This work was conducted at West Virginia University, University of Southern California, and the Jet Propulsion Laboratory under grants with NASA. Reference herein to any specific commercial product, process, or service by trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

The business case for automated software engineering

Tim Menzies (WVU) tim@menzies.us
Oussama Elrawas, Dan Baker (WVU)
Jairus Hihn, Martin Feather (JPL)
Ray Madachy, Barry Boehm (USC)

IEEE ASE 2007,
Atlanta, Georgia,
Nov 5, 2007

“Sociology beats Technology”?

- ICSE 2007 panel
  - Tim Lister (co-author of Peopleware”)
- “The major problems of our work are not so much technological as sociological in nature.”
- Focus less on new ASE tools and more on management / sociological factors
- E.g. More important than “software tools”
  - Any one of 1,2,3,4,5,6,7,8
  - Any two of 10,11,12,..,22
- So, is there a business case for automated software engineering?

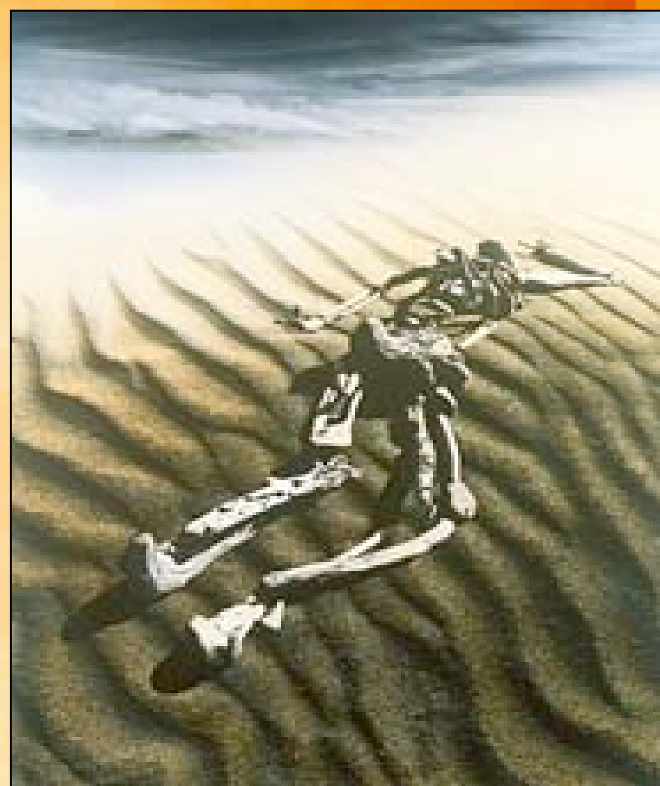
id	features	relative weight
1	Personnel/team capability	3.53
2	Product complexity	2.38
3	Time constraint	1.63
4	Required software reliability	1.54
5	Multi-site development	1.53
6	Doc. match to life cycle	1.52
7	Personnel continuity	1.51
8	Applications experience	1.51
9	Use of software tools	1.50
10	Platform volatility	1.49
11	Storage constraint	1.46
12	Process maturity	1.43
13	Language & tools experience	1.43
14	Required dev. schedule	1.43
15	Data base size	1.42
16	Platform experience	1.40
17	Arch. & risk resolution	1.39
18	Precedentedness	1.33
19	Developed for reuse	1.31
20	Team cohesion	1.29
21	Development mode	1.32
22	Development flexibility	1.26

Relative impact on development effort.
Regression analysis of 161 projects.
Boehm e.tal. 2000

So, can we ever make the case for automated software engineering?

26 inputs							3 outputs		
rely	plex	ksloc	…	pcap	time	aa	effort	schedule risk	defects
5	1	118.80	…	5	3	5	2083	69	0.50
5	1	105.51	…	1	3	5	4441	326	0.86
5	4	89.26	…	3	5	3	1242	63	0.96
5	2	89.66	…	1	4	5	2118	133	2.30
5	1	105.45	…	2	4	5	6362	170	2.66
5	3	118.43	…	2	6	2	7813	112	4.85
5	4	110.84	…	4	4	4	4449	112	6.81
…									

- Not just via final development cost
  - E.g. COCOMO II (Boehm et al 2000)
- Comment on all of {effort ,defects ,threats}; e.g.
  - COQUALMO a defect predictor (Boehm et al 2000)
  - Expert COCOMO a threat predictor (Madachy, 1994)
- Problem
  - These models need calibration
  - Calibration needs data
  - Usually, data incomplete (the “data drought”)
- Our thesis :
  - Precise tunings not required
  - Space of possible tunings is well-defined
  - Find and set the collars
    - Reveal policies that reduce effort/ defects /threats
    - That are stable across the entire space
- How will those policies rank “technology” vs “sociology”?



Road map

- Motivation
- Machinery
- Results
- Related & future work
- So what?

Q: what is "automated software engineering"
A: {automated analysis, execution-based testing} ∈ {5,6}

LOW ↔ MEDIUM ↔ HIGH

USC's software process model ontology

	Definition	Low-end = {1,2}	Medium ={3,4}	High-end= {5,6}
Defect removal features				
execution-based testing	all procedures and tools used for testing	none	basic testing at unit/ integration/ systems level; basic test data management	advanced test oralces, assertion checking, model-based testing
automated analysis	e.g. code analyzers, consistency and traceability checkers, etc	syntax checking with compiler	Compiler extensions for static code analysis, Basic requirements and design consistency, traceability checking.	formalized specification and verification, model checking, symbolic execution, pre/post condition checks
peer reviews	all peer group review activities	none	well-defined sequence of preperation, informal assignment of reviewer roles, minimal follow-up	formal roles plues extensive review checklists/ root cause analysis, continual reviews, statistical process control, user involvement integrated with life cycle
Scale factors:				
flex	development flexibility	development process rigorously defined	some guidelines, which can be relaxed	only general goals defined
pmat	process maturity	CMM level 1	CMM level 3	CMM level 5
prec	precedentedness	we have never built this kind of software before	somewhat new	thoroughly familiar
resl	architecture or risk resolution	few interfaces defined or few risk eliminated	most interfaces defined or most risks eliminated	all interfaces defined or all risks eliminated
team	team cohesion	very difficult interactions	basically co-operative	seamless interactions
Effort multipliers				
acap	analyst capability	worst 15%	55%	best 10%
aexp	applications experience	2 months	1 year	6 years
cplx	product complexity	e.g. simple read/write statements	e.g. use of simple interface widgets	e.g. performance-critical embedded systems
data	database size (DB bytes/SLOC)	10	100	1000
docu	documentation	many life-cycle phases not documented		extensive reporting for each life-cycle phase
ltex	language and tool-set experience	2 months	1 year	6 years
pcap	programmer capability	worst 15%	55%	best 10%
pcon	personnel continuity (% turnover per year)	48%	12%	3%
plex	platform experience	2 months	1 year	6 years
pvol	platform volatility $\left(\frac{frequency\ of\ major\ changes}{frequency\ of\ minor\ changes}\right)$	$\frac{12\ months}{1\ month}$	$\frac{6\ months}{2\ weeks}$	$\frac{2\ weeks}{2\ days}$
rely	required reliability	errors are slight inconvenience	errors are easily recoverable	errors can risk human life
ruse	required reuse	none	multiple program	multiple product lines
sced	dictated development schedule	deadlines moved to 75% of the original estimate	no change	deadlines moved back to 160% of original estimate
site	multi-site development	some contact: phone, mail	some email	interactive multi-media
stor	required % of available RAM	N/A	50%	95%
time	required % of available CPU	N/A	50%	95%
tool	use of software tools	edit,code,debug		integrated with life cycle

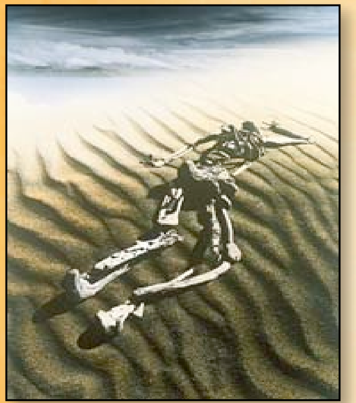
ASE :

automatedAnalysis = {5,6} or

executionBasedTestingTools = {5,6}

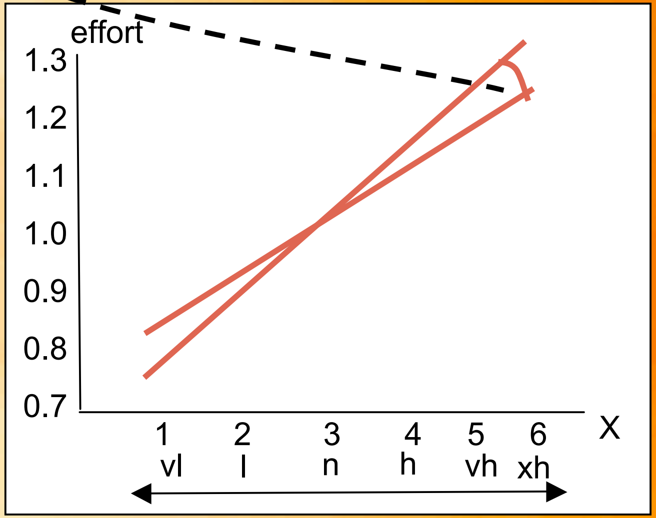
What to vary

- E.g. effort = mx + b
- Two kinds of unknowns
  - Unknowns in project ranges
    - E.g. range of “x”
  - Unknowns in internal ranges
    - E.g. range of {“m”, “b”}
- Standard practice:
  - Use history to learn internals {“m”,”b”}
  - Monte carlo over project range of “x”
  - Needs local data



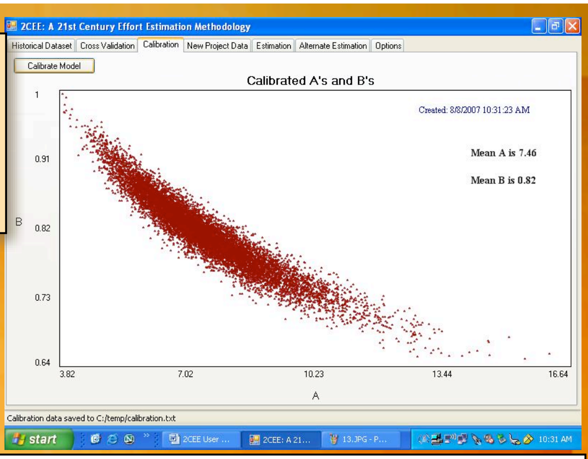
- Here: Monte carlo over the range over all of
  - { “x” , “m”, “b” }
  - No local data

	ranges			values	
project	feature	low	high	feature	setting
Flight:	rely	3	5	tool	2
	data	2	3	sced	3
	cplx	3	6		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	7	418		
Ground:	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	3	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	11	392		



Internal ranges (for effort)

92* 20*90% samples, regression to learn slope and"a" and intercept "b"



$Em_i = m_i x_i + b_i$

$Em_i = 1$ when x_i when $=3$

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

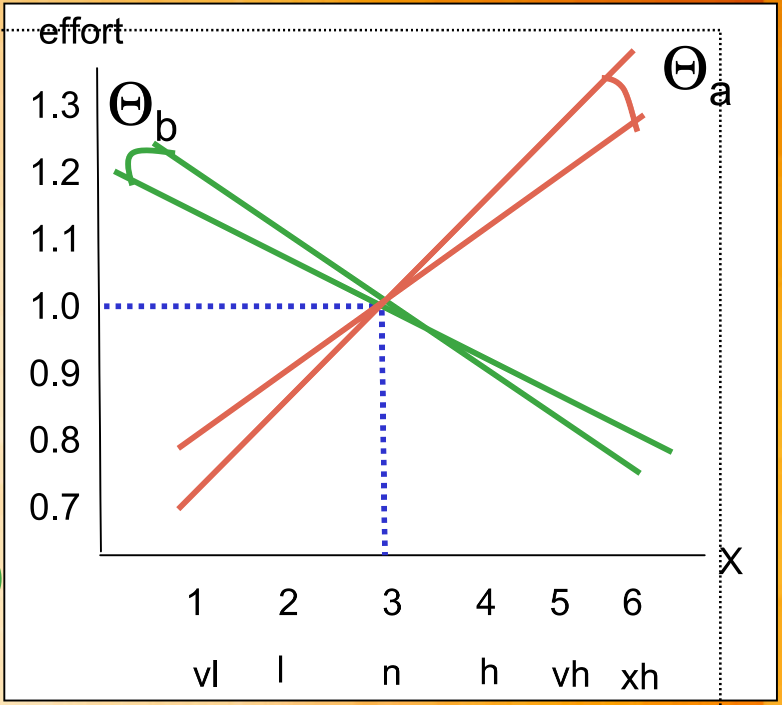
$$(0.073 \le m_a^+ \le 0.21) \wedge (-0.178 \le m_a^- \le -0.078)$$

increase effort

decrease effort

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

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



Ranges seen in 161 projects, Learned via regression, Boehm 2000

Other internal ranges (to effort and defects)

Ranges seen in 161 projects,
Learned via regression,
Boehm 2000

- COCOMO scale factors $\forall x \in \{1..6\}\ SF_i = m_b(x-6) \wedge (-1.56 \leq m_b \leq -1.014)$
- COQUALMO (defect prediction) $\forall x \in \{1..6\}\ EM_i = m_c(x-3)+1$
  - Defect introduction

$$requirements \begin{cases} 0 \leq m_c^+ \leq 0.112 \\ -0.183 \leq m_c^- \leq -0.035 \end{cases}$$

$$design \begin{cases} 0 \leq m_c^+ \leq 0.14 \\ -0.208 \leq m_c^- \leq -0.048 \end{cases}$$

$$coding \begin{cases} 0 \leq m_c^+ \leq 0.140 \\ -0.19 \leq m_c^- \leq -0.053 \end{cases}$$

e.g.increasing cplx increases introduced defects

  - Defect removal

$$\begin{array}{rl} \forall x \in \{1..6\} & SF_i = m_d(x-1) \\ requirements: & 0.08 \leq m_d \leq 0.14 \\ design: & 0.1 \leq m_d \leq 0.156 \\ coding: & 0.11 \leq m_d \leq 0.176 \end{array}$$

e.g.increasing acap decreases introduced defects

Yet more relationships (to threats)

From Madachy, KBSE 1994

- Expert COCOMO threat model:
  - Dozens of tables listing pairs of “dumb decisions”
  - E.g. very dumb to build high rely systems using constrained schedules

	rely= very low	rely= low	rely= nominal	rely= high	rely= very high
sced= very low	0	0	0	1	2
sced= low	0	0	0	0	1
sced= nominal	0	0	0	0	0
sced= high	0	0	0	0	0
sced= very high	0	0	0	0	0

wrong place to be

right place to be

- To mutate the threat model
  - Grab the “high” corner and push it “up” or pull it “down”
    - By a random factor 0.5 <=X <= 1.5

Project ranges *Internal ranges*

solution s = "({x1,x2,...} , {m1,m2,...}, {b1,b2,..})"

Find and rank solutions via simulated annealing

- Best = anything
- Run from "hot" to "cool"
  - Find something in the neighborhood of best
  - If better, then new best
  - Else
    - When "hot", maybe jump to worst
    - When "cooler", don't be so stupid
- (and as we cool, SA converges to greedy hill-climbing)
- Accumulate the total energy seen for each setting

$$0 \le E = \left(\sqrt{\overline{Ef}^2 + \overline{De}^2 + \overline{Th}^2}\right)/\sqrt{3} \le 1$$

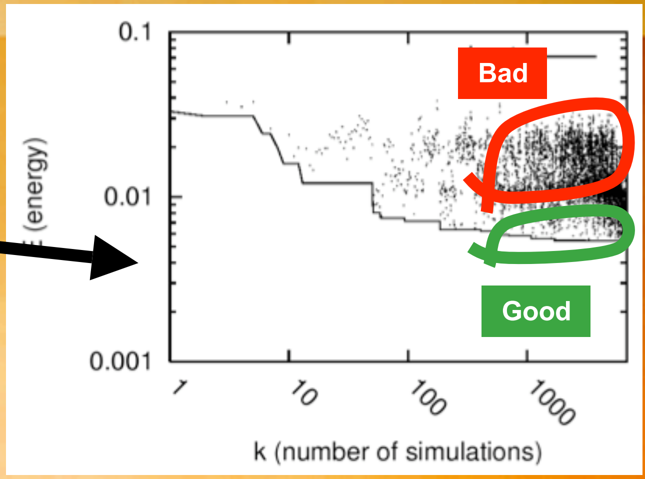
```
sample() {
    s := s0; e := E(s)      // Init state, energy.
    for t in 1 to tmax
    do
        sn = randomly change 1/3 of "s"        //   Pick neighbor.
        en := E(sn)                            //   Compute energy.
        for x in sn
            out[x] += en              // record project settings
        if P(e, en, temp(t/tmax)) > random() then  //   better?
            s := sn; e := en
    done
    return out, sorted ascending  by en
}
```

```
rank(sample());
```

```
rank(order) {
    for  I in order
    do  cache=();
        for n = 1 to 1000
        do   sn = random settings
             for j = 1 to I; do  sn[I] = order[I]; done
             cache[n]= E(sn)
        done
        sort cache
        print  cache[500], (cache[750] - cache[500])
    done
}
```

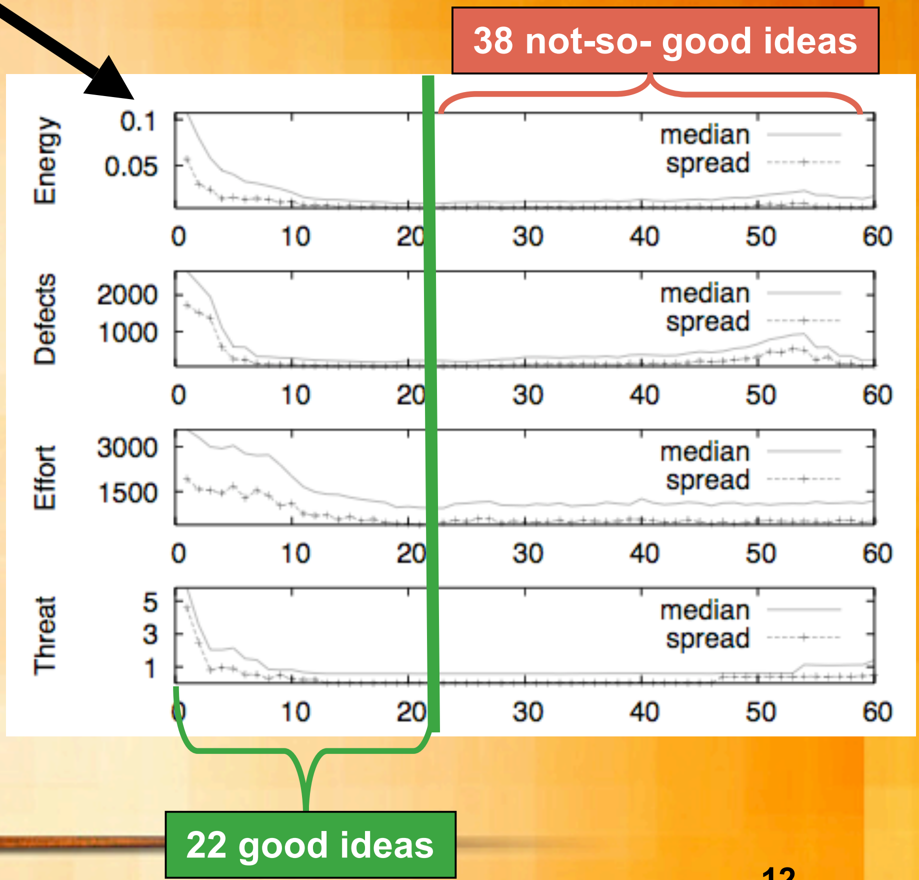
Sampling + ranking

1. sampling
- simulated annealing
2. ranking
- post-processor



$$E = \left(\sqrt{\overline{Ef}^2 + \overline{De}^2 + \overline{Th}^2} \right) / \sqrt{3}$$

- Median = 50% percentile
  - Spread = (75-50)% percentile
  - Small spread means stable across space of possible calibrations
- "Policy point" : smallest i with lowest E
- So, is automated software engineering a "good idea"?



Road map

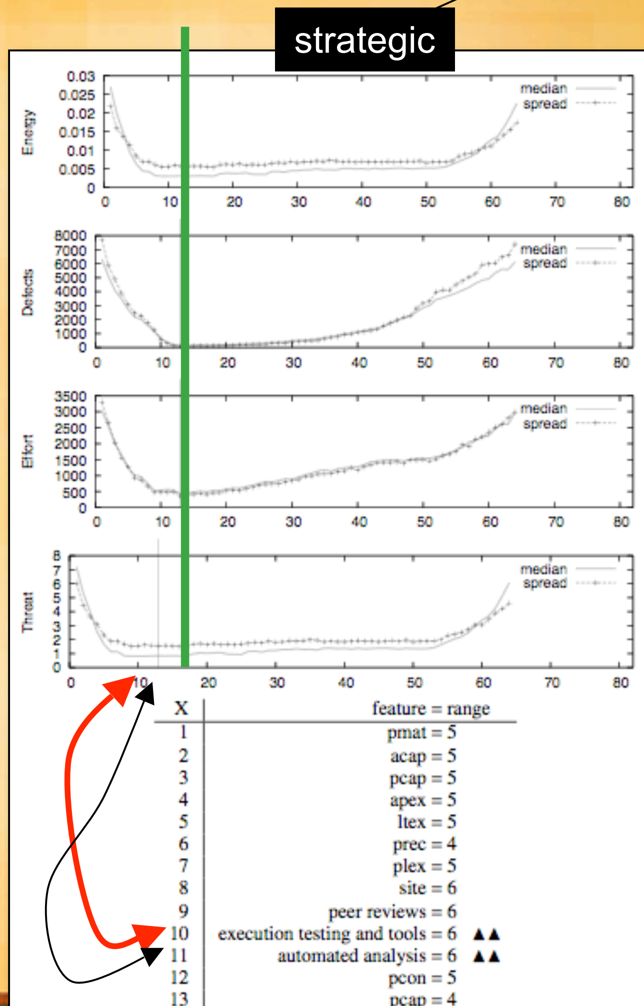
- Motivation
- Machinery
- Results
- Related & future work
- So what?

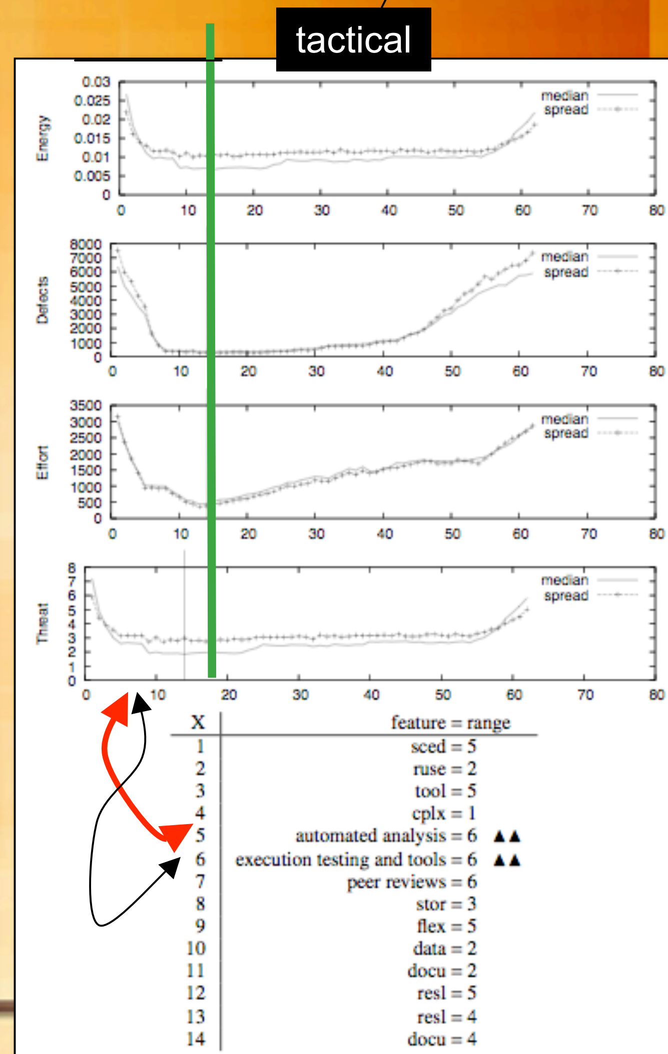
Unconstrained COCOMO (anything goes)

- KLSOC= 2 to 10,000
- All other variables, can take any value

prec pmat acap
pcap pcon apex
plex ltex site
aa ebt pr

flex resl stor
data ruse docu
tool sced cplx
aa ebt pr



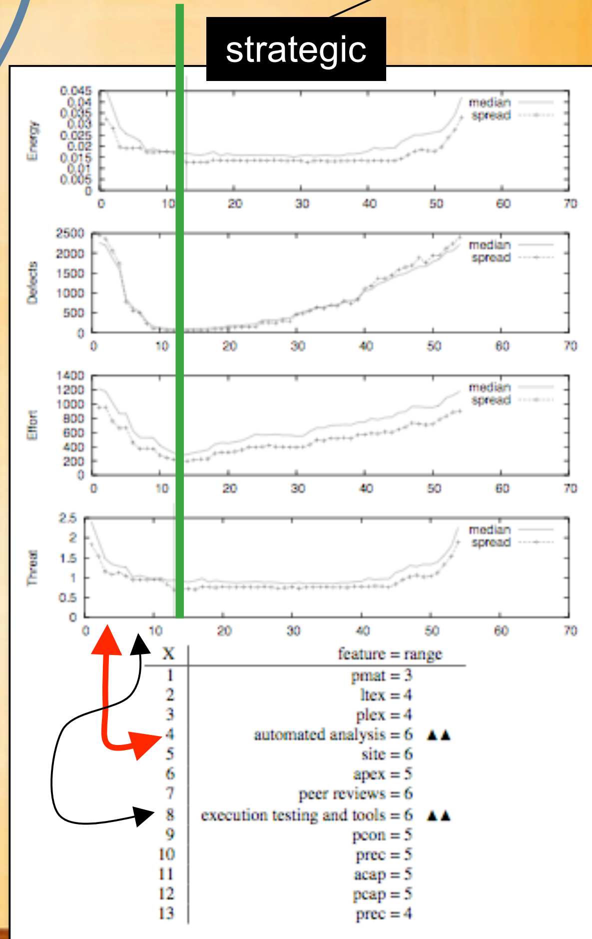


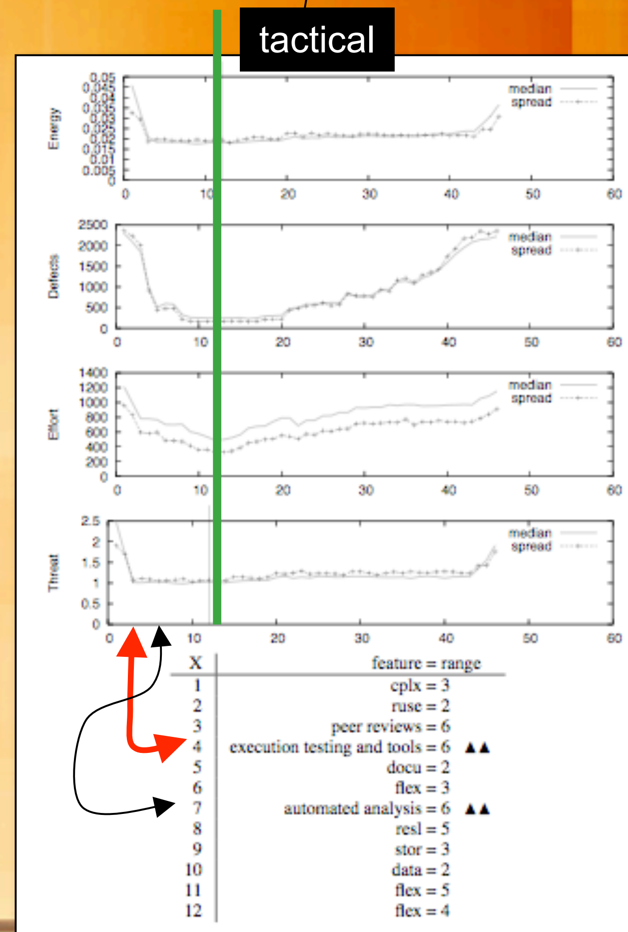
With constraints: JPL <u>flight</u> systems (GNC)

project	ranges feature	low	high	values feature	setting
Flight:	rely	3	5	tool	2
	data	2	3	sced	3
	cplx	3	6		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	7	418		
Ground:	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	3	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	11	392		

prec pmat acap
pcap pcon apex
plex ltex site
aa ebt pr

flex resl stor
data ruse docu
tool sced cplx
aa ebt pr





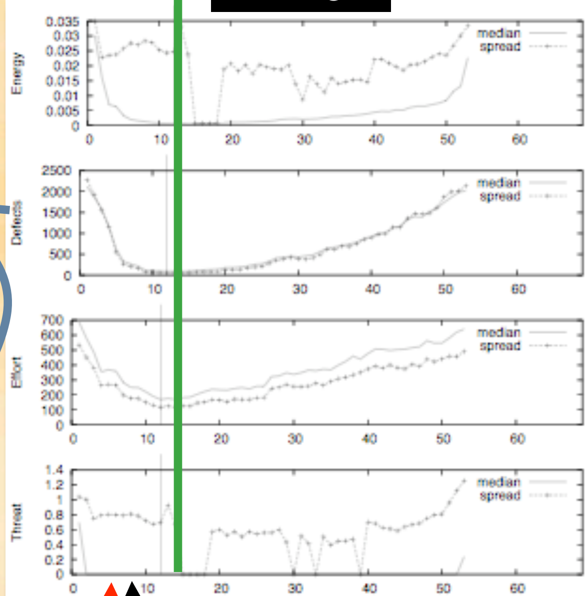
With other constraints: JPL ground systems

project	ranges: feature	low	high	values: feature	setting
Flight:	rely	3	5	tool	2
	data	2	3	sced	3
	cplx	3	6		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	7	418		
Ground:	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	3	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	11	392		

prec pmat acap
pcap pcon apex
plex ltex site
aa ebt pr

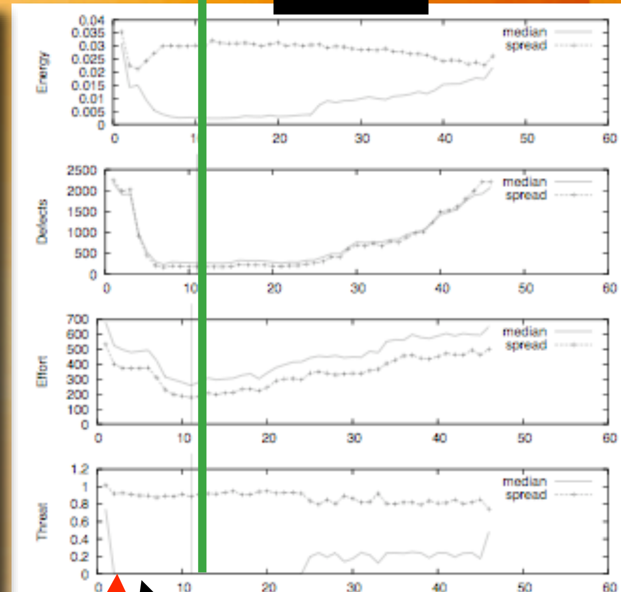
flex resl stor
data ruse docu
tool sced cplx
aa ebt pr

strategic



X	feature = range
1	ltex = 4
2	plex = 4
3	site = 6
4	peer reviews = 6
5	automated analysis = 6 ▲▲
6	pcon = 5
7	prec = 5
8	execution testing and tools = 6 ▲▲
9	apex = 5
10	acap = 5
11	pcap = 5
12	pmat = 2

tactical



X	feature = range
1	ruse = 2
2	flex = 4
3	automated analysis = 6 ▲▲
4	peer reviews = 6
5	execution testing and tools = 6 ▲▲
6	resl = 5
7	docu = 1
8	cplx = 2
9	stor = 3
10	data = 2
11	ruse = 3

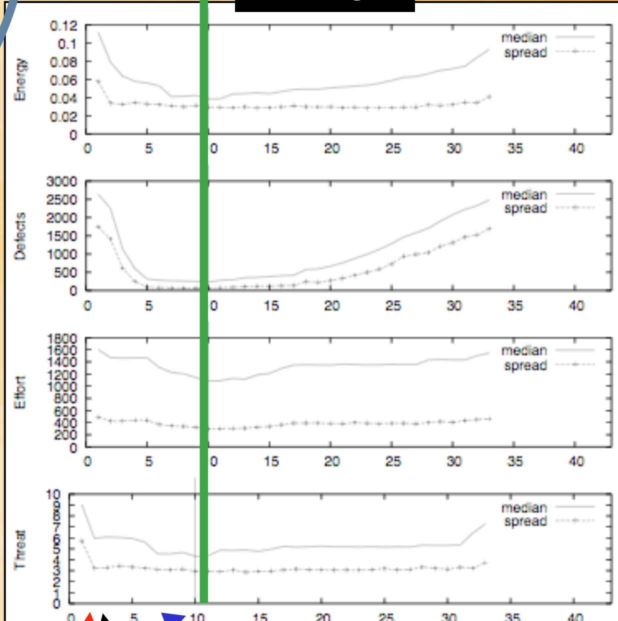
OSP: Orbital space plan (GNC)

project	ranges			values	
	feature	low	high	feature	setting
OSP: Orbital space plane	prec	1	2	data	3
	flex	2	5	pvol	2
	resl	1	3	rely	5
	team	2	3	pcap	3
	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		
OSP2	prec	3	5	flex	3
	pmat	4	5	resl	4
	docu	3	4	team	3
	ltex	2	5	time	3
	sced	2	4	stor	3
	KSLOC	75	125	data	4
				pvol	3
				ruse	4
				rely	5
				acap	4
				pcap	3
				pcon	3
				apex	4
				plex	4
				tool	5
				cplx	4

prec pmat acap
pcap pcon apex
plex ltex site
aa ebt pr

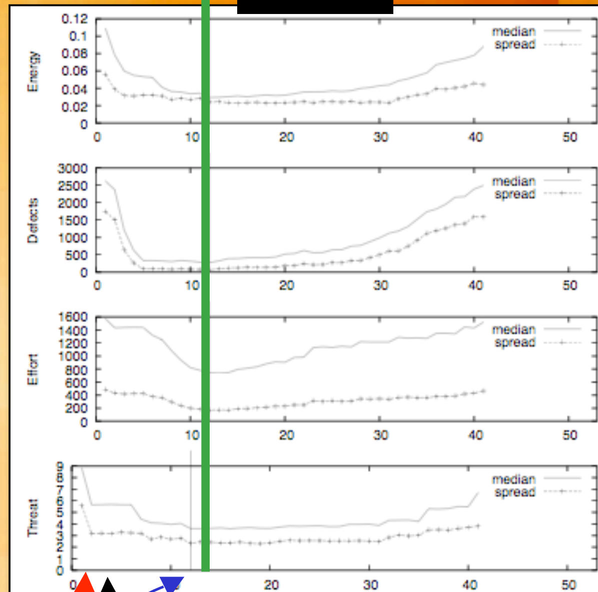
flex resl stor
data ruse docu
tool sced cplx
aa ebt pr

strategic



X	feature = range	
1	pmat = 4	
2	automated analysis = 6	▲▲
3	execution testing and tools = 6	▲▲
4	peer reviews = 6	
5	ltex = 4	
6	acap = 3	
7	prec = 2	
8	pcon = 3	
9	apex = 3	
10	execution testing and tools = 5	▲▲

tactical



X	feature = range	
1	sced = 3	
2	automated analysis = 6	▲▲
3	peer reviews = 6	
4	execution testing and tools = 6	▲▲
5	flex = 5	
6	cplx = 5	
7	ruse = 2	
8	docu = 2	
9	stor = 3	
10	resl = 3	
11	tool = 3	
12	automated analysis = 5	▲▲

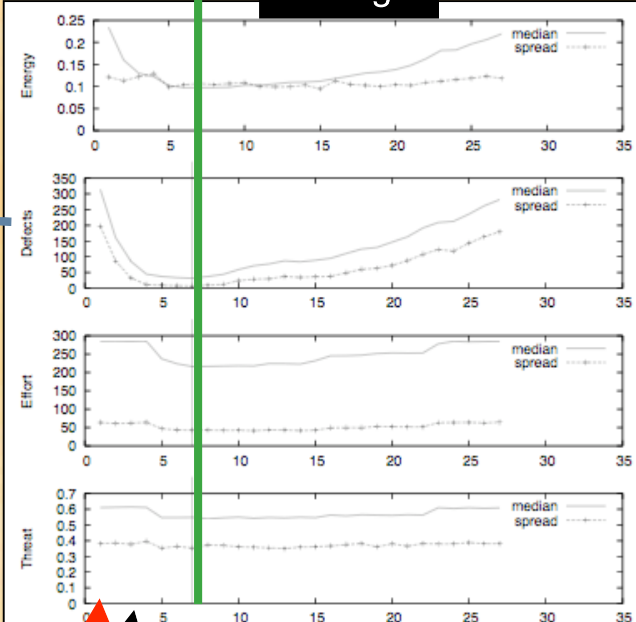
OSP2 : orbital space plane (version 2)

	ranges			values	
project	feature	low	high	feature	setting
OSP: Orbital space plane	prec	1	2	data	3
	flex	2	5	pvol	2
	resl	1	3	rely	5
	team	2	3	pcap	3
	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		
OSP2	prec	3	5	flex	3
	pmat	4	5	resl	4
	docu	3	4	team	3
	ltex	2	5	time	3
	sced	2	4	stor	3
	KSLOC	75	125	data	4
				pvol	3
				ruse	4
				rely	5
				acap	4
				pcap	3
				pcon	3
				apex	4
				plex	4
				tool	5
				cplx	4

prec pmat acap
pcap pcon apex
plex ltex site
aa ebt pr

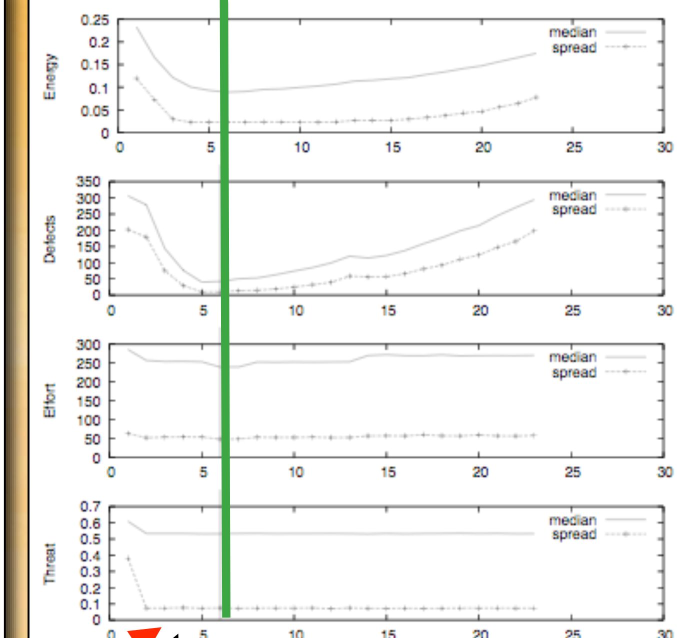
flex resl stor
data ruse docu
tool sced cplx
aa ebt pr

strategic



X	feature = range	
1	automated analysis = 6	▲▲
2	peer reviews = 6	
3	execution testing and tools = 6	▲▲
4	ltex = 5	
5	prec = 5	
6	pmat = 5	
7	peer reviews = 5	

tactical



X	feature = range	
1	sced = 4	
2	peer reviews = 6	
3	execution testing and tools = 6	▲▲
4	automated analysis = 6	▲▲
5	docu = 3	
6	peer reviews = 5	

Results (summary)

- There exists at least one oracle of software process, that says…
  - In ten case studies
  - ASE required for each minimum effort, defects, threats
  - But need to it very well, or not at all.

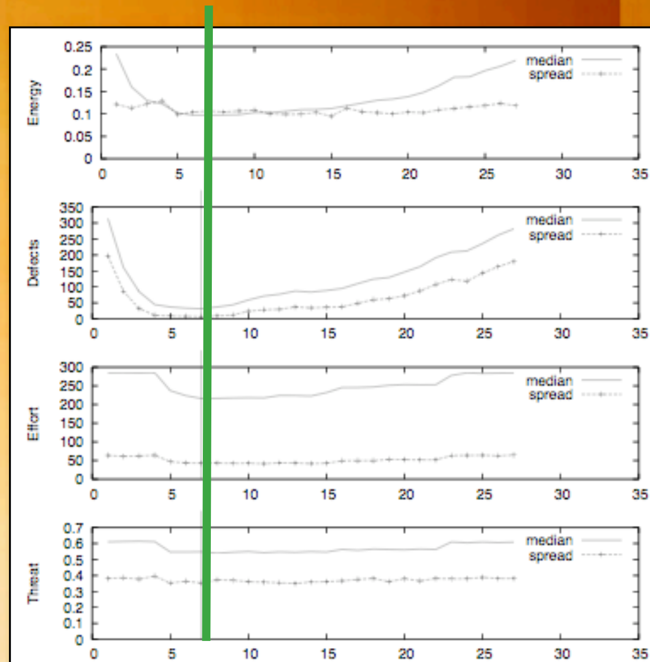
	Automated formal methods = 5	Automated formal methods = 6	Execution-based testing tools =5	Execution-based testing tools =6	Peer review =5	Peer review =6
All , strategic		♦		♦		♦
All, tactical		♦		♦		♦
Flight, strategic		♦		♦		♦
Flight, tactical		♦		♦		♦
Ground, strategic		♦		♦		♦
Ground, tactical		♦		♦		♦
OSP, strategic		♦	♦	♦		♦
OSP, tactical		♦	♦	♦		♦
OSP2, strategic		♦		♦	♦	♦
OSP2, tactical		♦		♦	♦	♦

Road map

- Motivation
- Machinery
- Results
- Related & future work
- So what?

Comparison to prior work

- COCOMO studied since 1981
  - Nothing like this in the literature
- ASE 2000: TAR1
  - TAR1 minimal contrast set learner,
  - Monte Carlo simulations of just COCOMO / THREAT models
  - Yielded one solution, not a trade space
- ASE 2002 & RE’03 : TAR2/TAR3
  - applied to other process models
- ? this work supercedes ….
  - IEEE Software 2005
    - Feature subset selection to reduce variance
  - TSE 2006 (Oct)
    - Better data mining methods to constrain model from historical data



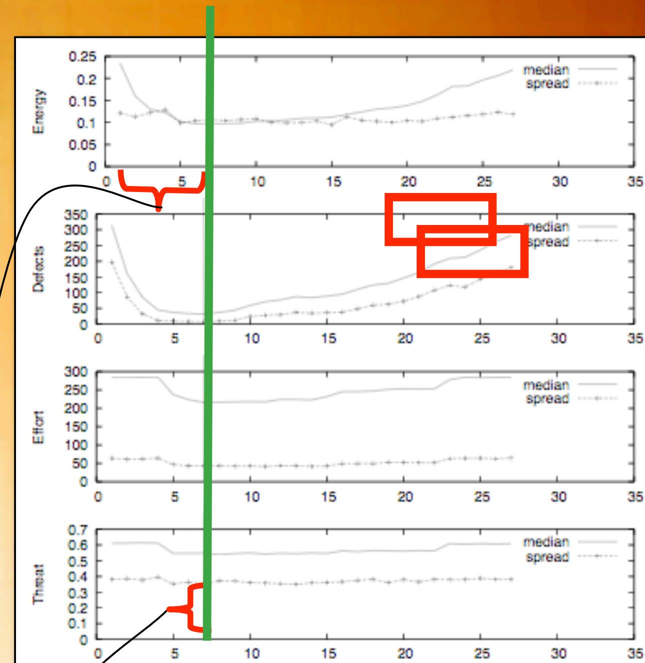
X	feature = range	
1	automated analysis = 6	▲▲
2	peer reviews = 6	
3	execution testing and tools = 6	▲▲
4	ltex = 5	
5	prec = 5	
6	pmat = 5	
7	peer reviews = 5	

Future work: checking validity

- Sampling bias
  - Our conclusions not biased by local data.
- Model bias
  - Are the USC models correct?
- Evaluation bias

$$E = \left(\sqrt{\overline{Ef}^2 + \overline{De}^2 + \overline{Th}^2}\right) / \sqrt{3}$$

  - Are defects as important as effort as threats?
- Search bias
  - Did we define ASE correctly?
  - Did we define “strategic” and ‘tactical” correctly?
  - Are the back select orderings the “best” orderings?
    - A “better” ordering would reach min energy with fewer settings and have lower spread at min energy



X	feature = range
1	automated analysis = 6 ▲▲
2	peer reviews = 6
3	execution testing and tools = 6 ▲▲
4	ltex = 5
5	prec = 5
6	pmat = 5
7	peer reviews = 5

prec	pmat	acap
pcap	pcon	apex
plex	ltex	site
aa	**ebt**	**pr**

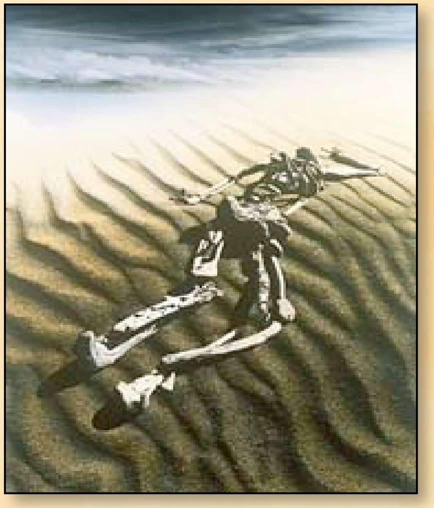
flex	resl	stor
data	ruse	docu
tool	sced	cplx

Road map

- Motivation
- Machinery
- Results
- Related & future work
- So what?

Conclusion:
does "Sociology beat Technology"?

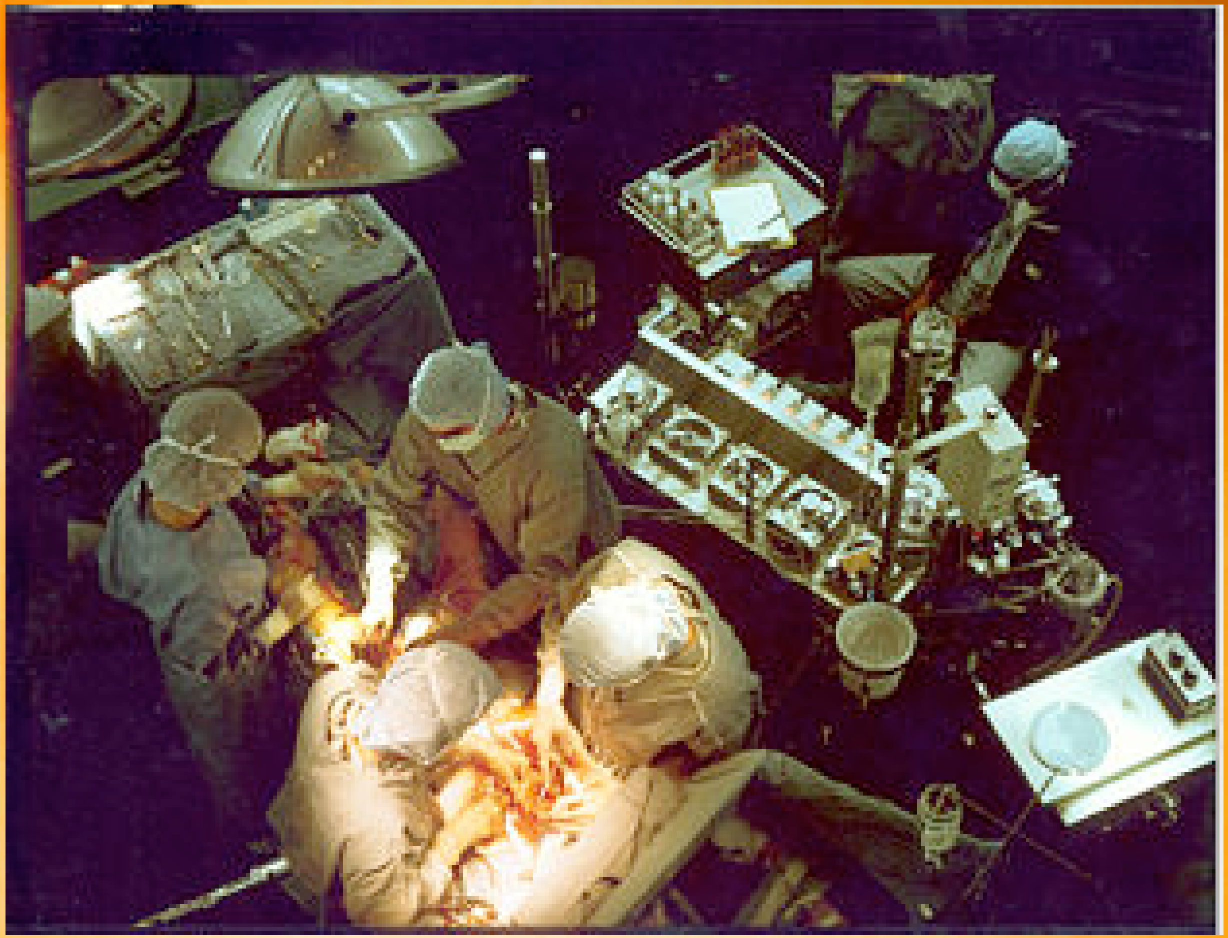
- No
  - Technology compliments sociology.
- There exists at least one oracle that says:
  - "Technology" (a.k.a. ASE) is an essential tool for reaching minimum {effort, defects, threats}
    - 10 case studies using USC models.
    - Problem of local calibration avoided with some AI search



- But:
  - Do ASE well, or not at all
    - No halfway measures
  - ASE could only reach min {effort, defects, threats} in conjunction with sociological decisions
    - e.g. about peer reviews, process maturity,schedule pressure
  - Hence researchers need to understand both
    - software development technology
    - and the sociological factors

pulmonary

cardiac



hematology

immunology

Questions?

Comments?

Extra Material

At the “policy point”, STAR’s random solutions are surprisingly accurate

LC : learn *impact[i]* via regression (JPL data)
STAR: no tuning, randomly pick *impact[i]*

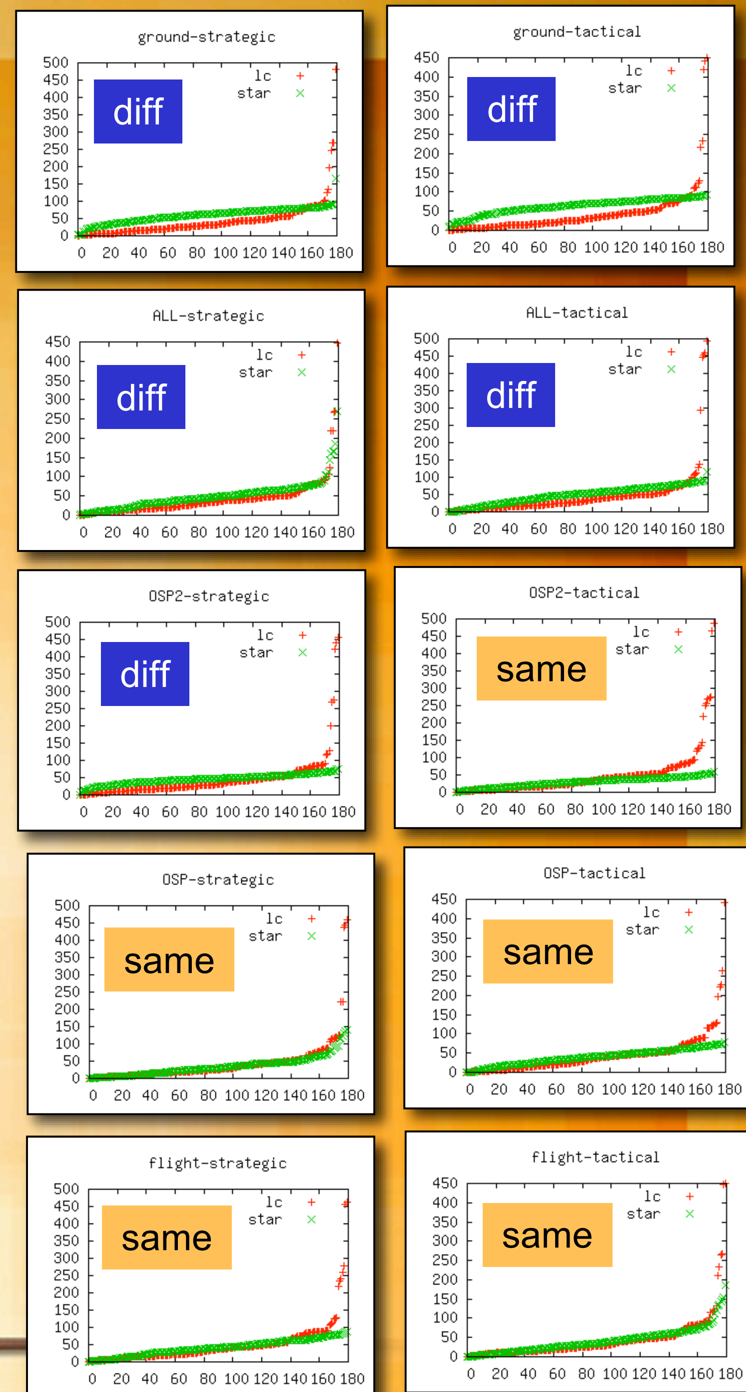
Diff = ∑ mre(lc)/ ∑ mre(star)
Mre = abs(predicted - actual) /actual

∑ mre(lc) / ∑ mre(star)	strategic	tactical
ground	66%	63%
all	91%	75%
OSP2	99%	125% ●○
OSP	112% ●○	111% ●○
flight	101% ●○	121% ●○

{“●” “○”} same at {95, 99}% confidence (MWU)

Why so little *Diff* (median= 75%)?

- Most influential inputs tightly constrained



Possible optimizations (not used here)

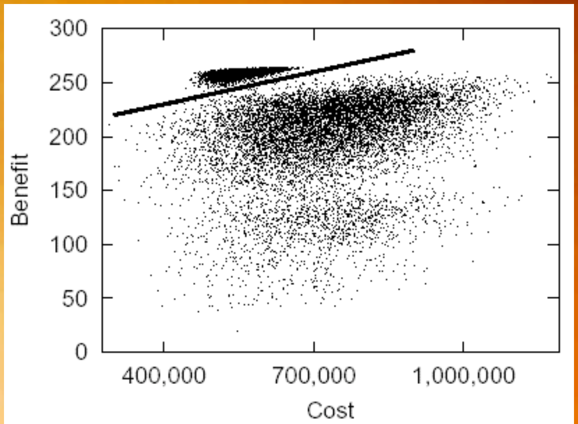
- STAR, an example of a general process:
  - Stochastic sampling
  - Sort settings by "value"
  - Rule generation experiments
    - favoring highly "value"-ed settings
    - See also, elite sampling in the cross-entropy method
- If SA convergence too slow
  - Try moving back select into the SA;
  - Constrain solution mutation to prefer highly "value"-ed settings

- BORE (best or rest)
  - n runs
  - Best= top 10% scores
  - Rest = remaining 90%
  - {a,b} = frequency of discretized range in {best, rest
  - Sort settings by $-1 * (a/n)^2 / (a/n + b/n)$

Ask me why, off-line

- Other valuable tricks:
  - Incremental discretization: Gama&Pinto's PID + Fayyad&Irani
  - Limited discrepancy search: Harvey&Ginsberg
  - Treatment learning: Menzies&Yu

Related work



- Abduction :
  - World *W* = minimal set of assumptions (w.r.t. size) such that
    - *T ∪ A => G*
    - *Not(T U A => error)*
  - Framework for
    - validation,
    - diagnosis,
    - planning,
    - monitoring,
    - explanation,
    - tutoring,
    - test case generation,
    - prediction,…
  - Theoretically slow (NP-hard) but this should be practical:
    - Abduction + stochastic sampling
    - Find collars
    - Learn constraints on collars

- Feather, DDP, treatment learning
  - Optimization of requirement models

- XEROC PARC, 1980s, qualitative representations (QR)
  - not overly-specific,
  - Quickly collected in a new domain.
  - Used for model diagnosis and repair
  - Can found creative solutions in larger space of possible qualitative behaviors,
    - than in the tighter space of precise quantitative behaviors

“Collar” variables set the other variables

- IEEE Computer, Jan 2007: “The strangest thing about software”
  - Narrows (Amarel 60s)
  - Minimal environments (DeKleer ‘85)
  - Master variables (Crawford & Baker ‘94)
  - Feature subset selection (Kohavi ‘97)
  - Back doors (Williams 03)
  - Etc
- Simpler reasoning, under uncertainty
- ASE 2000, ASE 2002
  - Rule generation & collars
- ASE 2007
  - Simulated annealing & collars
- ASE 2008
  - QFDs & collars
  - Genetic algorithms & collars

trust the force,
(collars)

IEEE Computer, Jan 2007
"The strangest thing about software"

- "Collar" variables set the other variables
  - Narrows (Amarel 60s)
  - Minimal environments (DeKleer '85)
  - Master variables (Crawford & Baker '94)
  - Feature subset selection (Kohavi '97)
  - Back doors (Williams 03)
  - Etc
- Simpler reasoning, under uncertainty
- ASE 2000, ASE 2002
  - Rule generation & collars
- ASE 2007
  - Simulated annealing & collars
- ASE 2008
  - QFDs & collars
  - Genetic algorithms & collars