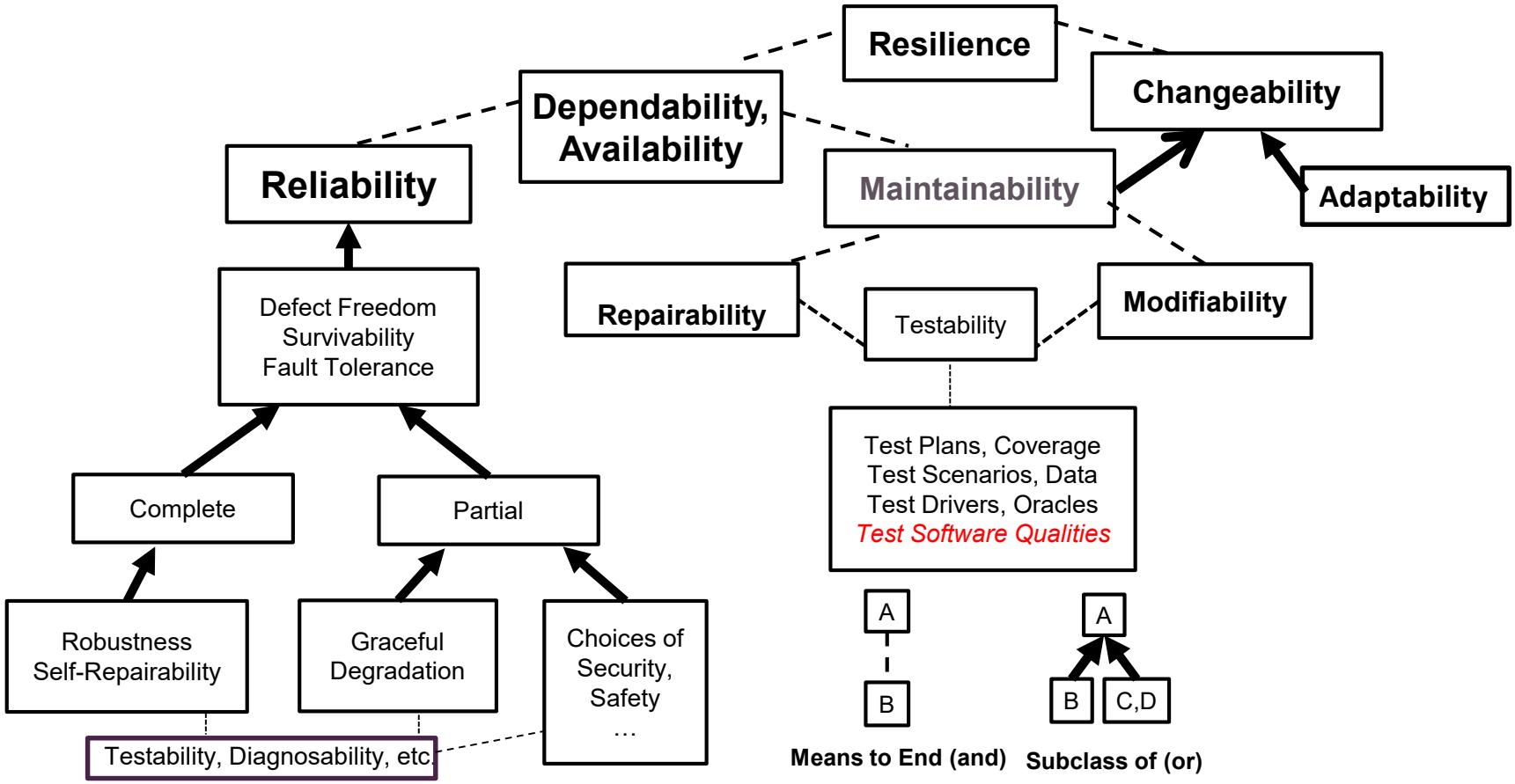
Current SERC SQs Ontology

- **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)

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

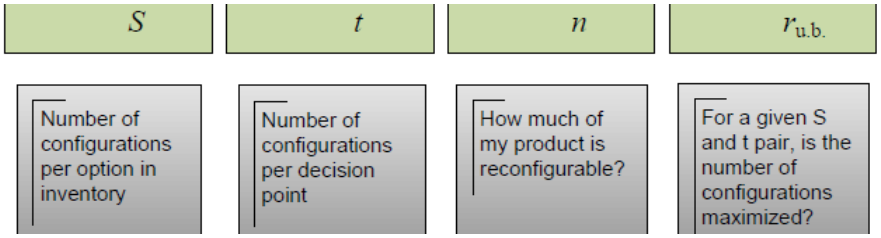


Outline

- **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**



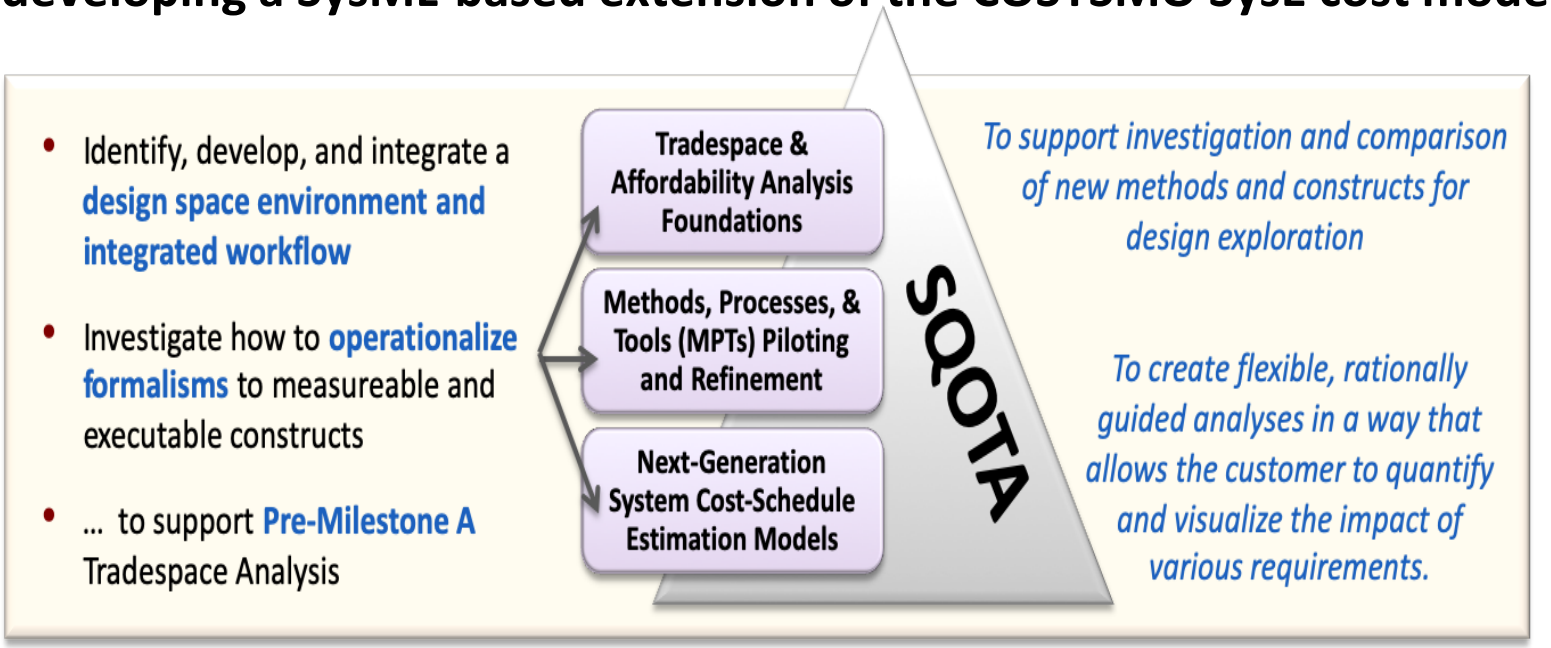
Control of swarms of autonomous vehicles; associated flight test at USAF facility

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.**



SYSTEMS ENGINEERING RESEARCH CENTER

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**

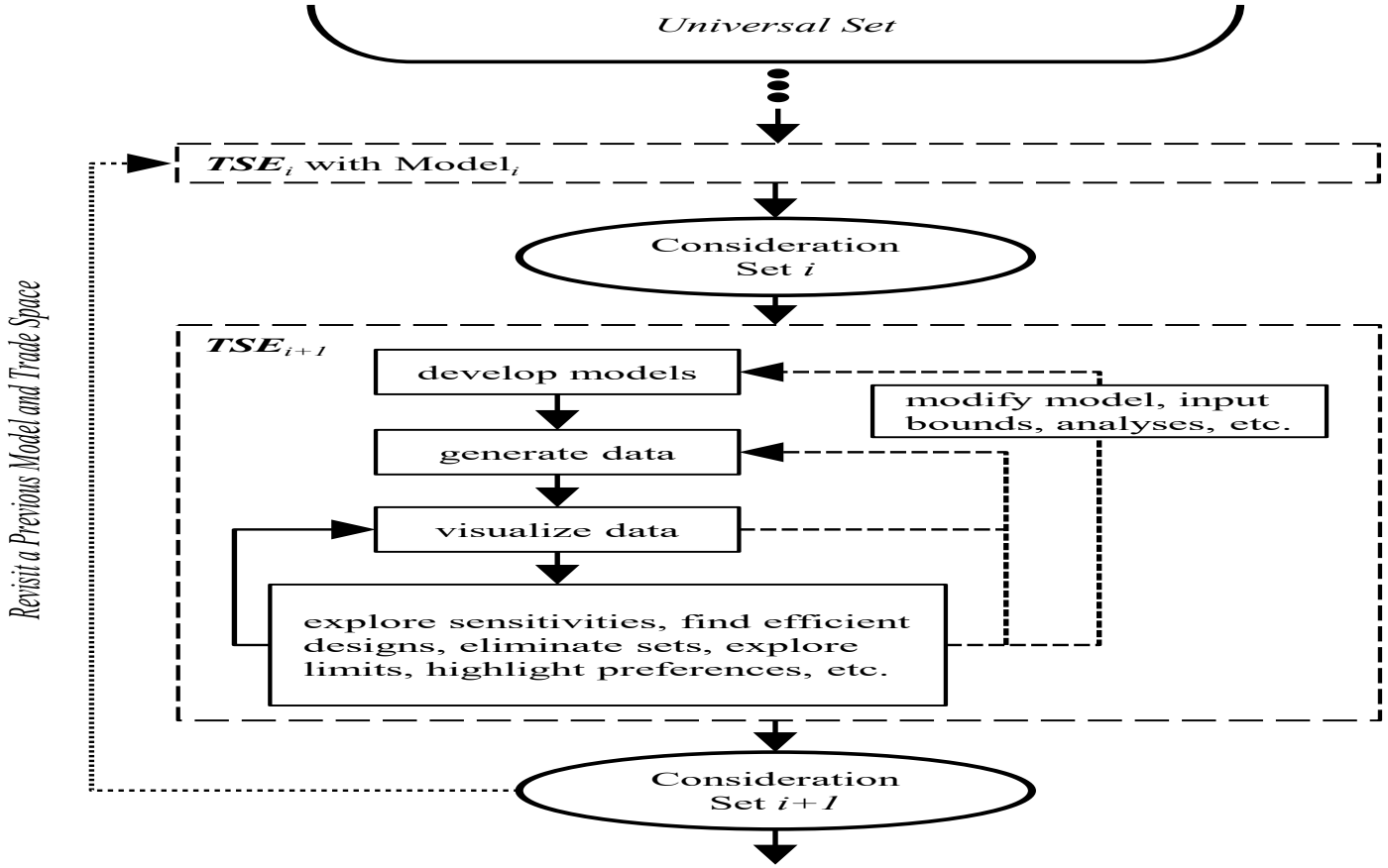
Perturbation	Agent	Impetus		Outcome	
		Nature	Aspect	Effect	Aspect
disturbance	internal	increase	form	same	form
defect	external	decrease	scope	not-same	scope
opportunity	<empty>	re-host	<empty>	reduced	<empty>
<empty>		<empty>		increased	
				<empty>	

USC Ility Label	USC-MIT Differences	MIT Ility Label	Perturbation	Agent	Impetus		Outcome	
					Nature	Aspect	Effect	Aspect
			none	none	decrease	form	decrease	form
			disturbance	internal	same	function	same	function
			shift	external	increase	operations	increase	operations
			<empty>	either	not-same	<empty>	not-same	<empty>
				<empty>	<empty>		<empty>	
Changeability	Match	Changeability		either	not-same		not-same	
Robustness	Shift vs. Disturbance	Robustness	shift				same	
Adaptability	Match	Adaptability		internal	not-same		not-same	
Modifiability	Just External (Internal = Adaptability)	Modifiability			not-same		not-same	
Reconfigurability	Similar	Form Reconfigurability			same	form	not-same	form
Extensibility	Same - Also Scalability-Up	Extensibility		either	not-same		increase	
Contractability	Not included - Also Scalability-Down	Scalability			not-same		not-same	
Versatility	Similar	Functional Versatility			same	form/ops	not-same	function
Survivability	Robustness + Adaptability?	Survivability	disturbance				same	
Evolvability	Same as Changeability?	Evolvability	shift		not-same		not-same	
Flexibility	External Adaptability?	Flexibility		external	not-same		not-same	
Agility	Focus on fix speed	Agility			not-same		not-same	
Reactivity	Focus on duration of fix needs	Reactivity			not-same		not-same	
Resilience	Not specified	Operational Reconfigurability			same	operations	not-same	operations
Fault-Tolerance	Not specified	Operational Versatility			same	form/fnct	not-same	operations
Self-repairability	Not specified	Substitutability			same	fnct/ops	not-same	form
Repairability	Not specified	Value Robustness	shift				same	
Maintainability	Not specified	Value Survivability	disturbance				same	
Graceful Degredation	Not specified	Active Robustness	shift		not-same		same	
Portability	Not specified	Passive Robustness	shift		same		same	
Exchangeability	Like Modifiability?	Classical Passive Robustness	shift	none	same		same	form

Perturbation	Agent	Impetus		Outcome	
		Nature	Aspect	Effect	Aspect
disturbance	internal	increase	form	same	form
disturbnce	internal			same	
disturb, opp'y	internal			not-same	
disturb, opp'y	external			not-same	
			form	not-same	form
disturb, opp'y		increase	scope	not-same	scope
disturb, opp'y		decrease	scope	not-same	scope
disturb, opp'y			scope	same	scope
disturbance		increase		same, increased	
defect	internal			same	
defect	internal			not-same	
defect	external				
	external				
	internal			reduced	
disturb, opp'y		re-host	form	same	form

Penn State Contributions

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



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.

SYSTEMS ENGINEERING RESEARCH CENTER

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.

SYSTEMS ENGINEERING RESEARCH CENTER

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)**

SYSTEMS ENGINEERING RESEARCH CENTER

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.**

SYSTEMS ENGINEERING RESEARCH CENTER

A Recent Experiment

Metrics

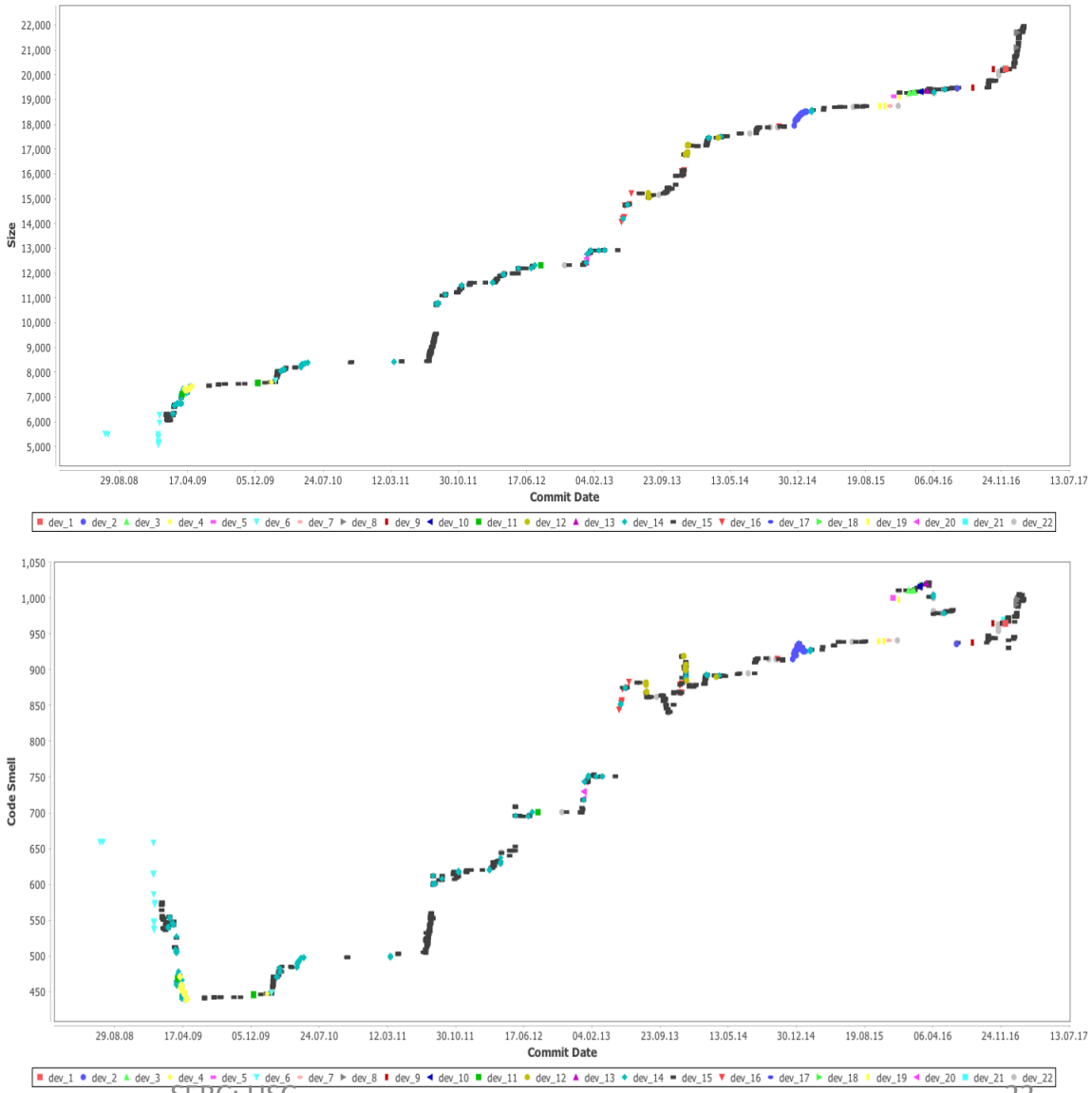
Group	Abbr.	Tool	Description
Basic	LC	SonarQube	Physical Lines excl. Whitespaces/Comments
	FN	SonarQube	Functions
	CS	FindBugs	Classes
Code Quality	CX	SonarQube	Complexity (Number of Paths)
	SM	SonarQube	Code Smells
	PD	PMD	Empty Code, Naming, Braces, Import Statements, Coupling, Unused Code, Unnecessary, 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

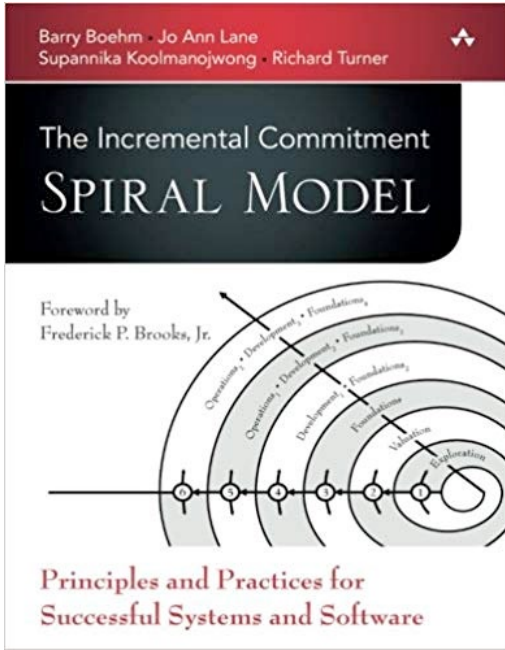
- How a single quality attribute evolves.
- Two metrics
  - Size (top)
  - Code Smells (bottom)
- One project
- 9 years



PARALLEL AGILE

Get to market faster without sacrificing qu

- 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



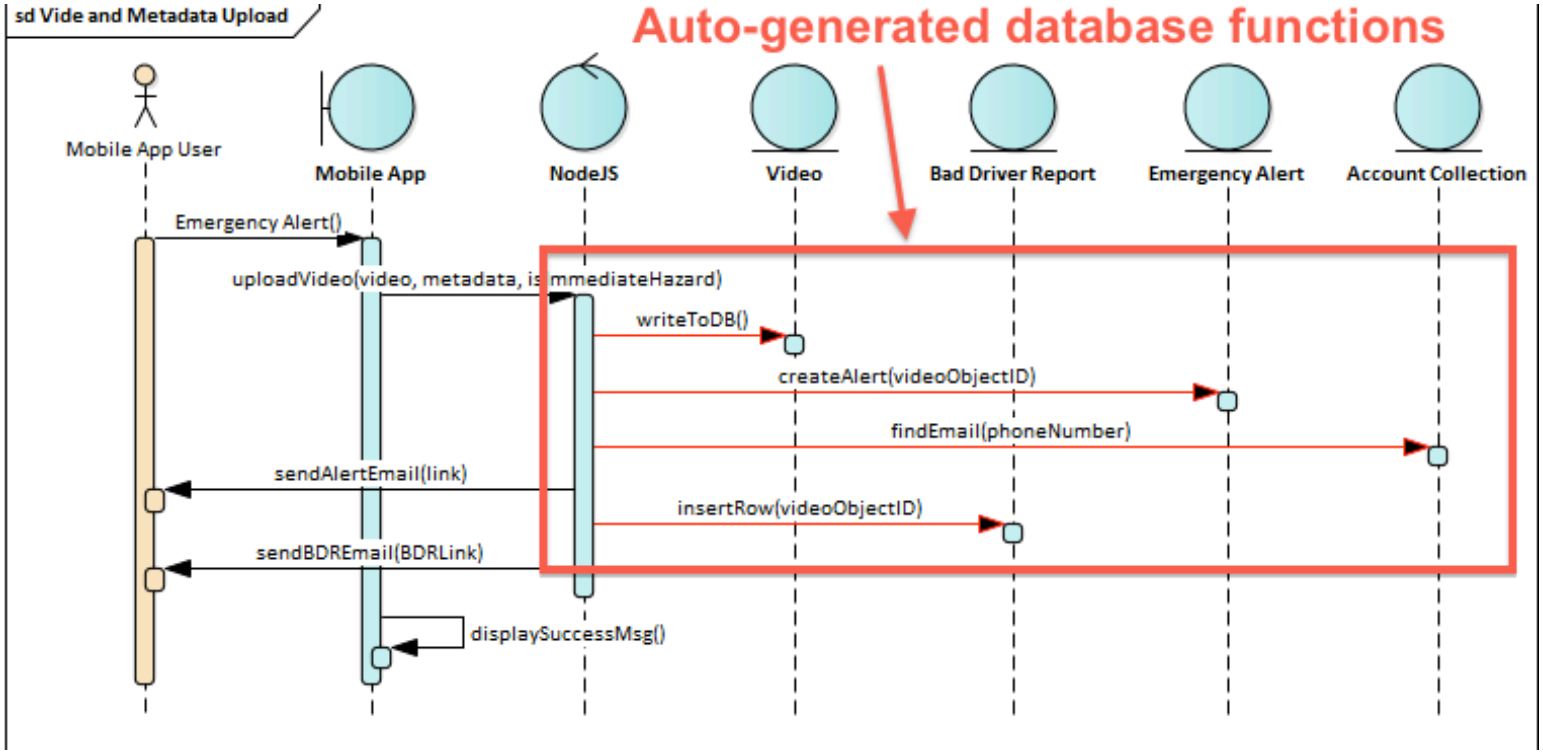
PARALLEL AGILE

—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

PARALLEL AGILE

Database access code doesn't get written manually



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