University of Southern California
Center for Systems & Software Engineering

Assessing and Avoiding Technical Debt

Barry Boehm, USC-CSSE

Managing Technical Debt Workshop

June 5, 2012

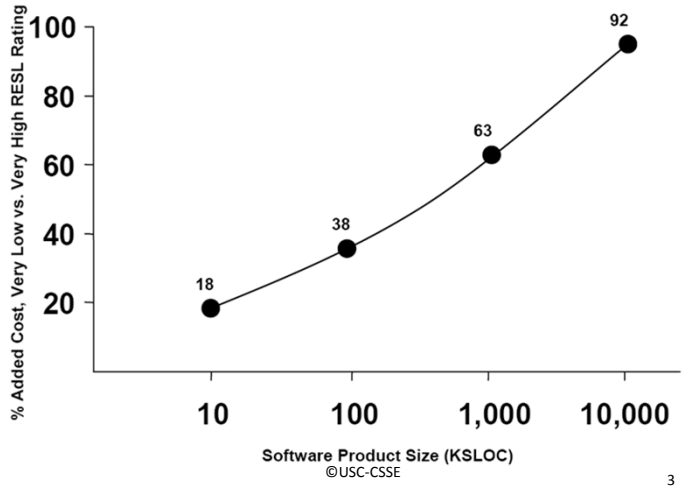
University of Southern California
Center for Systems & Software Engineering

Summary

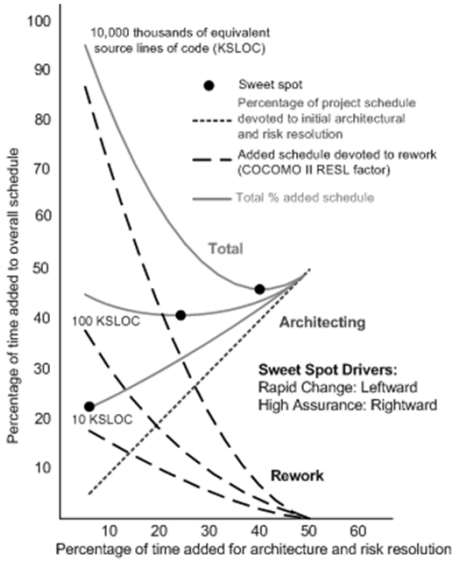
- **Assessing amount of technical debt as necessary rework**
  - **Shortfalls in architecture and risk resolution**
  - **Conspiracies of optimism**
  - **Neglecting change and its effects**
    - **Fixed-price, build-to-spec contracting**
    - **Incremental Development Productivity Decline**
- **Avoiding technical debt**
  - **Evidence-based decision making**
  - **Continuous verification and validation**
  - **Value-based time/cost boxing**

Rework Due to Weak Architecting

Calibration of COCOMO II Architecture and Risk Resolution factor to 161 project data points

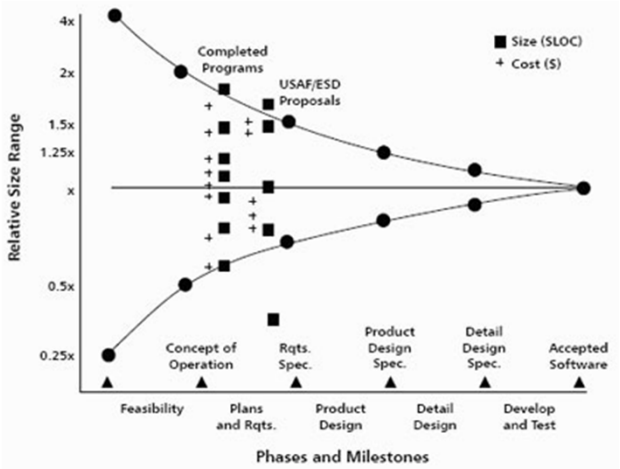


Effect of Size on Best Level of Architecting



The Conspiracy of Optimism

Take the lower branch of the Cone of Uncertainty

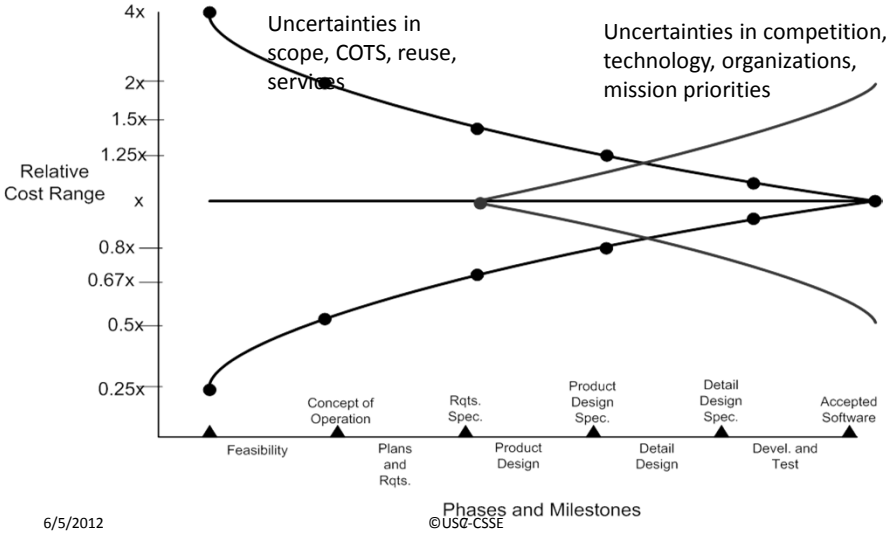


Summary

- Assessing amount of technical debt as necessary rework
  - Shortfalls in architecture and risk resolution
  - Conspiracies of optimism
  - → Neglecting change and its effects
    - Fixed-price, build-to-spec contracting
    - Incremental Development Productivity Decline
- Avoiding technical debt
  - Evidence-based decision making
  - Continuous verification and validation
  - Value-based time/cost boxing

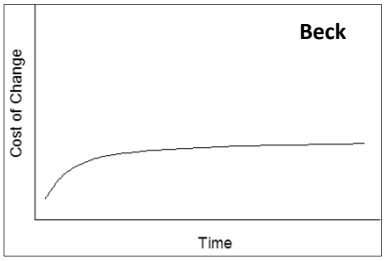
Adaptation Challenges: A Dual Cone of Uncertainty

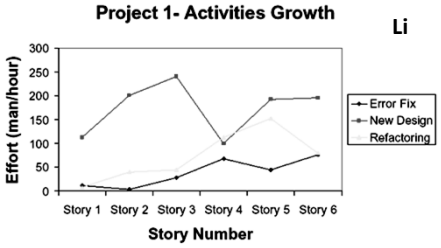
– Need early systems engineering, evolutionary development

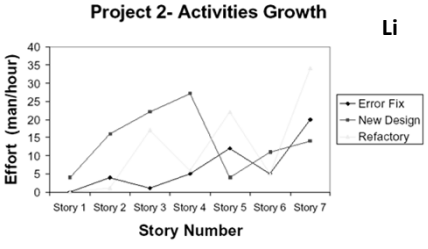


Technical Characteristics (2)

Cost of Change: Beck, Li

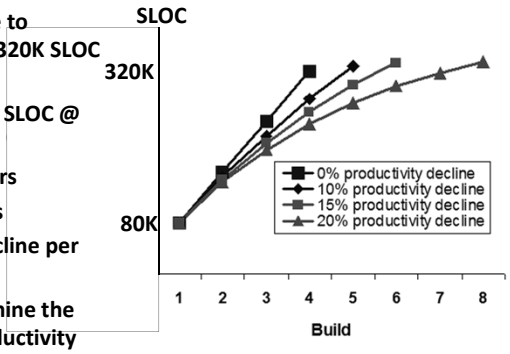






Effects of Incremental Development Productivity Decline

- Model relating productivity decline to number of builds needed to reach 320K SLOC Full Operational Capability
- Assumes Build 1 production of 80K SLOC @ 400 SLOC/PM
  - 200 PM/ 12 mo. = 17 developers
  - Constant staff size for all builds
- Analysis varies the productivity decline per build
  - Extremely important to determine the incremental development productivity decline (IDPD) factor per build



Summary

- Assessing amount of technical debt as necessary rework
  - Shortfalls in architecture and risk resolution
  - Conspiracies of optimism
  - Neglecting change and its effects
    - Fixed-price, build-to-spec contracting
    - Incremental Development Productivity Decline

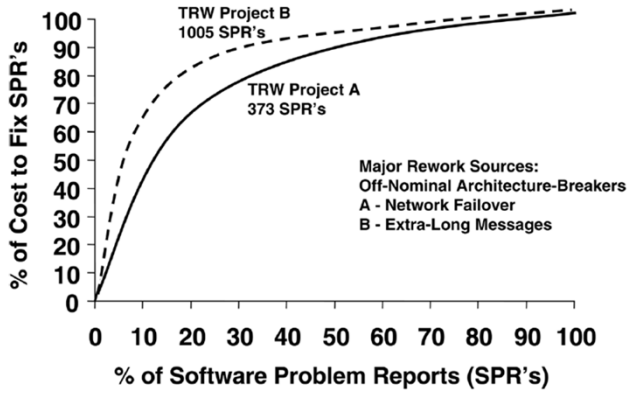
→ Avoiding technical debt
  - Evidence-based decision making
  - Continuous verification and validation
  - Value-based cost/time boxing

Types of Milestone Reviews

- **Schedule-based reviews (contract-driven)**
  - **We'll hold the PDR on April 1 whether we have a design or not**
  - **High probability of proceeding into a Death March**
- **Event-based reviews (artifact-driven)**
  - **The design will be done by June 1, so we'll have the review then**
  - **Large "Death by PowerPoint and UML" event**
    - **Hard to avoid proceeding with many unresolved risks and interfaces**
- **Evidence-based commitment reviews (risk-driven)**
  - **Evidence provided in Feasibility Evidence Description (FED)**
    - **A first-class deliverable**
  - **Shortfalls in evidence are uncertainties and risks**
  - **Should be covered by risk mitigation plans**
  - **Stakeholders decide to commit based on risks of going forward**

Examples of Evidence Utility: Pareto 80-20 Technical Debt Distribution

Contracts: Nominal-case requirements; 90 days to PDR



University of Southern California
Center for Systems & Software Engineering

C4ISR Project C: Architecting for Change

USAF/ESC-TRW CCPDS-R Project*

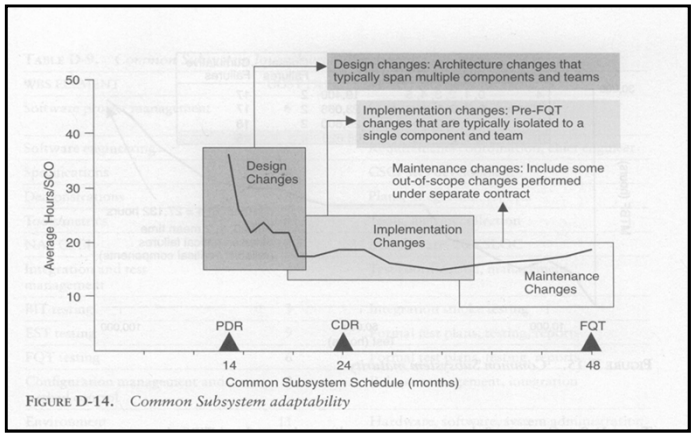


FIGURE D-14. *Common Subsystem adaptability*

When investments made in architecture, average time for change order becomes relatively stable over time…

* Walker Royce, *Software Project Management: A Unified Framework*. Addison-Wesley, 1998.

University of Southern California
Center for Systems & Software Engineering

Some Risk and Technical Debt is Inevitable

Use low-priority features as risk reserve

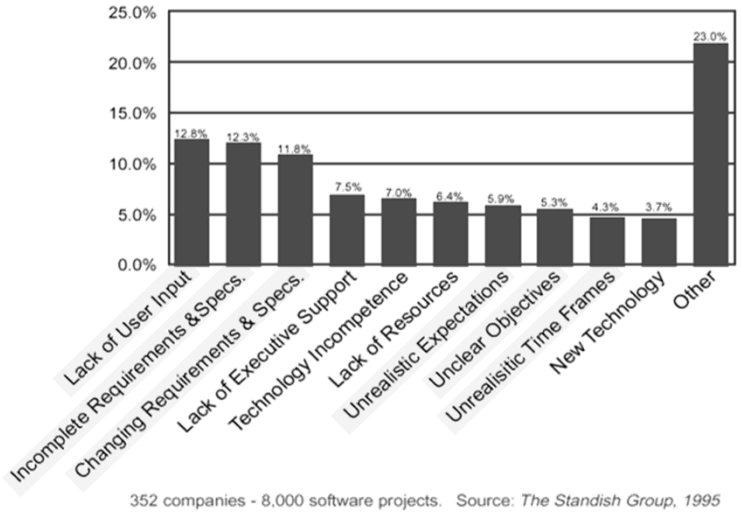
1. **Stakeholder value-based feature prioritization**
2. **Cost/Schedule range estimation and core-capability determination**
   - **Top-priority features achievable within cost/schedule with 90% confidence**
3. **Architecting for ease of adding or dropping borderline-priority features**
   - **And for accommodating post-IOC directions of growth**
4. **Incremental development**
   - **Core capability as increment 1**
5. **Change and progress monitoring and control**
   - **Add or drop borderline-priority features to meet cost/schedule**

Backup Charts

Magnitude of Software Failures Problem:
Standish Surveys of Commercial Projects

Year	2000	2002	2004	2006	2008
Within budget and schedule	28	34	29	35	32
Prematurely cancelled	23	15	18	19	24
Budget or schedule overrun	49	51	53	46	44

Why Software Projects Fail



352 companies - 8,000 software projects. Source: *The Standish Group, 1995*

University of Southern California
Center for Systems & Software Engineering

Example: Agility Underestimating Complexity: Thoughtworks Lease Management

- XP replaced ineffective traditional development
- Problems when project moved beyond XP assumptions
  - The effort to develop or modify a story really does not increase with time and story number
  - Trusting people to get everything done on time is compatible with fixed schedules and diseconomies of scale
  - Simple design and YAGNI scale up easily to large projects
- Disciplined practices enabled XP to scale up
  - High-level architectural plans to provide essential big-picture information
  - More careful definition of milestone completion criteria to avoid "finishing" but not finishing
  - Use of design patterns and architectural solutions rather than simple design to handle foreseeable change

Example: Human Desire to Please

Be careful what you ask for. You may get it.

Weinberg Programmer Objectives Experiment

	Resulting Rank of Performance[b]				
Team objective: Optimize	**Effort to Complete**	**Number of Statements**	**Memory Required**	**Program Clarity**	**Output Clarity**
Effort to complete	*1*	4	4	5	3
Number of statements	2-3	1	2	3	5
Memory required	5	2	1	4	4
Program clarity	4	3	3	2	2
Output clarity	2-3	5	5	1	1

Integration Matrix

Integration styles vs. Properties		Topology				Linkage		Connector					
		Point-to-Point	Hub and Spoke	Shared Bus	Peer-to-Peer	Shared Data	Messaging	Explicit invocation	Data Streaming	Adapter	Translator	Arbitrator	Distributor
Interaction	Distributed	o	+	+	+	o	+	+	+	o	o	+	+
	Local	-	-	+	-	+	o	+	+	o	o	o	-
	Secure	-	-	o	+/-	-	o	o	o	o	o	+	-
	Data intensive	+	-	-	-	+	-	o	+	o	-	+	+
	Data formats incompatible	o	+	o	-	-	+	o	o	o	+	o	o
	Data consistency	o	+	o	-	+	o	o	-	o	o	+	o
	Interaction protocols incompatible	o	+	o	-	+	o	-	o	+	o	o	o
	Reliable	+	-	-	+	-	+	+	o	o	o	+	o
	Real time	+	-	+/-	-	+	-	+	-	-	-	+	o
	One-to-many	-	+	+	+	+/-	+	+	+	+	+	+	+
	Many-to-one	-	+	o	+/-	o	+	-	-	-	-	-	+
	Always available	+	-	o	+	-	+	o	o	o	o	+	o
	Periodically scheduled	+	o	o	-	o	o	o	o	o	o	o	o
System	Loose coupling	-	+	+	+/-	-	+	+	+	+	+	+	+
	Robust	-	-	+	+	-	+	+/-	+	+	+	+	+
	Dynamically reconfigurable	-	o	+	+	o	+	+	+	+	+	+	o
	Scalable	-	-	+	+	-	+	o	o	o	o	+	+
	Caching	-	+	+	o	+	o	-	-	o	-	+	+
	Distributed transactions	-	+	+	+/-	+	+	+	o	o	o	+	+

P2P architectures effective at quickly disseminating data

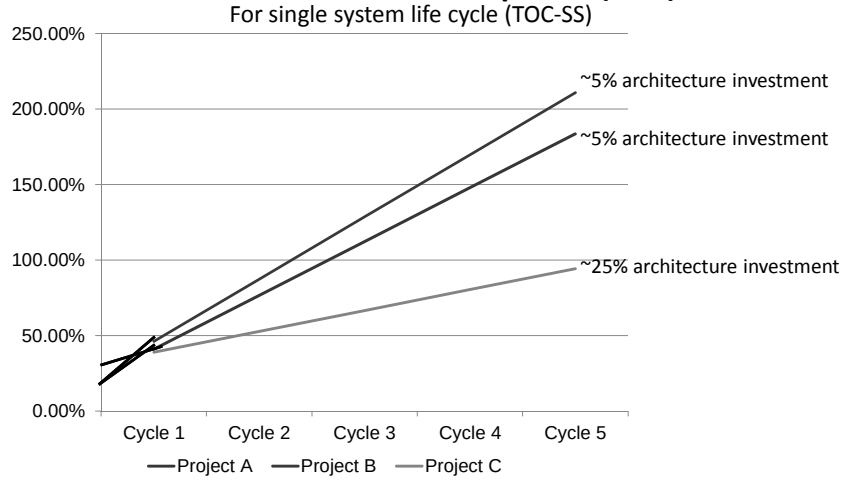
Rework Sources Analysis: Projects A and B

- Change processing over 1 person-month = 152 person-hours

Category	Project A	Project B
Extra long messages		3404+626+443+328+244= 5045
Network failover	2050+470+360+160= 3040	
Hardware-software interface	620+200= 820	1629+513+289+232+166= 2832
Encryption algorithms		1247+368= 1615
Subcontractor interface	1100+760+200= 2060	
GUI revision	980+730+420+240+180 =2550	
Data compression algorithm		910
External applications interface	770+330+200+160= 1460	
COTS upgrades	540+380+190= 1110	741+302+221+197= 1461
Database restructure	690+480+310+210+170= 1860	
Routing algorithms		494+198= 692
Diagnostic aids	360	477+318+184= 979
TOTAL:	**13620**	**13531**

Relative* Total Ownership Cost (TOC)

For single system life cycle (TOC-SS)



* Cumulative architecting and rework effort relative to initial development effort

Evidence- and risk-based decision making

- **Evidence provided by developer and validated by independent experts that:**

  If the system is built to the specified architecture, it will
  - **Satisfy the requirements: capability, interfaces, level of service, and evolution**
  - **Support the operational concept**
  - **Be buildable within the budgets and schedules in the plan**
  - **Generate a viable return on investment**
  - **Generate satisfactory outcomes for all of the success-critical stakeholders**
- **All major risks resolved or covered by risk management plans (shortfalls in evidence are uncertainties and risks)**
- **Serves as basis for stakeholders' commitment to proceed**

Can be used to strengthen current schedule- or event-based reviews

EvDev Budgeting and Scheduling Concerns: Incremental Development Productivity Decline (IDPD)

- **Example: Site Defense BMD Software**
  - **5 builds, 7 years, $100M; operational and support software**
  - **Build 1 productivity over 200 LOC/person month**
  - **Build 5 productivity under 100 LOC/PM**
    - **Including Build 1-4 breakage, integration, rework**
    - **318% change in requirements across all builds**
    - **IDPD factor = 20% productivity decrease per build**
  - **Similar trends in later unprecedented systems**
  - **Not unique to DoD: key source of Windows Vista delays**
- **Maintenance of full non-COTS SLOC, not ESLOC**
  - **Build 1: 200 KSLOC new; 200K reused@20% = 240K ESLOC**
  - **Build 2: 400 KSLOC of Build 1 software to maintain, integrate**

IDPD Cost Drivers: Conservative 4-Increment Example

- Some savings: more experienced personnel (5-20%)
  - Depending on personnel turnover rates
- Some increases: code base growth, diseconomies of scale, requirements volatility, user requests
  - Breakage, maintenance of full code base (20-40%)
  - Diseconomies of scale in development, integration (10-25%)
  - Requirements volatility; user requests (10-25%)
- Best case: 20% more effort (IDPD=6%)
- Worst case: 85% (IDPD=23%)