The Economics of Software Reliability

Barry Boehm, USC

ISSRE 2003 Keynote Address

November 19, 2003

(boehm@sunset.usc.edu)

(http://sunset.usc.edu)

Outline

- **The business case for software reliability**
  - **Example: net value of test aids**
  - **Value depends on stakeholder value propositions**
- **What are stakeholders really relying on?**
  - **Safety, accuracy, response time, …**
  - **Attributes often conflict**
- **Software dependability in a competitive world**
- **Conclusions**

Software Reliability Business Case

- Software reliability investments compete for resources
  - With investments in functionality, response time, adaptability, speed of development, …
- Each investment option generates curves of net value and ROI
  - Net Value NV = PV(benefit flows-cost flows)
  - Present Value PV = Flows at future times discounted
  - Return on Investment ROI = NV/PV(cost flows)
- Value of benefits varies by stakeholder and role

University of Southern California
Center for Software Engineering

Software Testing Business Case

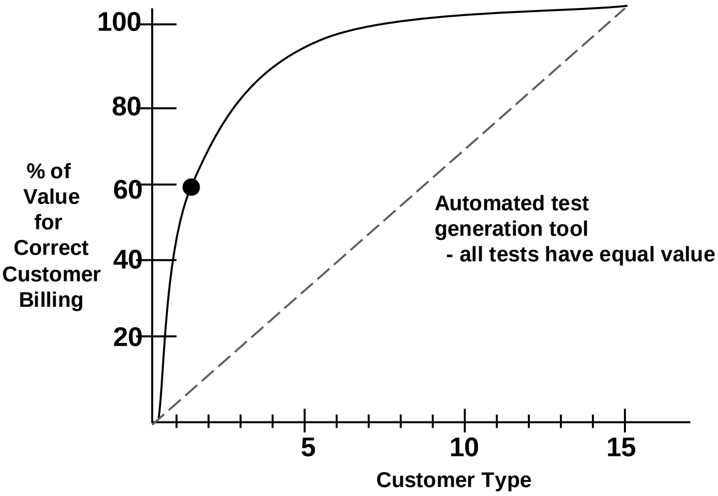
- Vendor proposition
  - Our test data generator will cut your test costs in half
  - We'll provide it to you for 30% of your test costs
  - After you run all your tests for 50% of your original costs, you're 20% ahead
- Any concerns with vendor proposition?

Software Testing Business Case

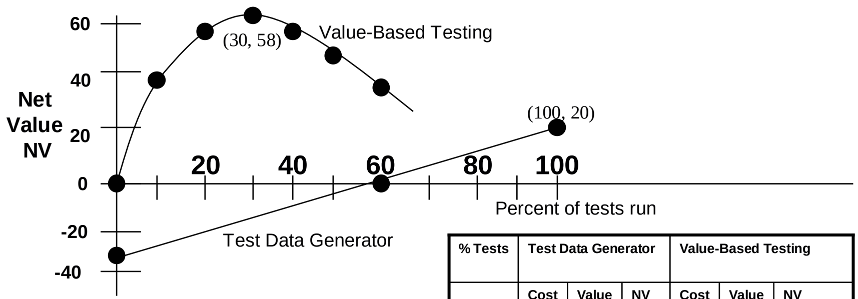
- Vendor proposition
  - Our test data generator will cut your test costs in half
  - We'll provide it to you for 30% of your test costs
  - After you run all your tests for 50% of your original costs, you're 20% ahead
- Any concerns with vendor proposition?
  - Test data generator is value-neutral*
  - Every test case, defect is equally important
  - Usually, 20% of test cases cover 80% of business value

* As are most current software engineering techniques

20% of Features Provide 80% of Value: Focus Testing on These (Bullock, 2000)



Value-Based Testing Provides More Net Value

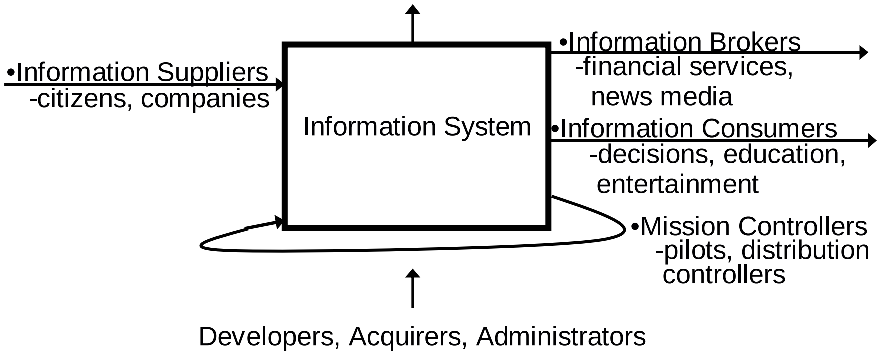


% Tests	Test Data Generator			Value-Based Testing		
	Cost	Value	NV	Cost	Value	NV
0	30	0	-30	0	0	0
10	35	10	-25	10	50	40
20	40	20	-20	20	75	55
30	45	30	-15	30	88	58
40	50	40	-10	40	94	54
….	….	….	….	….	….	….
100	80	100	+20	100	100	0

There is No Universal Dependability-Value Metric

- Different stakeholders rely on different value attributes
  - Protection: safety, security, privacy
  - Robustness: reliability, availability, survivability
  - Quality of Service: performance, accuracy, ease of use
  - Adaptability: evolvability, interoperability
  - Affordability: cost, schedule, reusability
- Value attributes continue to tier down
  - Performance: response time, resource consumption (CPU, memory, comm.)
- Value attributes are scenario-dependent
  - 5 seconds normal response time; 2 seconds in crisis
- Value attributes often conflict
  - Most often with performance and affordability

Major Information System Dependability Stakeholders



Overview of Stakeholder/Value Dependencies

- **Strength of direct dependency on value attribute**

  **- Critical ; *-Significant; blank-insignificant or indirect

Stakeholders \ Attributes	Protection	Robustness	Quality of Service	Adaptability	Affordability	
Info. Suppliers, Dependents	**	*	*			
Info. Brokers	**	**	**	**	*	
Info. Consumers		*	**	*	*	
Mission Controllers, Administrators	*	**	**	**		
Developers, Acquirers		*	*	**	**	

Elaboration of Stakeholder/Value Dependencies

D-Attributes \ Stakeholders	Info Suppliers	Dependents	Info Brokers	Consumers	Info	Mission Controllers	Developers, Maintenance	Administrators	Acquirers
				Mission -					
Protection				crit	uncrit				
Safety		**							
Security	*	**	**						
Privacy	**		*						
Robustross									
Reliability			**	**		**		**	
Availability	*		**	**		**	**	**	
Survivability		**	**	**		**			
Quality of Service									
Performance			**	**	**	**		*-**	
Accuracy, Consistency	**		**	**	*	**		*	
Accessibility, ease of use;	**		**	**	**	**		**	
difficulty of misuse									
Evolvability			**	**	*	*	**	**	**
Interoperability			**					**	**
Cost							*		**
Schedule			*				**		**
Reusability							**	*	*

Implications for Dependability Engineering

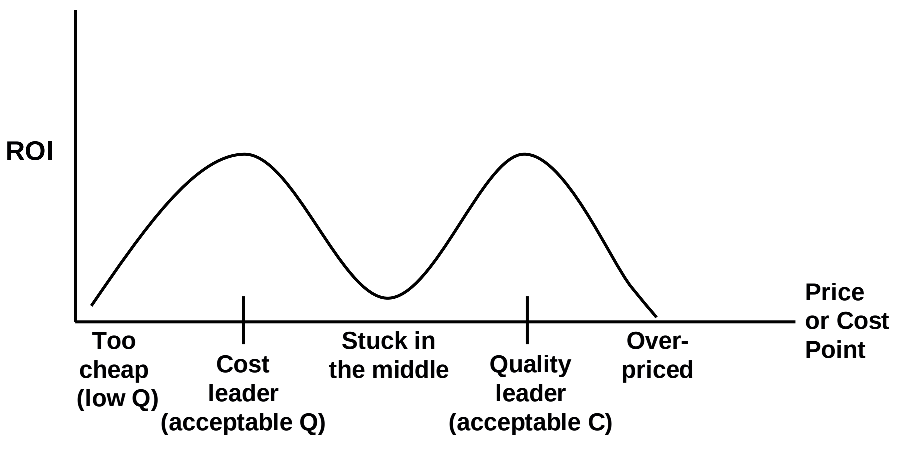
- **There is no universal dependability metric to optimize**
- **Need to identify system's success-critical stakeholders**
  - **And their dependability priorities**
- **Need to balance satisfaction of stakeholder dependencies**
  - **Stakeholder win-win negotiation**
  - **Dependability attribute tradeoff analysis**
- **Need value-of-dependability models, methods, and tools**

Outline

- The business case for software reliability
- What are stakeholders really relying on?
- Software dependability in a competitive world
  - Is market success a monotone function of dependability?
  - Is quality really free?
  - Is “faster, better, cheaper” really achievable?
  - Value-based vs. value-neutral methods
- Conclusions

Competing on Cost and Quality

- adapted from Michael Porter, Harvard

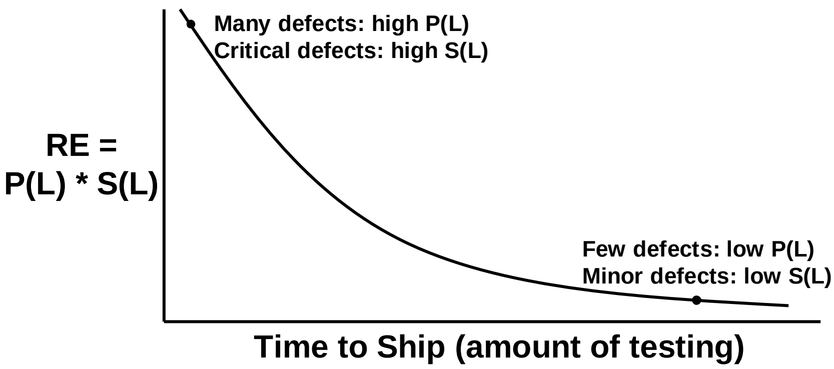


Competing on Schedule and Quality
- A risk analysis approach

- **Risk Exposure RE = Prob (Loss) * Size (Loss)**
  - **"Loss" – financial; reputation; future prospects, …**
- **For multiple sources of loss:**

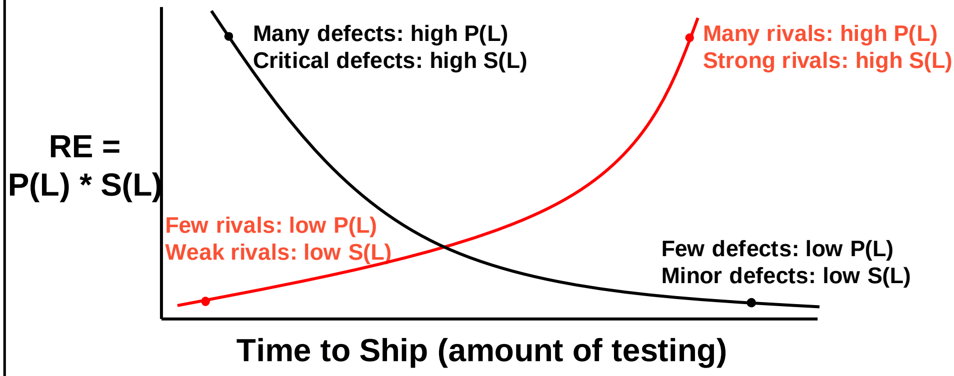
$$RE = \sum_{sources} [\text{Prob (Loss)} * \text{Size (Loss)}]_{source}$$

Example RE Profile: Time to Ship
- Loss due to unacceptable dependability



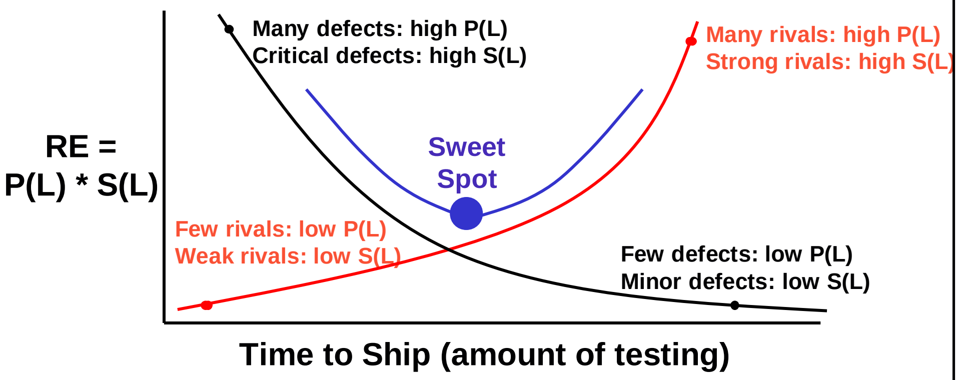
Example RE Profile: Time to Ship

- Loss due to unacceptable dependability
- Loss due to market share erosion

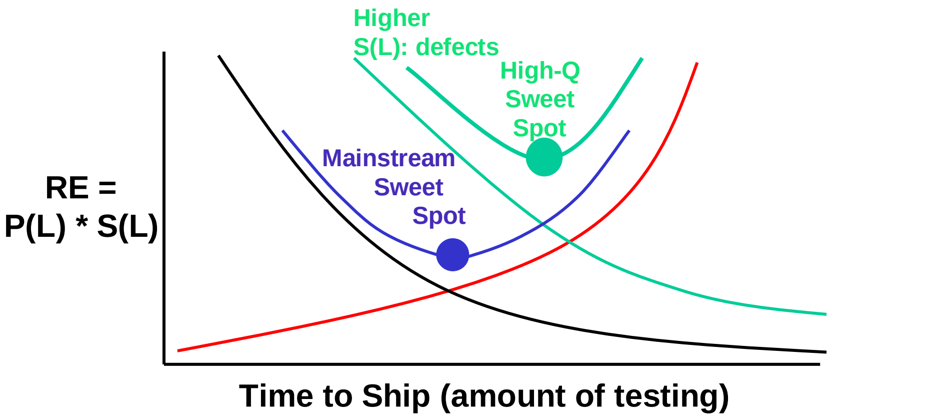


Example RE Profile: Time to Ship

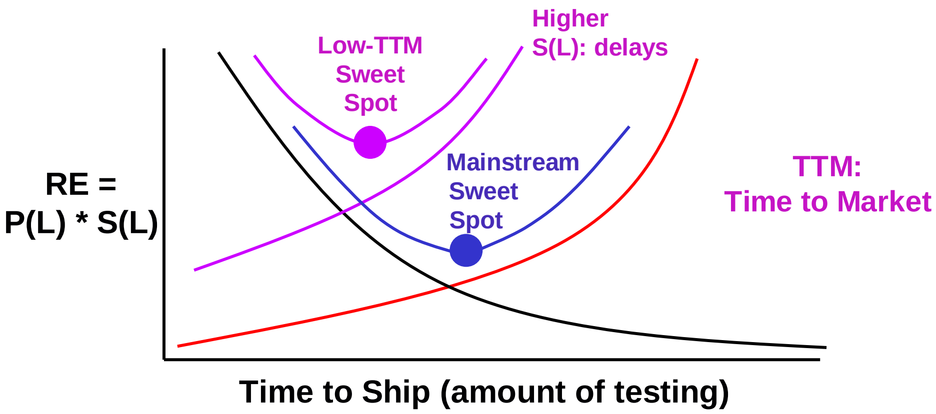
- Sum of Risk Exposures



Comparative RE Profile: Safety-Critical System



Comparative RE Profile: Internet Startup

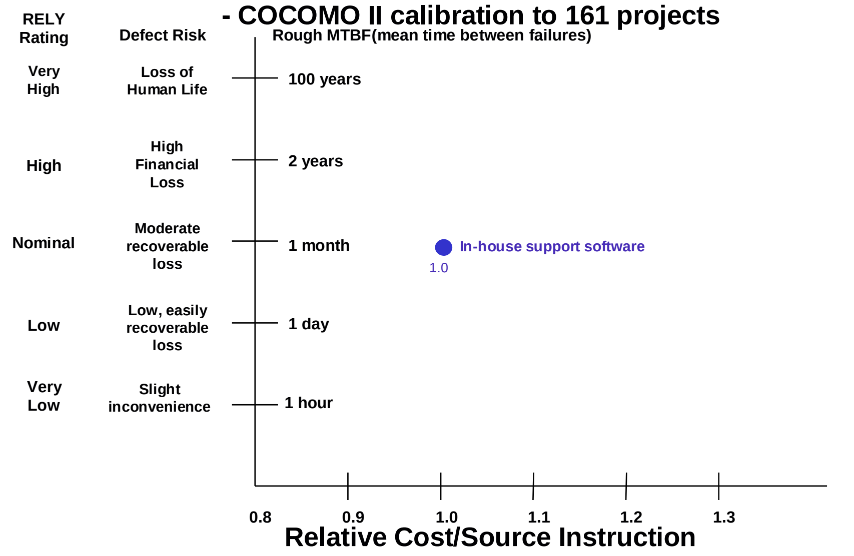


Interim Conclusions

- Unwise to try to compete on both cost/schedule and quality
  - Some exceptions: major technology or marketplace edge
- There are no one-size-fits-all cost/schedule/quality strategies
- Risk analysis helps determine how much testing (prototyping, formal verification, etc.) is enough
  - Buying information to reduce risk
- Often difficult to determine parameter values
  - Some COCOMO II values discussed next

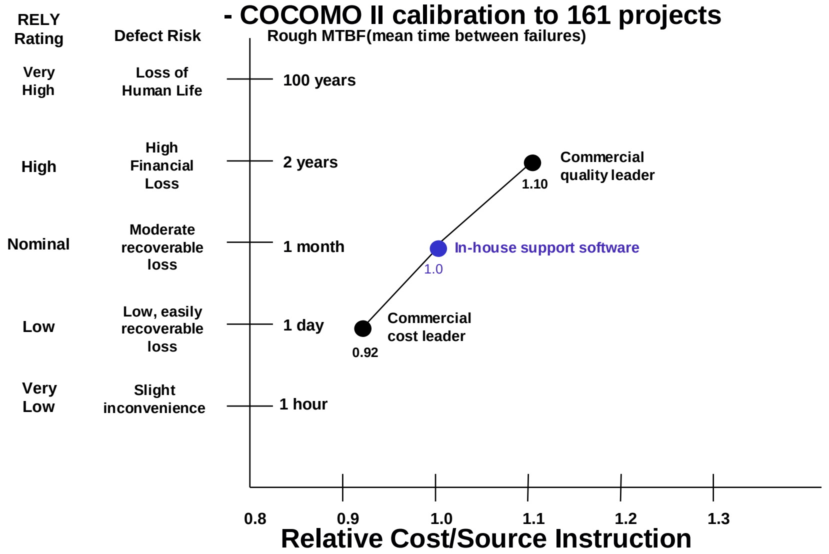
Software Development Cost/Quality Tradeoff

- COCOMO II calibration to 161 projects



Software Development Cost/Quality Tradeoff

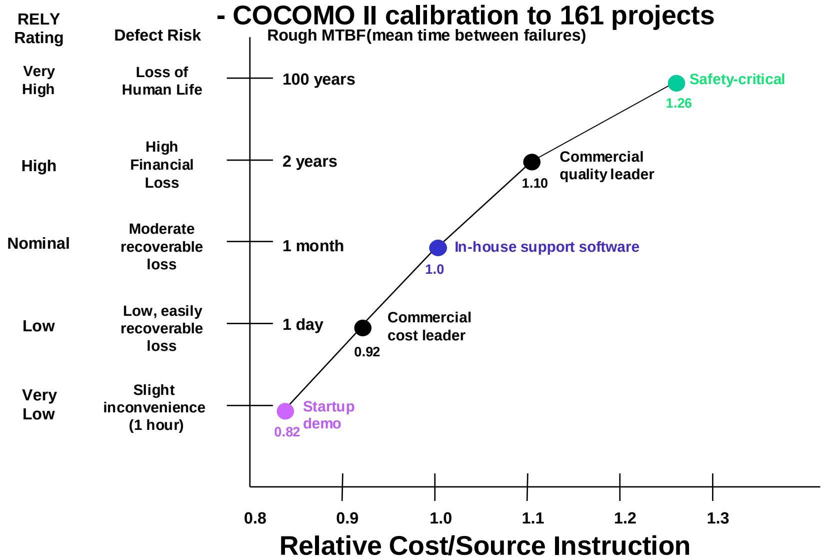
- COCOMO II calibration to 161 projects



University of Southern California
Center for Software Engineering

Software Development Cost/Quality Tradeoff

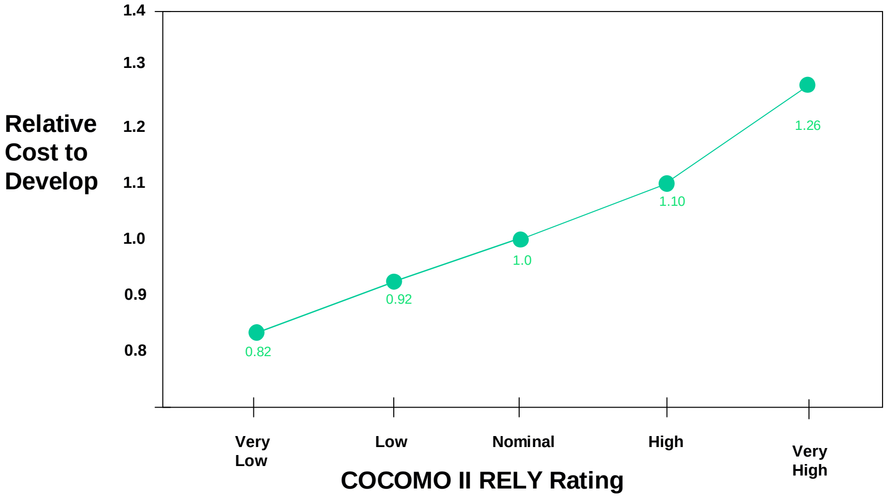
- COCOMO II calibration to 161 projects



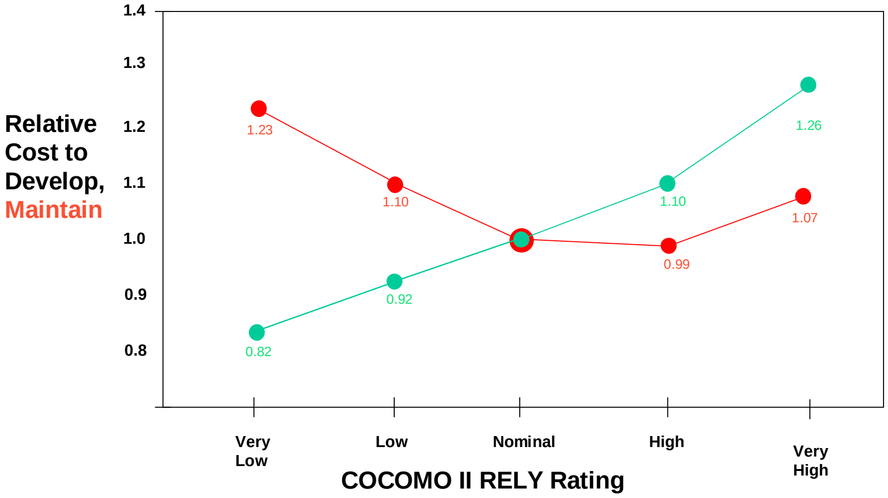
"Quality is Free"

- Did Philip Crosby's book get it all wrong?
- Investments in dependable systems
  - Cost extra for simple, short-life systems
  - Pay off for high-value, long-life systems

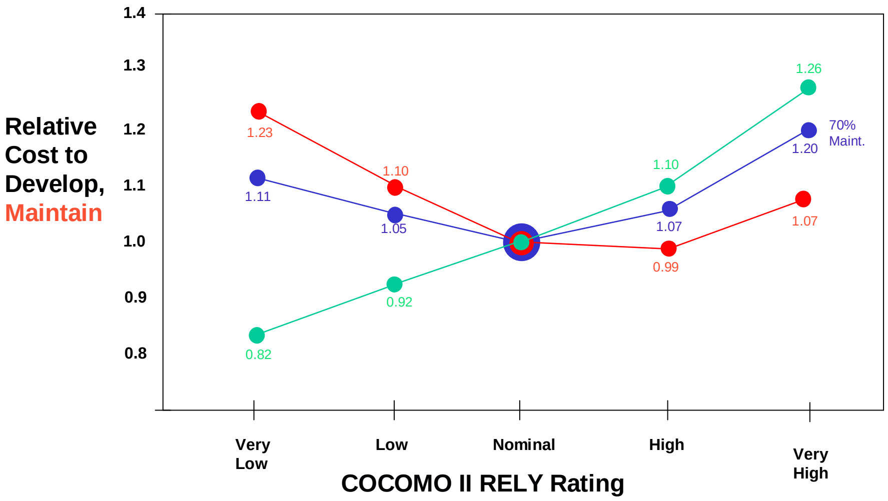
Software Life-Cycle Cost vs. Dependability



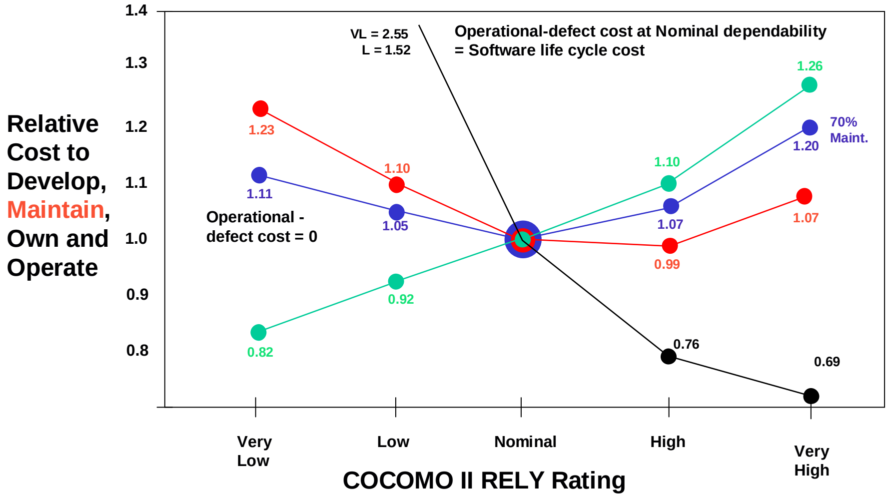
Software Life-Cycle Cost vs. Dependability



Software Life-Cycle Cost vs. Dependability



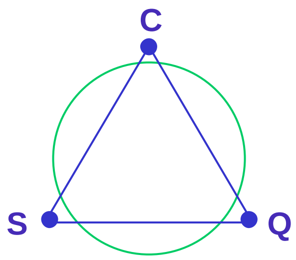
Software Ownership Cost vs. Dependability



Outline

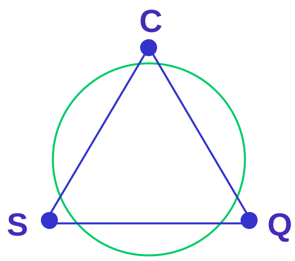
- The business case for software reliability
- What are stakeholders really relying on?
- Software dependability in a competitive world
  - Is market success a monotone function of dependability?
  - Is quality really free?
  - → Is "faster, better, cheaper" really achievable?
  - Value-based vs. value-neutral methods
- Conclusions

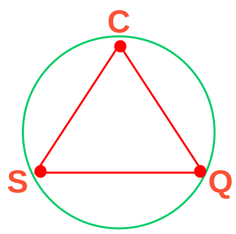
Cost, Schedule, Quality: Pick any Two?



Cost, Schedule, Quality: Pick any Two?

- Consider C, S, Q as Independent Variable
  - Feature Set as Dependent Variable





C, S, Q as Independent Variable

- Determine Desired Delivered Defect Density (D4)
  - Or a value-based equivalent
- Prioritize desired features
  - Via QFD, IPT, stakeholder win-win
- Determine Core Capability
  - 90% confidence of D4 within cost and schedule
  - Balance parametric models and expert judgment
- Architect for ease of adding next-priority features
  - Hide sources of change within modules (Parnas)
- Develop core capability to D4 quality level
  - Usually in less than available cost and schedule
- Add next priority features as resources permit
- Versions used successfully on 32 of 34 USC digital library projects

Value-Based vs. Value Neutral Methods

- Value-based defect reduction
- Constructive Quality Model (COQUALMO)
- Information Dependability Attribute Value Estimation (iDAVE) model

Value-Based Defect Reduction Example: Goal-Question-Metric (GQM) Approach

Goal: Our supply chain software packages have too many defects. We need to get their defect rates down

Question: ?

Value-Based GQM Approach – I

Q: How do software defects affect <u>system value</u> goals?

ask <u>why</u> initiative is needed

- Order processing
- Too much downtime on operations critical path
- Too many defects in operational plans
- Too many new-release operational problems

G: New <u>system-level</u> goal: Decrease software-defect-related losses in operational effectiveness

- With high-leverage problem areas above as specific subgoals

New Q: ?

Value-Based GQM Approach – II

New Q: Perform system problem-area root cause analysis: ask why problems are happening via models

Example: Downtime on critical path

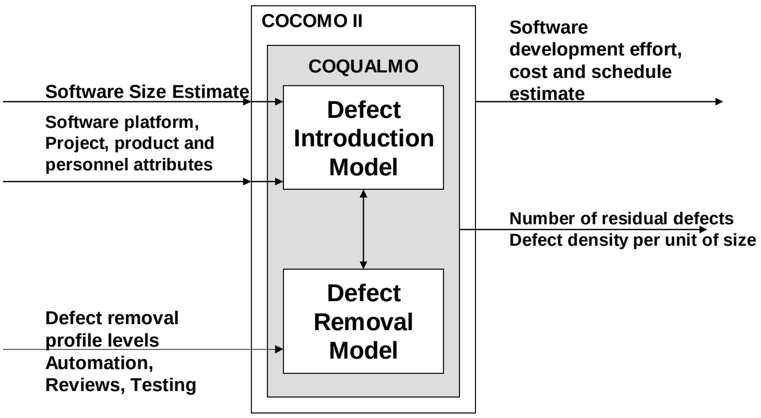
Order items → Validate order → Validate items in stock → Schedule packaging, delivery → Produce status reports → Prepare delivery packages → Deliver order

- **Where are primary software-defect-related delays?**
- **Where are biggest improvement-leverage areas?**
  - Reducing software defects in Scheduling module
  - Reducing non-software order-validation delays
  - Taking Status Reporting off the critical path
  - Downstream, getting a new Web-based order entry system
- **Ask “why not?” as well as “why?”**

Value-Based GQM Results

- **Defect tracking weighted by system-value priority**
  - Focuses defect removal on highest-value effort
- **Significantly higher effect on bottom-line business value**
  - And on customer satisfaction levels
- **Engages software engineers in system issues**
  - Fits increasing system-criticality of software
- **Strategies often helped by quantitative models**
  - COQUALMO, iDAVE

Current COQUALMO System



University of Southern California
Center for Software Engineering

Defect Removal Rating Scales

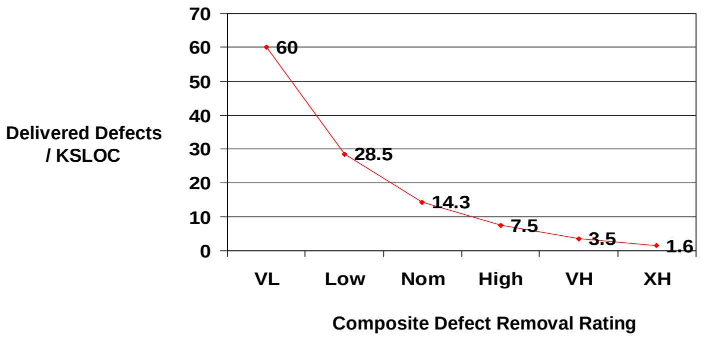
COCOMO II p.263

	Very Low	Low	Nominal	High	Very High	Extra High
Automated Analysis	Simple compiler syntax checking	Basic compiler capabilities	Compiler extension Basic req. and design consistency	Intermediate-level module Simple req./design	More elaborate req./design Basic dist-processing	Formalized specification, verification. Advanced dist-processing
Peer Reviews	No peer review	Ad-hoc informal walk-through	Well-defined preparation, review, minimal follow-up	Formal review roles and Well-trained people and basic checklist	Root cause analysis, formal follow Using historical data	Extensive review checklist Statistical control
Execution Testing and Tools	No testing	Ad-hoc test and debug	Basic test Test criteria based on checklist	Well-defined test seq. and basic test coverage tool system	More advance test tools, preparation. Dist-monitoring	Highly advanced tools, model-based test

University of Southern California
Center for Software Engineering

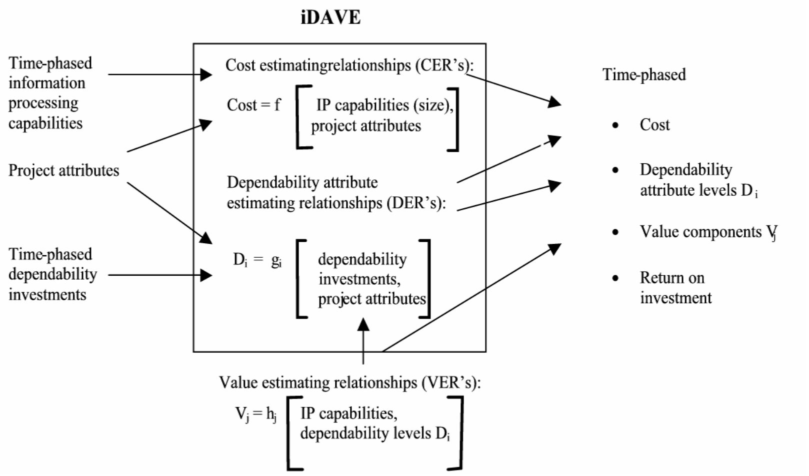
Defect Removal Estimates

- Nominal Defect Introduction Rates



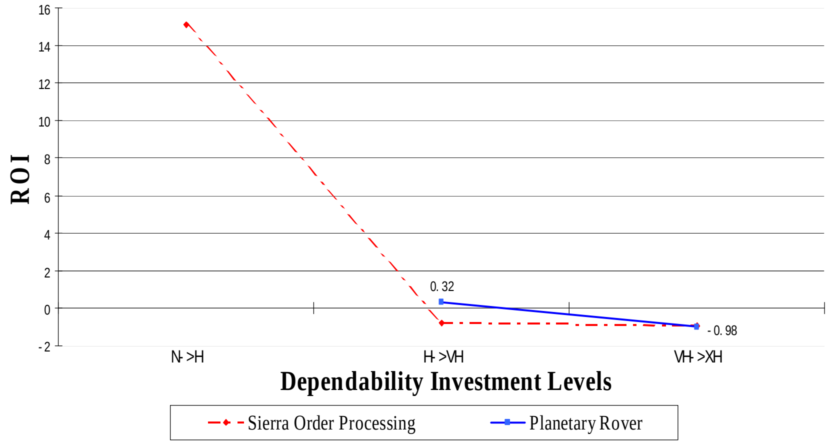
University of Southern California
Center for Software Engineering

iDAVE Model



ROI Analysis Results Comparison

- Business Application vs. Space Application



Conclusions: Software Reliability (SWR)

- There is no universal SWR metric to optimize
  – Need to balance stakeholder SWR value propositions
- Increasing need for value-based approaches to SWR
  – Methods and models emerging to address needs
- “Faster, better, cheaper” is feasible
  – If feature content can be a dependent variable
- “Quality is free” for stable, high-value, long-life systems
  – But not for dynamic, lower-value, short life systems
- Future trends intensify SWR needs and challenges
  – Criticality, complexity, decreased control, faster change