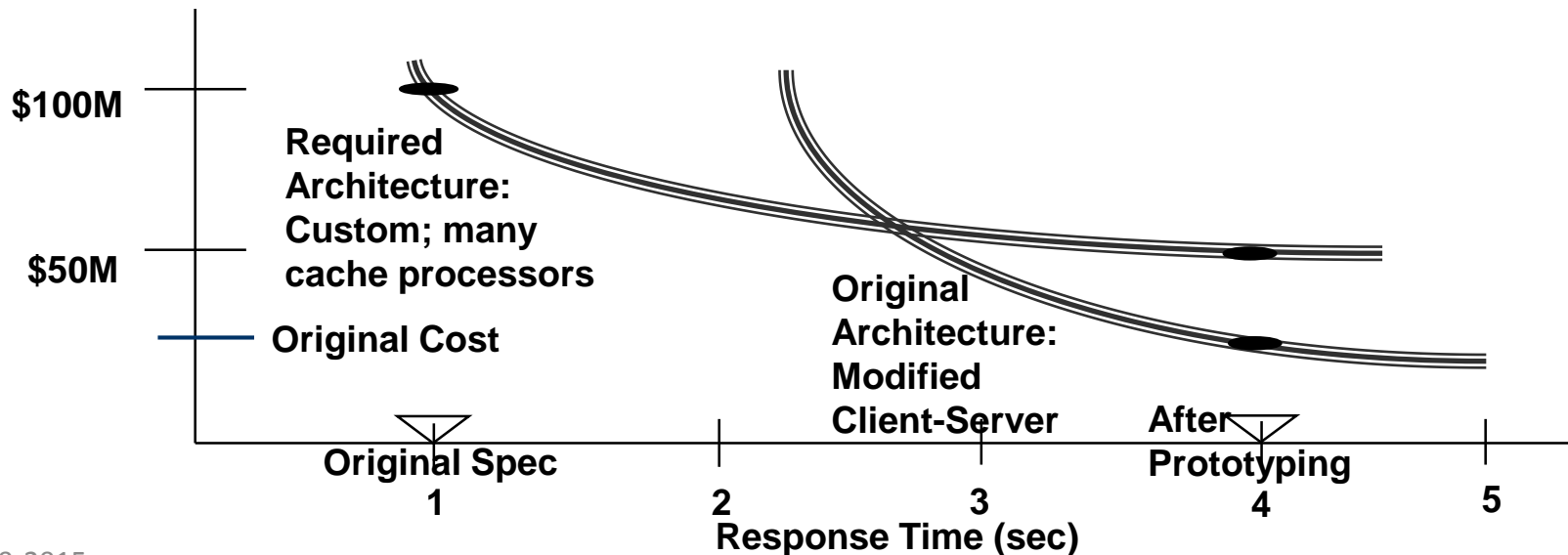


Message: Criticality of Software Qualities (SQs)

Boehm SAM 2015: Major source of system overruns

- SQs have systemwide impact
 - System elements generally just have local impact
- SQs often exhibit asymptotic behavior
 - Watch out for the knee of the curve
- Best architecture is a discontinuous function of SQ level
 - “Build it quickly, tune or fix it later” highly risky
 - Large system example below



Issue: Need for SQs Ontology

- **Oversimplified one-size-fits all definitions**
 - **ISO/IEC 25010, Reliability:** the degree to which a system , product, or component performs specified functions under specified conditions for a specified period of time
 - **OK if specifications are precise, but increasingly “specified conditions” are informal, sunny-day user stories.**
 - **Satisfying just these will pass “ISO/IEC Reliability,” even if system fails on rainy-day user stories**
 - **Need to reflect that different stakeholders rely on different capabilities (functions, performance, flexibility, etc.) at different times and in different environments**
- **Proliferation of definitions, as with Resilience**
- **Weak understanding of inter-SQ relationships**
 - **Reliability Synergies and Conflicts with other qualities**

Question: SQ Ontology KnowledgeBase

- **Modified version of IDEF5 ontology framework**
 - Classes, Subclasses, and Individuals
 - Referents, States, Processes, and Relations
- **Top classes cover stakeholder value propositions**
 - Mission Effectiveness, Resource Utilization, Dependability, Flexibility
- **Subclasses identify means for achieving higher-class ends**
 - Means-ends one-to-many for top classes
 - Ideally mutually exclusive and exhaustive, but some exceptions
 - Many-to-many for lower-level subclasses
- **Referents, States, Processes, Relations cover SQ variation**
 - Referents: Sources of variation by context: Product Q.; Q. In Use
 - States: Internal (beta-test); External (rural, temperate, sunny)
 - Processes: Operational scenarios (normal vs. crisis; experts vs. novices)
 - Relations: Impact of other SQs (synergies & conflicts)

	Flexibility	Dependability	Mission Effectivenss	Resource Utilization	Physical Capability	Cyber Capability	Interoperability
Flexibility		Domain architecting within domain	Adaptability	Adaptability	Adaptability	Adaptability	Adaptability
		Modularity	Many options	Agile methods	Spare capacity	Spare capacity	Loose coupling
		Self Adaptive	Service oriented	Automated I/O validation			Modularity
		Smart monitoring	Spare capacity	Loose coupling for sustainability			Product line architectures
		Spare Capacity	User programmability	Product line architectures			Service-oriented connectors
		Use software vs. hardware	Versatility	Staffing, Empowering			Use software vs. Hardware
Dependability	Accreditation		Accreditation	Automated aids	Fallbacks	Fallbacks	Assertion Checking
	Agile methods assurance		FMEA	Automated I/O validation	Lightweight agility	Redundancy	Domain architecting within domain
	Encryption		Multi-level security	Domain architecting within domain	Redundancy	Value prioritizing	Service oriented
	Many options		Survivability	Product line architectures	Spare capacity		
	Multi-domain modifiability		Spare capacity	Staffing, Empowering	Value prioritizing		
	Multi-level security			Total Ownership Cost			
	Self Adaptive defects			Value prioritizing			
	User programmability						
Mission Effectivenss	Autonomy vs. Usability	Anti-tamper		Automated aids	Automated aids	Automated aids	Automated aids
	Modularity slowdowns	Armor vs. Weight		Domain architecting within domain	Domain architecting within domain	Domain architecting within domain	Domain architecting within domain
	Multi-domain architecture interoperability conflicts	Easiest-first development		Staffing, Empowering	Staffing, Empowering	Staffing, Empowering	Staffing, Empowering
	Versatility vs. Usability	Redundancy		Value prioritizing	Value prioritizing	Value prioritizing	
		Scalability					
		Spare Capacity					
Resource Utilization		Usability vs. Security					
	Agile Methods scalability	Accreditation	Agile methods scalability		Automated aids	Automated aids	Automated aids
	Assertion checking overhead	Acquisition Cost	Cost of automated aids		Domain architecting within domain	Domain architecting within domain	Domain architecting within domain
	Fixed cost contracts	Certification	Many options		Staffing, Empowering	Staffing, Empowering	Rework cost savings
	Modularity	Easiest-first development	Multi-domain architecture interoperability conflicts		Value prioritizing	Value prioritizing	Staffing, Empowering
	Multi-domain architecture interoperability conflicts	Fallbacks	Spare capacity				
	Spare capacity	Multi-domain architecture interoperability conflicts	Usability vs. Cost savings				
	Tight coupling	Redundancy	Versatility				
Physical Capability	Use software vs. hardware	Spare Capacity, tools costs					
		Usability vs. Cost savings					
	Multi-domain architecture interoperability conflicts	Lightweight agility	Multi-domain architecture interoperability conflicts	Cost of automated aids		Automated aids	Automated aids
	Over-optimizing	Multi-domain architecture interoperability conflicts	Over-optimizing	Multi-domain architecture interoperability conflicts		Staffing, Empowering	Domain architecting within domain
Cyber Capability	Tight coupling	Over-optimizing		Over-optimizing		Value prioritizing	
	Use software vs. hardware						
	Agile Methods scalability	Multi-domain architecture interoperability conflicts	Multi-domain architecture interoperability conflicts	Cost of automated aids	Over-optimizing		Automated aids
	Multi-domain architecture interoperability conflicts	Over-optimizing	Over-optimizing	Multi-domain architecture interoperability conflicts	Physical architecture or cyber architecture		Domain architecting within domain
	Over-optimizing			Over-optimizing			
Interoperability	Tight coupling						
	Use software vs. hardware						
	Multi-domain architecture interoperability conflicts	Encryption interoperability	Multi-domain architecture interoperability conflicts	Assertion checking	Over-optimizing	Reduced speed of Assertion checking	4
4-9-2015	User-programmed interoperability	Multi-domain architecture interoperability conflicts		Cost, duration of added connectors	Tight vs. Loose coupling	Reduced speed of connectors, standards compliance	
						Tight vs. Loose coupling	

Software Ownership Cost vs. Reliability

Relative
Cost to
Develop,
Maintain,
Own and
Operate

