How to Avoid Drastic Software Process Change (using Stochastic Stability)

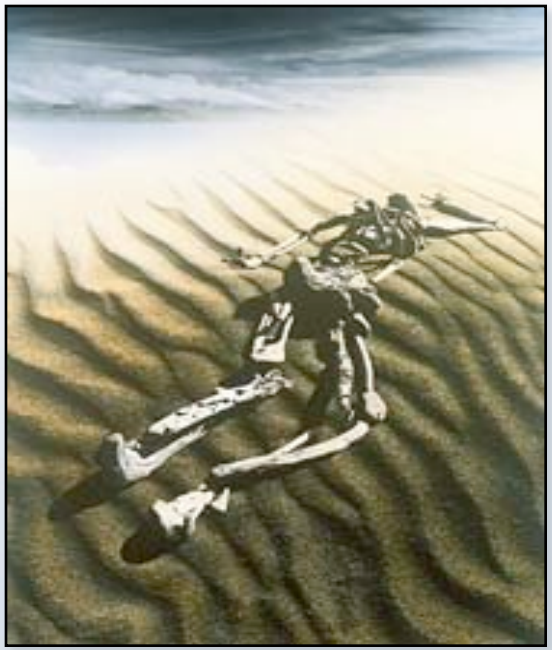
Tim Menzies, Steve Williams, Oussama El-Rawas (WVU)
Jairus Hihn (JPL)
Barry Boehm (USC)

ICSE'09

Stochastic stability

- "For all is but a woven web of guesses."
  - -- Xenophanes (570 – 480 BCE)
- Seek what holds true over the space of all guesses.
  - Surprisingly, happily, such stable conclusions exist.
- Bad idea for:
  - The safety-critical guidance system of a manned rocket.
- Good idea for:
  - Exploring the myriad space of possibilities associated with software project manager.

The process "data drought"



Yet another drought victim

Fenton07: "….much of the current software metrics research is inherently irrelevant to the industrial mix … any software metrics program that depends on some extensive metrics collection is doomed to failure."

e.g. After 26 years, Boehm collected less than 200 sample projects for the COCOMO effort database

A different kind of learning

Past

- If data mining: data = lots; experts= not
  then (e.g.) decision trees
  - Learn model from data

Future?

- If data = lots; experts = lots then (e.g.) Bayes nets
  - Initialize using expert
  - Tune with data
  - Audit with experts
- If data=no and experts = yes then (e.g.) STAR
  - Reuse model, replace uncertain point with ranges
  - Monte Carlo across ranges
  - AI search seeks stable
    conclusions across simulations

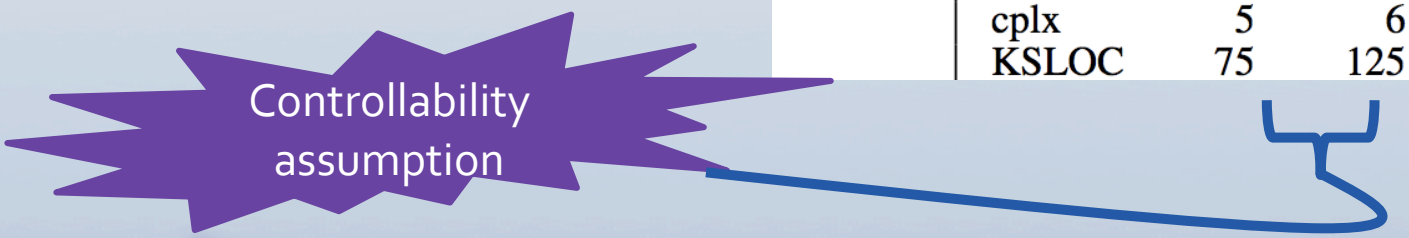
This talk

A stochastic stability study of
"internal" vs "drastic" project changes

Internal change (twiddle current project)

- Project options
  - Search within (Min .. Max)

	ranges:	min	max	fixed	
	prec	1	2	data	3
OSP:	flex	2	5	pvol	2
Orbital	resl	1	3	rely	5
space	team	2	3	pcap	3
plane	pmat	1	4	plex	3
	stor	3	5	site	3
	ruse	2	4		
	docu	2	4		
	acap	2	3		
	pcon	2	3		
	apex	2	3		
	ltex	2	4		
	tool	2	3		
	sced	1	3		
	cplx	5	6		
	KSLOC	75	125		



Drastic change (massive project reorganization)

From Hoh Peter In And Barry Boehm, 1999

(not all drastic changes, just a sample)

Drastic change	Effects on Figure 4
1 Improve personnel	acap = 5; pcap = 5; pcon = 5 apex = 5 ; plex = 5 ; ltex = 5
2 Improve tools, techniques, or development platform	time = 3; stor = 3 pvol = 2; tool = 5 site = 6
3 Improve precedentness / development flexibility	prec = 5; flex = 5
4 Increase architectural analysis / risk resolution	resl = 5
5 Relax schedule	sced = 5
6 Improve process maturity	pmat = 5
7 Reduce functionality	data = 2; kloc * 0.5
8 Improve the team	team = 5
9 Reduce quality	rely = 1 ; docu = 1 time = 3 ; cplx = 1

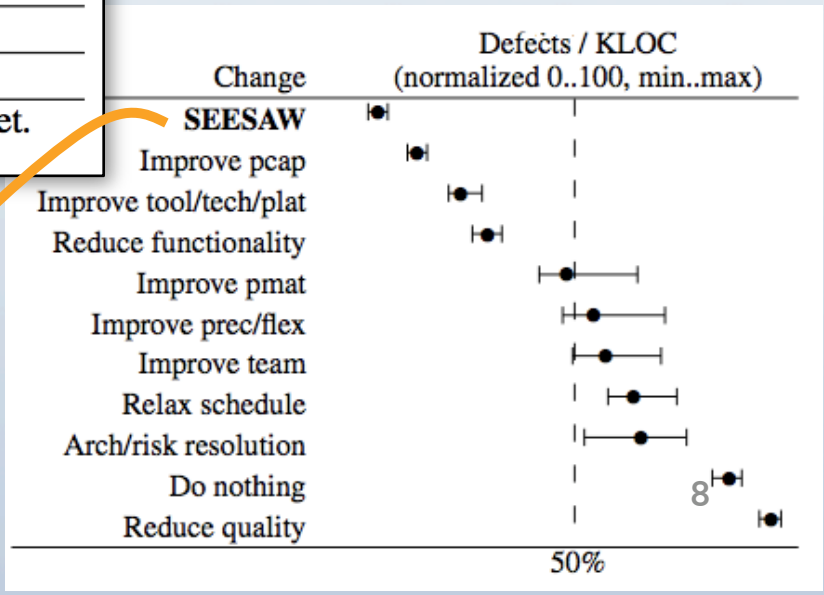
	ranges			values	
project	feature	low	high	feature	setting
JPL ground software with improved personnel	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4	**acap**	**5**
	time	3	4	**pcap**	**5**
	stor	3	4	**pcon**	**5**
	pmat	2	3	**apex**	**5**
	KSLCO	11	392	**plex**	**5**
				ltex	**5**

Drastic changes can be pretty drastic

	Drastic change	Possible undesirable impact
1	Improve personnel	Firing and re-hiring personnel leading to wide-spread union unrest.
2	Improve tools, techniques, or development platform	Changing operating systems, IDEs, coding languages
3	Improve precedentness / development flexibility	Changing the goals of the project and the development method.
4	Increase architectural analysis / risk resolution	Far more elaborate early life cycle analysis.
5	Relax schedule	Delivering the system later.
6	Improve process maturity	May be expensive in the short term.
7	Reduce functionality	Delivering less than expected.
8	Improve the team	Requires effort on team building.
9	Reduce quality	Less user approval, smaller market.

Q: can we do better than drastic change via internal changes?

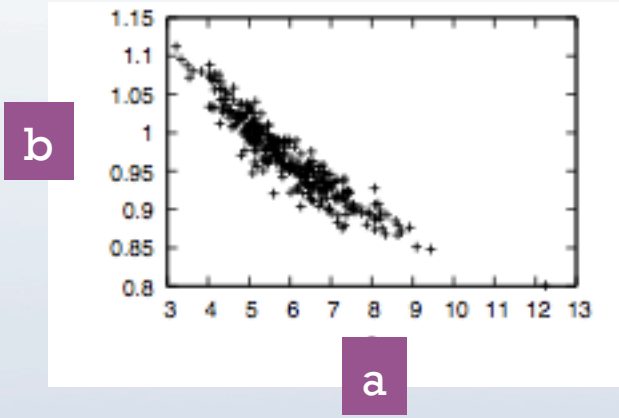
A: in many cases, yes



How to check if internal beats drastic

- Easy! (not)
  - Conduct what-if queries across process models
  - Cloud computing, overnight run, try everything!
- Problems
  - How to get models people trust?
  - How to avoid mountains of irrelevant data?
  - How to tune those models to local conditions?

- Estimate = a * loc ^{b} * stuff
  - Repeat: find <a,b> in 90% of data



- "Tuning variance" not tamed by
  - outlier removal,
  - feature selection,
  - more data collection,
  - better statistical analysis...

Need 5 things for this to work

<G, M, P, T, S>

- G = A goal function to guide the search
- M = model
- P= Project options
- T= Tuning options
- S= A search engine to explore subsets p of P

<G, M, P, T, S>
G = goal

- Goal = minimize score

$$score = \frac{\sqrt{f.M^2 + b.D^2 + c.E^2}}{\sqrt{f + b + c}}$$

- M: development months (calendar)
- D: development effort (total staff assigned)
- E: effort

<G, M, P, T, S>
M= model

- COCOMO
  - Total development effort
  - Development months
  - E.g. 4 people, 1 year then effort =48 and months = 12
- COQUALMO
  - Defects / KLOC
- **Should work for other models where**
  - project options, not tuning options, are the dominant effect on estimates

<G, M, P, T, S>
P= project options

- AI searches (Min .. Max)
- As project grow, they grow less flexible
- OSP2 < OSP < (flight, ground)
  - Flight: general description
  - OSP: orbital space plan
  - OSP2: OSP v2.

	ranges:	min	max	fixed	
	prec	1	2	data	3
OSP:	flex	2	5	pvol	2
Orbital	resl	1	3	rely	5
space	team	2	3	pcap	3
plane	pmat	1	4	plex	3
	stor	3	5	site	3
	ruse	2	4		
	docu	2	4		
	acap	2	3		
	pcon	2	3		
	apex	2	3		
	ltex	2	4		
	tool	2	3		
	sced	1	3		
	cplx	5	6		
	KSLOC	75	125		

Controllability assumption

<G, M, P, T, S>
T= tuning options

- COCOMO effort estimation
  - Effort multipliers are straight (ish) lines
  - when EM = 3 = nominal , multiple effort by one (I.e. nothing)
  - i.e. they pass through the point {3,1};

$$\forall x \in \{1..6\}\ EM_i = m_a(x-3)+1$$

$$(0.073 \le m_a^+ \le 0.21) \wedge (-0.178 \le m_a^- \le -0.078)$$

Increase effort

cplx, data, docu
pvol, rely, ruse,
stor, time

decrease effort

acap, apex, ltex, pcap,
pcon, plex, sced,
site, tool

- Similarly for scale factors (and COQUALMO)

<G, M, P, T, S>
S = search

$$\arg\max_{x} \left(\overbrace{r_x \subseteq p}^{AI\ search} , \underbrace{t \subseteq T, value(model(r_x, t))}_{Monte\ Carlo} \right)$$

- Which search engine?
  - This paper:
    - a constraint satisfaction method (a variant of MaxWalkSat)
- And many others

Results

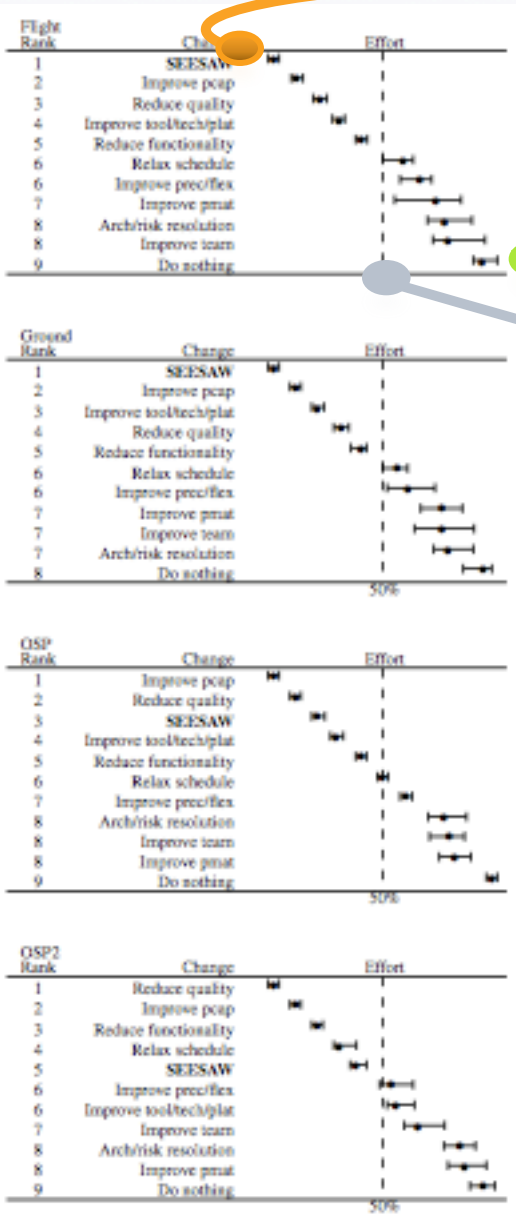


Figure 8. EFFORT: total staff months (normalized 0..100%).

Results normalized min=0 to max=100

50% percentile

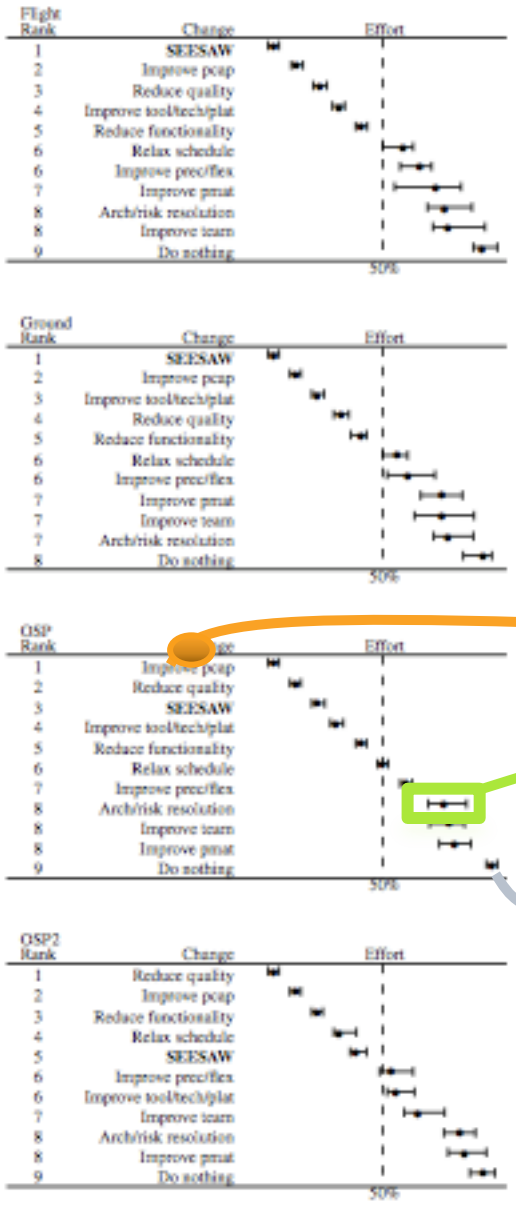


Figure 8. EFFORT: total staff months (normalized 0..100%).

Treatment

25% to 75% range

50% (median)

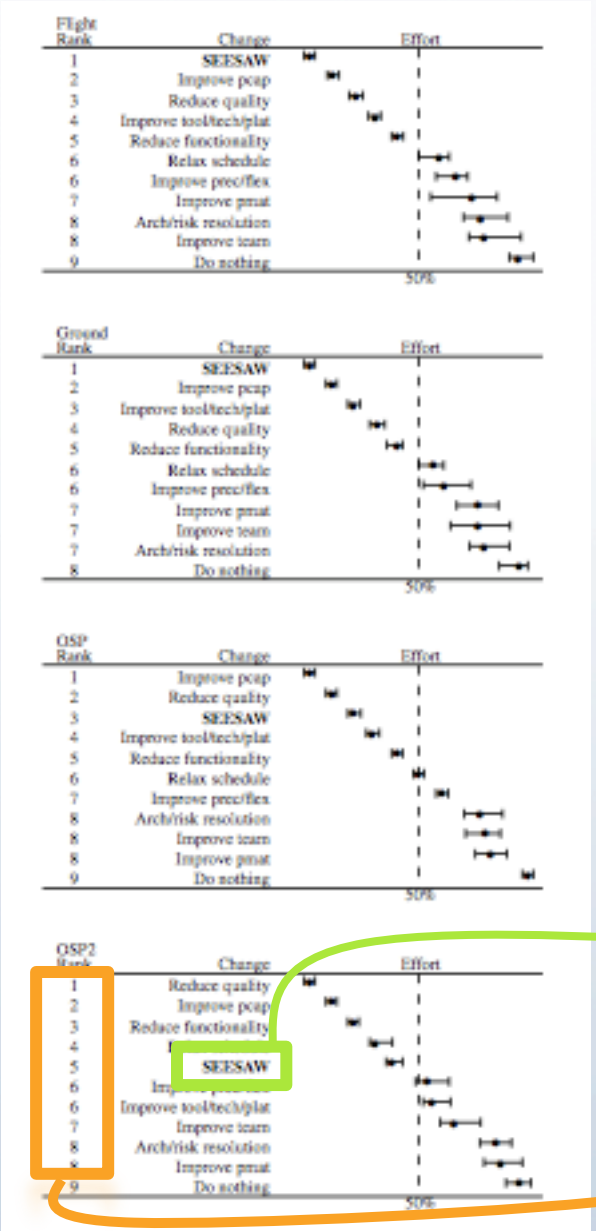


Figure 8. EFFORT: total staff months (normalized 0..100%).

Internal change

Mann-Whitney results (95% confidence)

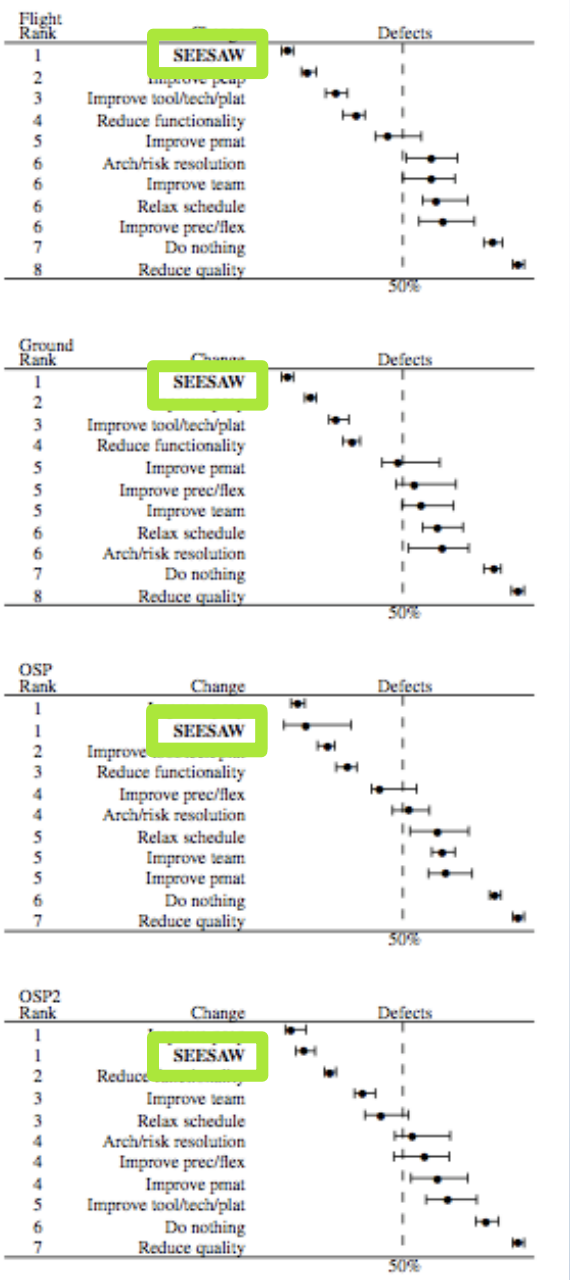


Figure 10. Defect / KLOC (normalized 0..100%).

Defect predictions after SEESAW

- Always first, or ties with first

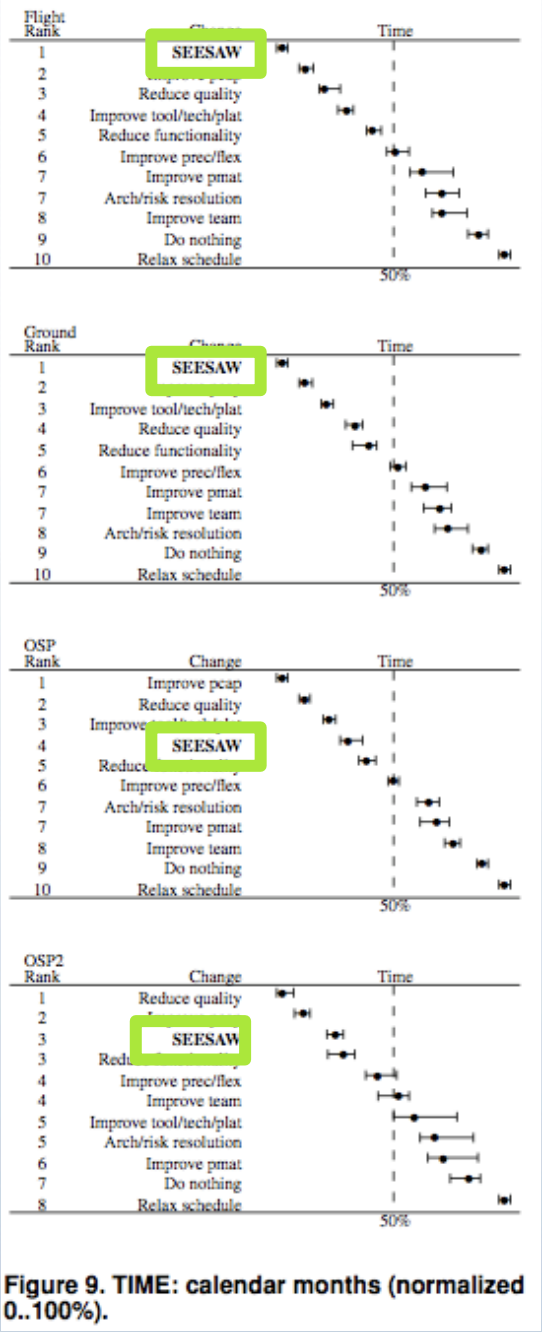


Figure 9. TIME: calendar months (normalized 0..100%).

Development Time predictions after SEESAW

- Flight= first
- Ground = first
- OSP = fourth (beaten by reduce quality, improve pcap)
- OSP2= third (beaten by reduce quality, improve pcap)

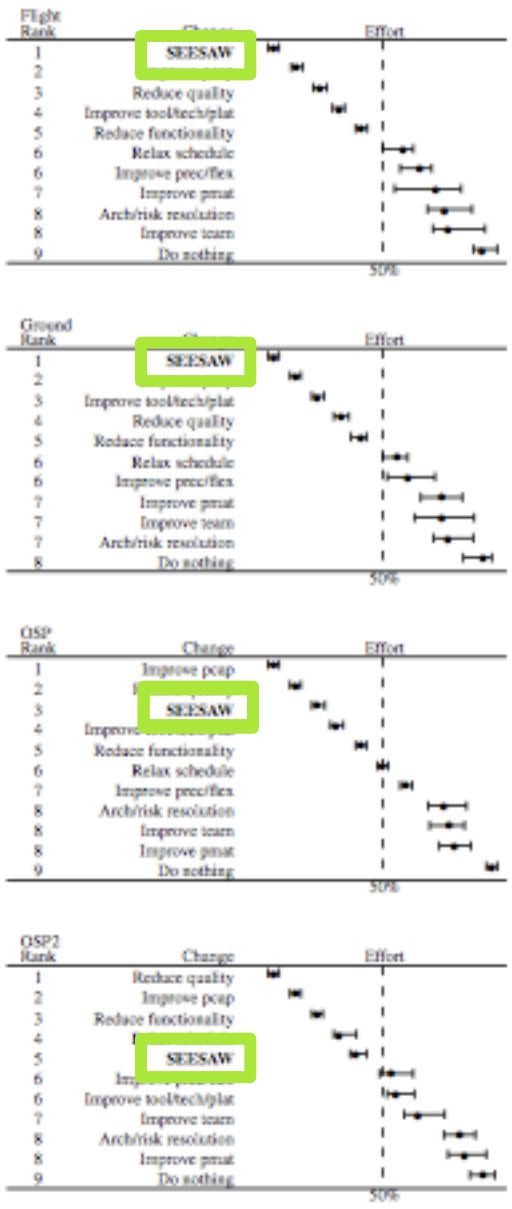


Figure 8. EFFORT: total staff months (normalized 0..100%).

Effort predictions after SEESAW

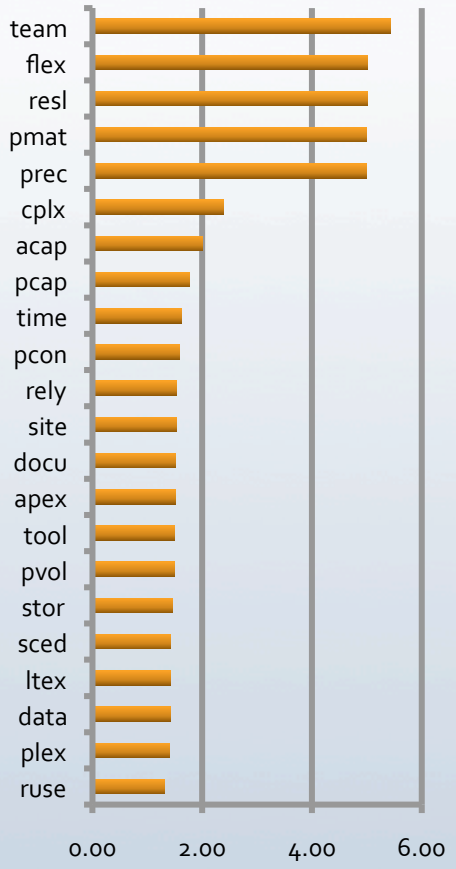
- Flight= first
- Ground = first
- OSP = third (beaten by reduce quality, improve pcap)
- OSP2 = fifth (but still in bottom half)

12 case studies

- (flight, ground, OSP,OSP2) * (effort, defect, time)
- Usually (8/12):
  - SEESAW ties for first rank
- In the remaining (3/4):
  - beaten by reducing quality, pcap changes
- Always(12/12):
  - better than at least half the others
- Usually (10/12):
  - in bottom quarter
- Worst results with OSP2

Validity

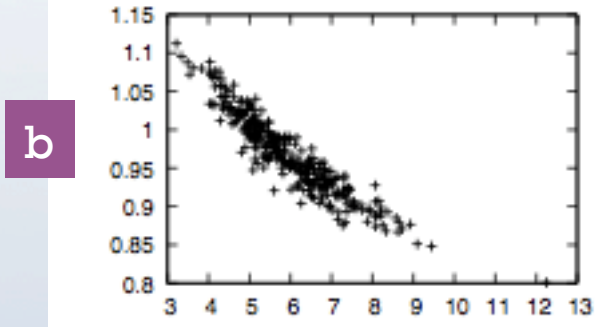
- Internal validity
  - Results stable across space of tunings
- External Validity
  - Conclusions based on one search engine
    - Active area of exploration
  - Conclusions based on COCOMO/COQUALMO
    - 28 years of active development review
  - Assumes estimates can be controlled by controlling project options,, not tunings
    - Which is true for COCOMO/COQUALMO [Menzies, Boehm, Madachy, El-waras, et al ICSP 2008]
- Construct validity
  - The change cost issue
  - Results are "estimates", not "actuals"
    - Estimates as odd as the underlying model



Relative impact above lowest value

Related Work

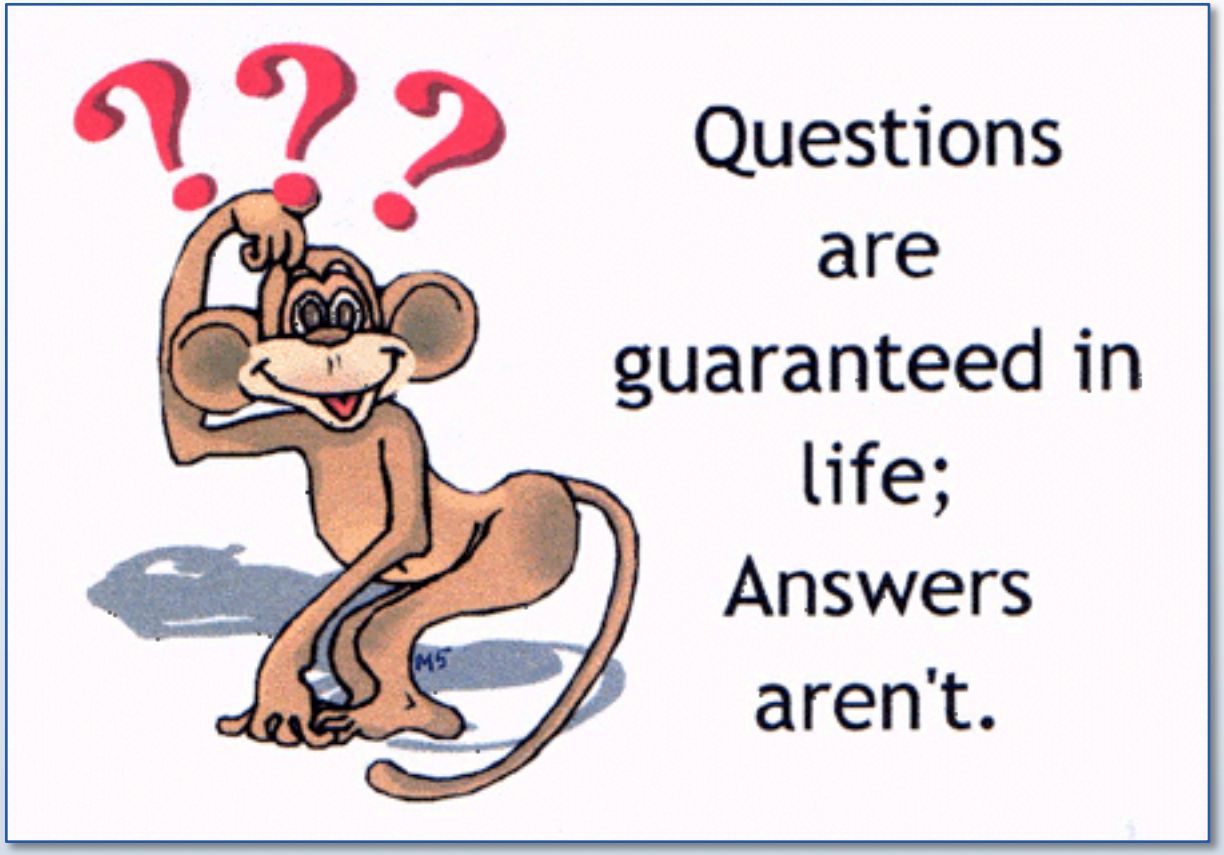
- Other work in this framework
  - Studying the effects of changing weights in goal function [Promise'09]
  - With Orrego [ICSP'09]: studying reuse
  - Submitted to [ASE'09]: ranking different search algorithms
- Uncertainty in software engineering often Bayesian; eg.
  - E.g., Pendharkar et al. [TSE'05]
  - E.g. Fenton and Neil et al [Many places, including PROMISE'07]
  - Not combinations of defect, effort, time
- Search-Based Software Engineering (SBSE) [Harmon et al]
  - e.g. simulated annealing, genetic algorithms, tabu search
- AI search algorithms
  - Integer programming, BDDs, constraint satisfaction, etc etc
- Numeric optimization
  - Gradient descent



- Variance reduction
  - Feature subset selection
  - Instance-based learning
  - Collect more data

Summary

- Drastic changes are disruptive.
  - Can we avoid them?
- Problem:
  - What –if queries over process models complicated by tuning variance
- Solution:
  - Use models where project effects dominate tuning effects
  - E.g. USC COCOMO suite
- SEESAW
  - AI searches project options
  - Each option assessed by Monte Carlo over randomly selected tuning options
- Using SEESAW:
  - Usually, internal changes are as good, or better, as drastic
- Next generation of empirical SE
  - Don't just build models,
  - But also report tricks on how to best use them


Questions
are
guaranteed in
life;
Answers
aren't.