CSE503: SOFTWARE ENGINEERING

RESEARCH APPROACHES, ECONOMICS AND GOVERNANCE

David Notkin
Spring 2011

Evaluation of SE research

2

- What convinces you?
- Why?

Possible answers include

3

- Intuition
- Quantitative assessments
- Qualitative assessments
- Case studies
- … other possible answers?

Brooks on evaluation

4

- The first user gives you infinite utility – that is, you learn more from the first person who tries an approach than from every person thereafter
- In HCI, Brooks compared
  - "narrow truths proved convincingly by statistically sound experiments, and
  - broad 'truths', generally applicable, but supported only by possibly unrepresentative observations."
  - Grasping Reality Through Illusion -- Interactive Graphics Serving Science. *Proc 1988 ACM SIGCHI*

More on Brooks by Mary Shaw

5

- "Brooks proposes to relieve the tension through a certainty-shell structure – to recognize three nested classes of results,
  - Findings: well-established scientific truths, judged by truthfulness and rigor;
  - Observations: reports on actual phenomena, judged by interestingness;
  - Rules of thumb: generalizations, signed by their author but perhaps incompletely supported by data, judged by usefulness."
- What Makes Good Research in Software Engineering? *International Journal of Software Tools for Technology Transfer*, 2002

503 11sp © UW CSE • D. Notkin

Shaw: research questions in SE

6

Type of question	Examples
Method or means of development	How can we do/create (or automate doing) X? What is a better way to do/create X?
Method for analysis	How can I evaluate the quality/correctness of X? How do I choose between X and Y?
Design, evaluation, or analysis of a particular instance	What is a (better) design or implementation for application X? What is property X of artifact/method Y? How does X compare to Y? What is the current state of X / practice of Y?
Generalization or characterization	Given X, what will Y (necessarily) be? What, exactly, do we mean by X? What are the important characteristics of X? What is a good formal/empirical model for X? What are the varieties of X, how are they related?
Feasibility	Does X even exist, and if so what is it like? Is it possible to accomplish X at all?

503 11sp © UW CSE • D. Notkin

Shaw: types of SE results

7

Type of result	Examples
Procedure or technique	New or better way to do some task, such as design, implementation, measurement, evaluation, selection from alternatives, Includes operational techniques for implementation, representation, management, and analysis, but not advice or guidelines
Qualitative or descriptive model	Structure or taxonomy for a problem area; architectural style, framework, or design pattern; non-formal domain analysis Well-grounded checklists, well-argued informal generalizations, guidance for integrating other results,
Empirical model	Empirical predictive model based on observed data
Analytic model	Structural model precise enough to support formal analysis or automatic manipulation
Notation or tool	Formal language to support technique or model (should have a calculus, semantics, or other basis for computing or inference) Implemented tool that embodies a technique
Specific solution	Solution to application problem that shows use of software engineering principles – may be design, rather than implementation Careful analysis of a system or its development Running system that embodies a result; it may be the carrier of the result, or its implementation may illustrate a principle that can be applied elsewhere
Answer or judgment	Result of a specific analysis, evaluation, or comparison
Report	Interesting observations, rules of thumb

• D. Notkin

Shaw

8

- Types of validation

Type of validation	Examples
Analysis	I have analyzed my result and find it satisfactory through ...ormal analysis) ... rigorous derivation and proof (empirical model) ... data on controlled use(controlled ... carefully designed statistical experiment) experiment
Experience	My result has been used on real examples by someone other than me, and the evidence of its correctness / usefulness / effectiveness is ...alitative model) ... narrative(empirical model, ... data, usually statistical, on practice (notation, tool) ... comparison of this with similar results in technique) actual use
Example	Here's an example of how it works on (toy example) ... a toy example, perhaps motivated by reality (slice of life) ...a system that I have been developing
Evaluation	Given the stated criteria, my result... (descriptive model) ... adequately describes the phenomena of interest ... (qualitative model) ... accounts for the phenomena of interest... (empirical model) ... is able to predict ... because ..., or ... gives results that fit real data ... Includes feasibility studies, pilot projects
Persuasion	I thought hard about this, and I believe... (technique) ... if you do it the following way, ... (system) ... a system constructed like this would ... (model) ... this model seems reasonable Note that if the original question was about feasibility, a working system, even without analysis, can be persuasive
Blatant assertion	No serious attempt to evaluate result

Tichy et al. on quantitative evaluation

9

- Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software* 1995
  - Tichy, Lukowicz, Prechelt & Heinz
- Abstract:
  A survey of 400 recent research articles suggests that computer scientists publish relatively few papers with experimentally validated results. The survey includes complete volumes of several refereed computer science journals, a conference, and 50 titles drawn at random from all articles published by ACM in 1993. The journals of *Optical Engineering (OE)* and *Neural Computation (NC)* were used for comparison. .. (con't)

Con't

10

Of the papers in the random sample that would require experimental validation, 40% have none at all. In journals related to software engineering, this fraction is 50%. In comparison, the fraction of papers lacking quantitative evaluation in *OE* and *NC* is only 15% and 12%, respectively. Conversely, the fraction of papers that devote one fifth or more of their space to experimental validation is almost 70% for *OE* and *NC*, while it is a mere 30% for the computer science (CS) random sample and 20% for software engineering. The low ratio of validated results appears to be a serious weakness in computer science research. This weakness should be rectified for the long-term health of the field. The fundamental principle of science, the definition almost, is this: the sole test of the validity of any idea is experiment. —Richard P. Feynman. Beware of bugs in the above code; I have only proved it correct, not tried it. —Donald E. Knuth

Technology transfer: briefly

11

- Not a consumer problem
- Not a producer problem
- An ecosystem issue

Evolving the High Performance Computing and Communications Initiative to Support the Nation's Information Infrastructure (1995)
"Brooks-Sutherland" report
Computer Science and Telecommunications Board (CSTB)

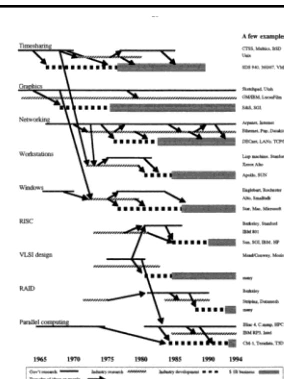


FIGURE 1.2 Government-sponsored computing research and development stimulates creation of innovative ideas and industries. Dates apply to horizontal bars, but not to arrows showing transfer of ideas and people. Table 1.1 is a companion to this figure.

Software engineering economics

13

- The phrase dates to around 1981, when Barry Boehm published his tome with the same title
- Boehm identified engineering economics as one "scientific principle" in which software engineering fell short of hardware engineering
- To the first order, the focus of his book was on how to better estimate effort, cost and schedule for large software projects – **COCOMO** (**CO**nstructive **CO**st **MO**del)

COCOMO basics

14

- Algorithmic software cost estimation modeled with a regression formula that has parameters derived from historical project data and current project characteristics
- The basic COCOMO equations take the form
  - Effort Applied = $\mathbf{a}(\text{KLOC})^{\mathbf{b}}$ (person-months)
  - Development Time = $\mathbf{c}(\text{Effort Applied})^{\mathbf{d}}$ (months)
  - People required = Effort Applied / Development Time (count)

	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Regression parameters Basic COCOMO

15

- Based on waterfall-based 63 projects at TRW Aerospace
- Projects from 2KLOC to 100KLOC, languages from assembler to PL/I
- The Basic Model designed for rough order-of-magnitude estimates, focused on small to medium-sized projects
  - Three sets of parameters: organic, semi-detached and embedded

Intermediate COCOMO

16

- Uses more parameters (cost drivers) that account for additional differences estimates
- Product attributes: required software reliability, complexity of the product, …
- Hardware attributes: run-time performance constraints, memory constraints, …
- Personnel attributes: software engineering capability, applications experience, programming language experience, …
- Project attributes: use of software tools, application of software engineering methods, …

Intermediate COCOMO

17

- The 15 sub-attributes are each rated from “very low” to “extra-high” with six discrete choices
- Effort multipliers are empirically derived and the EAF is the product of the multipliers

Cost Drivers	Ratings: Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine envionment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Software engineer capability	1.29	1.13	1.00	0.91	0.82	
Applications experience	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Use of software tools	1.24	1.10	1.00	0.91	0.82	
Application of software engineering methods	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

http://neohumanism.org/i/in/intermediate_cocomo_1.html

Intermediate COCOMO

18

- $E = a(KLOC)^b \times EAF$
  - And similarly for development time and people counts
- There is a separate table for parameters a and b across organic, semi-detached, embedded for Intermediate COCOMO

Detailed COCOMO & COCOMO II

19

- Detailed COCOMO also accounts for the influence of individual project phases
- COCOMO II was developed and released in 1997, aimed at (then) modern software projects
  - Newly tuned parameters
  - Accounted for move from mainframes to desktops, from batch to interface computation, to code reuse, etc.

1981 Boehm book also discusses

20

- Multiple-goal decision analysis
  - Most optimization theory assumes that there is a single objective function to maximize
  - Models like this one account for multiple goals that must be balanced in a definable manner
- Risk analysis
  - Foundation for his later work in the spiral model
- And more…

Boehm & Sullivan "Software Economics" roadmap (ICSE 2000)

21

- "The core competency of software engineers is in making technical software product and process design decisions. Today, however, there is a 'disconnect' between the decision criteria that tend to guide software engineers and the value creation criteria of organizations in which software is developed. It is not that technical criteria, such as information hiding architecture, documentation standards, software reuse, and the need for mathematical precision, are wrong. On average, they are enormously better than no sound criteria.

Con't

22

- "However, software engineers are usually not involved in or often do not understand enterprise-level value creation objectives. The connections between technical parameters and value creation are understood vaguely, if at all. There is rarely any real measurement or analysis of how software engineering investments contribute to value creation. And senior management often does not understand success criteria for software development or how investments at the technical level can contribute fundamentally to value creation. As a result, technical criteria tend to be applied in ways that in general are not connected to, and are thus usually not optimal for, value creation."

Thinking about value

23

- Decision theory (or utility theory) defines a framework for decisions under uncertainty, depending on the risk characteristics of decision makers
- This is closely related to (again) multi-objective decision-making
- Classical corporate finance uses net present value (NPV) as an investment decision criterion and computes it by discounted cash flow analysis (DCF) – can't make a business case without these

NPV example from Wikipedia

24

- A corporation must decide whether to introduce a new product line. The new product will have startup costs, operational costs, and incoming cash flows over six years. This project will have an immediate (t=0) cash outflow of $100,000 (which might include machinery, and employee training costs). Other cash outflows for years 1-6 are expected to be $5,000 per year. Cash inflows are expected to be $30,000 each for years 1-6. All cash flows are after-tax, and there are no cash flows expected after year 6. The required rate of return is 10%.

Con't

- The table shows the present value (PV) for each year
- The NPV is the sum of the PVs
- In this case, it's $8,881.52
- A positive NPV means it would be better to invest in the project than to do nothing – but there might be other opportunities with higher NPV

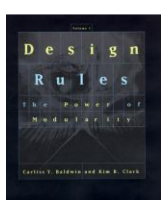
Year	Cashflow	Present Value
T=0	$\frac{-100,000}{(1+0.10)^0}$	-$100,000
T=1	$\frac{30,000-5,000}{(1+0.10)^1}$	$22,727
T=2	$\frac{30,000-5,000}{(1+0.10)^2}$	$20,661
T=3	$\frac{30,000-5,000}{(1+0.10)^3}$	$18,783
T=4	$\frac{30,000-5,000}{(1+0.10)^4}$	$17,075
T=5	$\frac{30,000-5,000}{(1+0.10)^5}$	$15,523
T=6	$\frac{30,000-5,000}{(1+0.10)^6}$	$14,112

Real options

- DCF/NPV treats assets as passively held – not actively managed
- But projects are (or can be ☺) actively managed
  - Management usually has the flexibility to make changes to real investments in light of new information. (e.g., to abandon a project, enter a new market, etc.)
- The key idea of real options is to treat such flexibility as an option, and to (in some cases) price them using techniques related to those for financial options

Baldwin and Clark (2000)

- Baldwin and Clark view Parnas' information hiding modules as creating options
- They value these and develop a theory of how modularity in design influenced the evolution of the industry structure for computers over the last forty years
- Non-modular systems must be kept or replaced as a whole
- A system of independent modules can be kept or replaced (largely) individually based on judgments of improvement (or not)
- Modularity provides a portfolio of options vs. an option on a portfolio



DSMs: design structure matrices

- The parameters are A, B, and C
- The X in row B, column A means that good choice for B depends on the choice made for A
- Parameters requiring mutual consistency are *interdependent, resulting in symmetric marks:* (B,C) and (C,B)
- When one parameter choice must precede another the parameters are said to be *hierarchically dependent:* (B,A)
- *Independent* parameters can be changed without coordination

	A	B	C
A	.		
B	X	.	X
C		X	.

Figure 1: DSM for a design of three parameters.

Material from
Sullivan, Griswold, Cai, Hallen. The structure and value of modularity in software design. ESEC/FSE 2001

Splitting

29

- DSMs may not show largely independent designs
- In these cases, one approach is to apply splitting
- Break a dependence with a new parameter that constrains the values of the original parameters – this means, in part, that they depend on it
- Fix the value of the new parameter so that the original parameters to be changed independently as long as they are only changed in ways consistent with the new constraint
- For example, introduce a new interface (I, in the below example)

	I	A	B	C
I	.			
A	X	.		
B	X		.	X
C			X	.

Figure 3: DSM for a modular design obtained by splitting.

Parnas KWIC

30

	A	D	G	J	B	E	H	K	C	F	I	L	M
A - Input Type	.												
D - Circ Type		.											
G - Alph Type			.										
J - Out Type				.									
B - In Data					.	X	X						
E - Circ Data					X	.	X						
H - Alph Data					X	X	.						
K - Out Data								.					
C - Input Alg	X				X				.				
F - Circ Alg		X			X	X				.			
I - Alph Alg			X		X	X	X				.		
L - Out Alg				X	X		X	X				.	
M - Master	X	X	X	X									.

Figure 5: DSM for strawman modularization

	N	A	D	G	J	O	P	B	C	E	F	H	I	K	L	M
N - Line Type	.															
A - In Type		.														
D - Circ Type			.													
G - Alph Type				.												
J - Out Type					.											
O - Line Data	X					.	X									
P - Line Alg	X					X	.									
B - In Data		X						.	X							
C - In Alg	X	X						X	.							
E - Circ Data	X		X							.	X					
F - Circ Alg	X		X							X	.					
H - Alph Data	X			X								.	X			
I - Alph Alg	X			X								X	.			
K - Out Data					X									.	X	
L - Out Alg	X				X									X	.	
M - Master	X	X	X	X	X											.

Figure 7: DSM for information hiding modularization

NOV (net option value)

31

- A module creates an opportunity
  - to invest in k experiments to create candidate replacements,
  - each at a cost related to the complexity of the module
  - if any of the results are better than the existing choice, to substitute in the best of them
  - at a cost that related to the visibility of the module to other modules in the system

KWIC NOV

32

- The option value of each module is the value at the peak
- Sum the module NOV's
  - 0.26 for the strawman design
  - 1.56 for the information-hiding

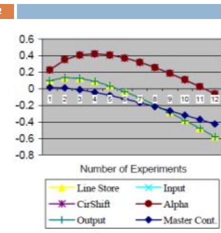


Figure 13: Options Values for Information Hiding Design

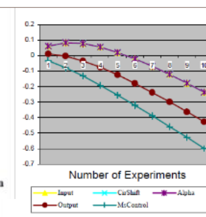


Figure 14: Options Values for Strawman Design in

Status

- The basic idea seems to make sense to many people
- One of the core problems is the notion of how to tune the model parameters
  - Financial markets set parameters based primarily on scads of historic data
  - COCOMO set parameters based on careful studies of a reasonably large set of reasonably similar software projects
  - Tuning parameters for modularity seems more complicated

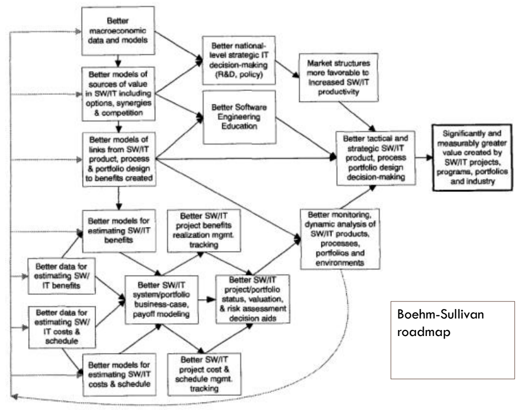
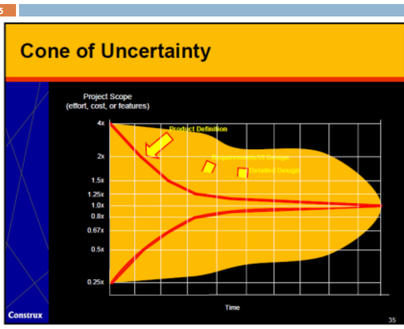


Figure 1: Roadmap for research in software engineering economics.

Boehm-Sullivan roadmap

McConnell's cone of uncertainty ICSE 2009 keynote



Governance of Software Development

- Clay Williams, IBM Research
- Slides directly taken from an NSF workshop presentation Governance @ IBM Future Directions

Governance of Software Development Strategic Initiative

- Goal: Develop the science and technology that enables the Rational software delivery platform to provide support for *governing the business* of software development.

Development Governance

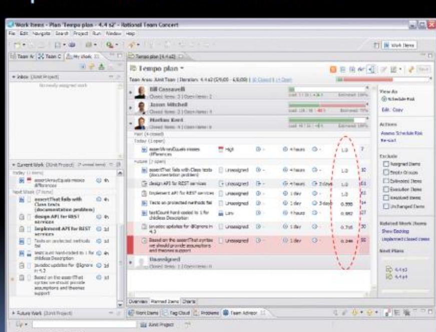
Value and Risk Management

Organizational Design and Collaboration

Tempo - Overview

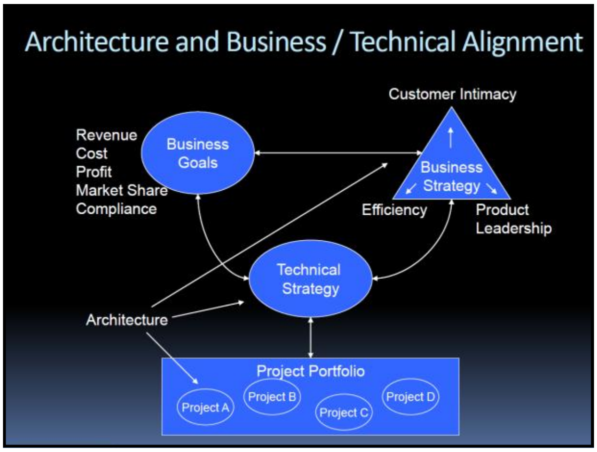
- Problem Statement
  - When project teams commit to a schedule, they are placing a bet. It would be extremely valuable for them to know the odds of winning.
- Approach
  - Capture "bottom-up" predictions regarding the time necessary to complete each task in a work breakdown.
  - Rather than discrete predictions, capture triangular distribution that reflects the fact predications are random variables.
  - Develop optimized scheduling approaches that rapidly reduce schedule risk in the project
  - Surface schedule risks to allow teams to better manage scheduling issues.
- What is hard?
  - Providing a tool that is easy to use and supportive of "what-if" risk mitigation analysis requires addressing subtle and difficult usability issues.
  - The variety of optimizations and analyses require significant mathematical skill.

Tempo in Rational Team Concert



Architectural and Social Governance of Software Development

- Research Goals
  - Exploit / expand the role architectures play as "boundary objects" spanning multiple domains of discourse.
  - Develop techniques for exploring key structural and behavioral properties of architectures (software, IT, and EA), the socio-technical dynamics of the teams producing and consuming them, and how these two areas can be aligned and engender communication beyond the technical domain.
  - Develop / extend architectural approaches to support business decisions and value management.
  - Understand the interplay across the value / architectural / socio-technical domains.
- Collaborations
  - CMU (Jim Herbsleb)
  - Harvard Business School (Carliss Baldwin) - pending
  - Virginia (Kevin Sullivan)



My bottom line

42

- The long-term goal of software engineering economics is to help everybody make more sensible decisions
  - Technical decisions
  - Business decisions
  - Project management decisions
- Not one of these is primary with the others secondary – but that is how we each seem to treat the others
- Better understanding the links among them is crucial; the models may give us opportunities to better understand these links
- I am always scared that quantification tends to lead to a focus on the quantities, and there is often a disconnect between the quantities we can measure and want we want to do

503 11sp © UW CSE • D. Notkin