Managing Uncertainty in Value-based SE

Tim Menzies (tim@menzies.us)
Phillip Green,
Oussama Elwaras

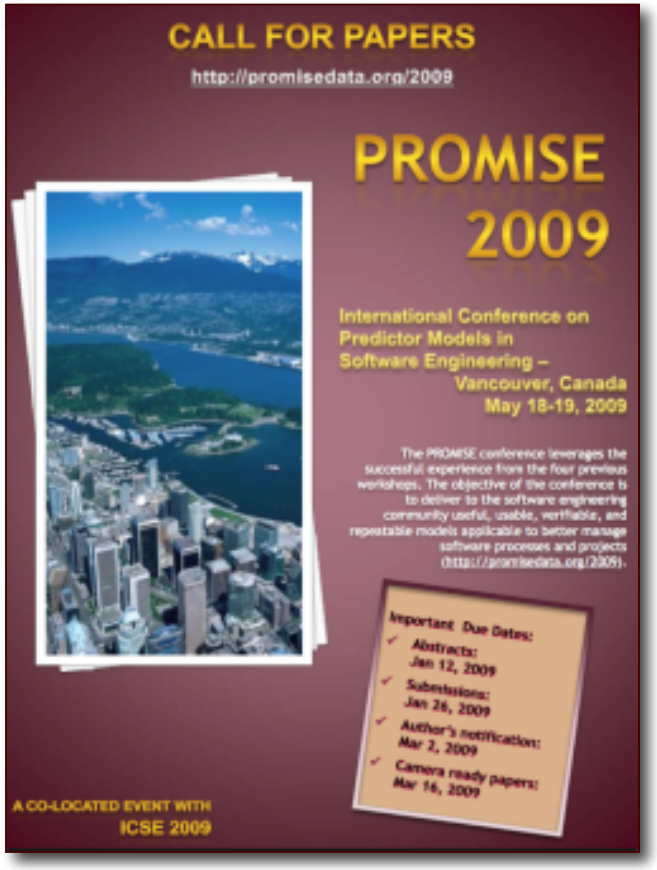
West Virginia University.

10/27/08
23rd International Forum on
COCOMO and Systems/Software Cost Modeling

Sound bites

- Come to PROMISE '09
- Value-based SE:
  - not even wrong?
- Data drought leading to conclusion uncertainty
  - Seek stability over samples
- On sampling some systems, we see
  1. Value does not take more time
  2. Value takes more effort
  3. Value (<u>is</u> , <u>isn't</u>) harder to control
  4. More value = more defects
- Community challenge:
  - when does 1,2,3,4 hold?

PROMISE '09



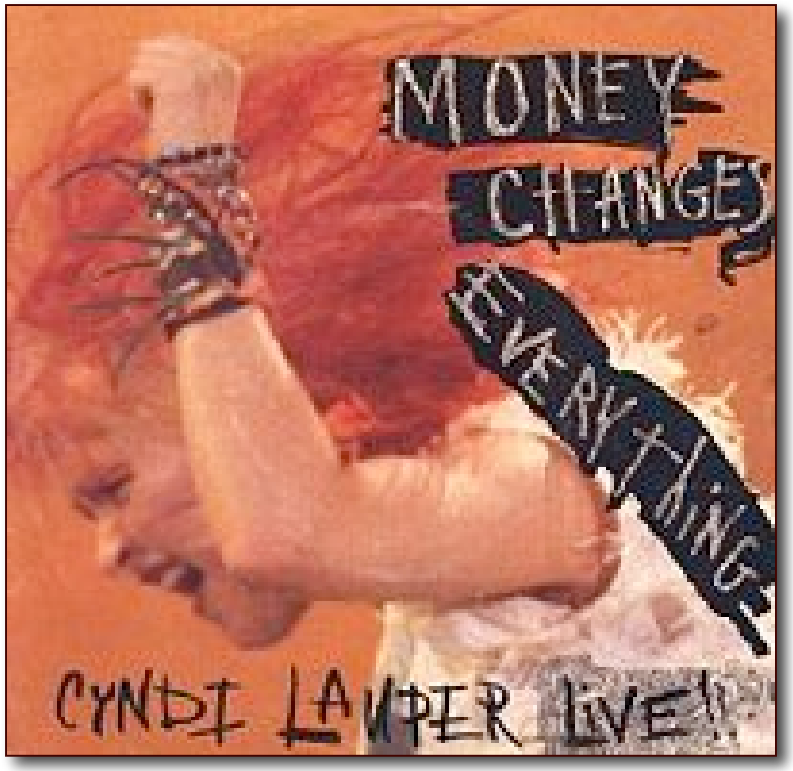
- www.promisedata.org/2009
- Reproducible SE results
- Papers:
  - and the data used to generate those papers
  - www.promisedata.org/data
- Keynote speaker:
  - Barry Boehm, USC
- Motto:
  - Repeatable, refutable, improvable
  - Put up or shut up

Value-based Software Engineering

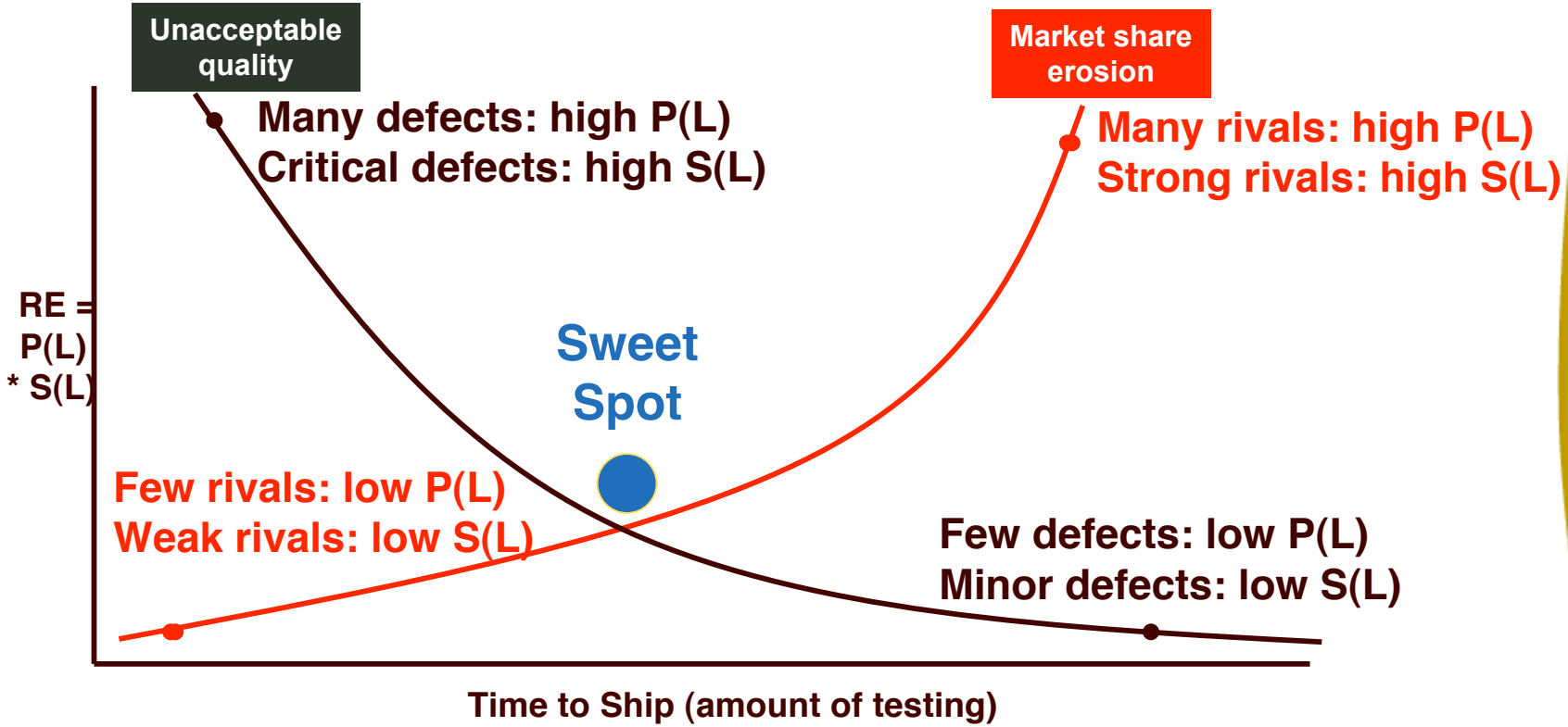
The future of SE?

Thesis: value changes everything!

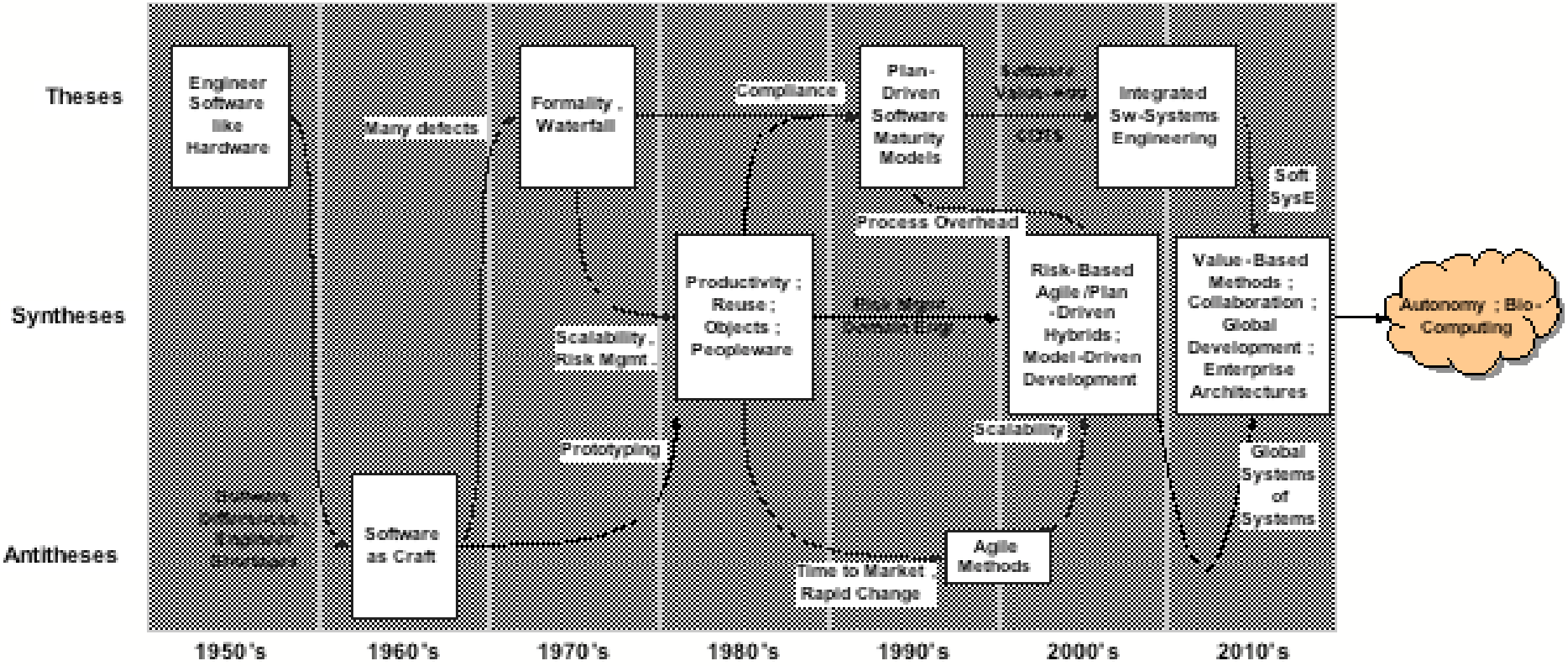
- Q: what is SE
  - A: The application of science and mathematics by which the properties of software are made <u>useful to people</u>
- Most SE techniques are "value-neutral"
  - Boehm, ASE 2004
  - Euphuism for "useless"?
- Value-based SE makes a difference
  - Yeah? Really?



Risk Exposure RE
= Prob (Loss) * Size (Loss)



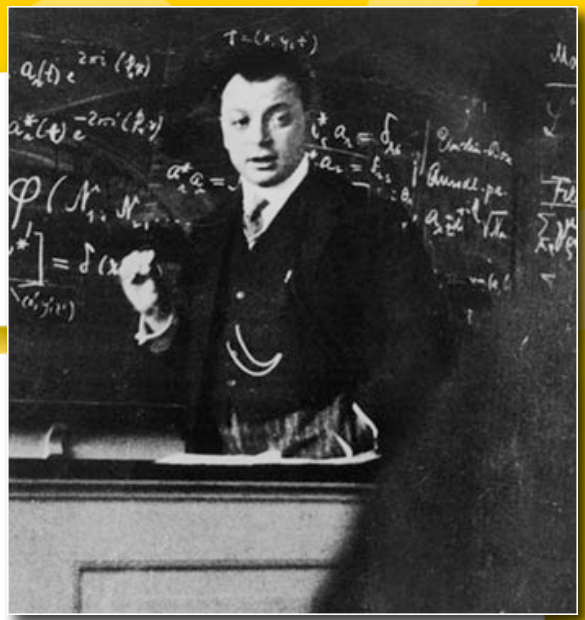
The History of Computing Naturally Leads to Value-based SE



Value-based SE

Not even false?

Is the value-thesis not even wrong?



- Wolfgang Pauli
- The "conscience of physics",
  - the critic to whom his colleagues were accountable.
- Scathing in his dismissal of poor theories
  - often labeling it *ganz falsch*, utterly false.
- But *"ganz falsch"* was not his most severe criticism,
  - He hated theories so unclearly presented as to be
    - untestable
    - unevaluatable,
  - Worse than wrong because they could not be proven wrong.
  - Not properly belonging within the realm of science,
    - even though posing as such.
  - Famously, he wrote of of such unclear paper:
    - "This paper is right. It is not *even* wrong."

So is the value thesis refutable?

- Find a domain general "value" proposition
  - Menzies, Boehm, Madachy Hihn, et al, [ASE 2007]
  - Reduce effort, defects, schedule
  - "energy"
- Find a local value proposition
  - A variant of USC Ph.D. thesis
    - [Huang 2006]: Software Quality Analysis: a Value-Based Approach
  - "value"
- Use them in a what-if scenario
- Any difference in the conclusions?

```
(defun unnormalized-energy ()
 "Calculates unnormalized energy."
 (let* ((effort   (effort))
        (months   (months effort))
        (defects  (defects))
        (threat   (threat))
         (neffort  (normalize 'effort  effort))
        (nmonths  (normalize 'months  months))
        (ndefects (normalize 'defects defects))
        (nthreat  (if (< threat 5) 0 (normalize 'threat threat))))
   (sqrt (+                  (expt (* neffort  (effort-weight)) 2)
                             (expt (* nmonths  (months-weight)) 2)
                             (expt (* ndefects (defect-weight)) 2)
                             (expt (* nthreat  (threat-weight)) 2)))))

(defun effort-weight () 1)
(defun months-weight ()  1)
(defun defect-weight ()  (+ 1 (expt *rely-defect* (- (em-range (! 'rely)) 3))))
(defun threat-weight () 1)
```

```
(defun curve-size (attribute)  (expt 0.5 (1- (rating? (! attribute)))))
(defun curve-market (attribute)   (- 1 (curve-size attribute)))
(defun size-coefficient ()  (* (curve-size 'rely)))
(defun market-coefficient ()  (* (curve-market 'rely)))
(defun market-erosion-risk-exposure ()   (* (effort)   (market-coefficient)))
(defun loss-size ()
  (* (expt 3 (/ (- (rating? (! 'cplx)) 3) 2) ) (effort)
     (size-coefficient)))
(defun sofware-quality-risk-exposure () (* (loss-probability)  (loss-size)))
(defun risk-exposure ()   (+ (market-erosion-risk-exposure)
     (sofware-quality-risk-exposure)))
```

Aside

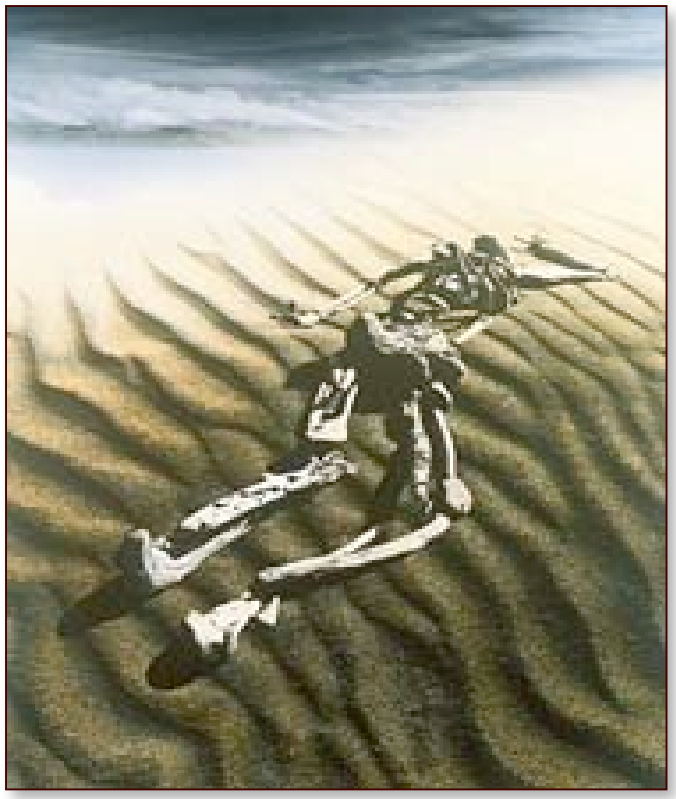
- Note really [Huang06]
  - But some variant Huang06
- Had to use some “engineering judgment”
  - a.k.a. guesses
- Apologies to Dr. Huang

Tools

- Four USC models
  - COCOMO effort prediction: staff months
  - COCOMO schedule predictor: calendar months
  - COQUALMO defect predictor: defects/KLOC
  - THREATS: “how many dumb things are you doing right now?”
- Monte Carlo simulator
- AI search engine
  - Search for the least number of project changes …
  - … that most improves the “target”
  - “Target” is either
    - [Ase07]’s “energy” function
    - [Huang06]’s “value” proposition

Problem: local tuning

- Problem
  - 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 months
    - That are stable across the entire space



The details

Using AI to find stable conclusions
in a space of options

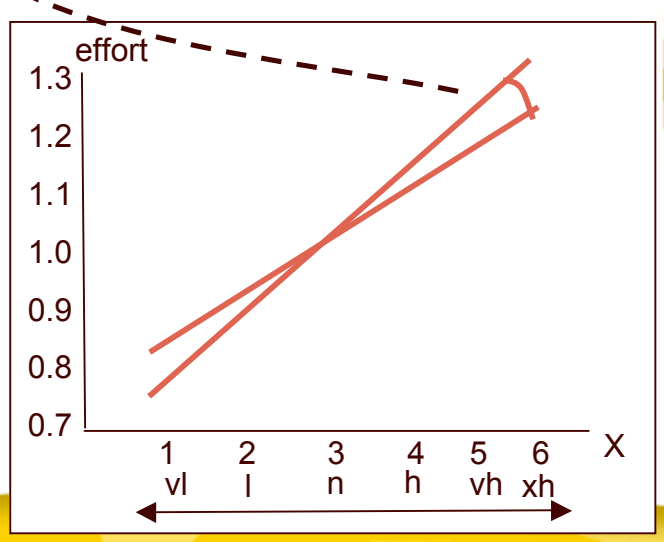
Run Delphi Sessions to Gather Project Ranges (e.g. ICSE 2008)

- Target application picked
  - A mission critical, real-time system;
  - Built by contractors (not in-house)
  - That has an operational life of 5 to 10 years (since have invested much effort into a mission critical system, an organization is most likely to use it for many years to come).
- For each COCOMO input variable
  - Boehm defines each variable
  - 5 minutes “open comments”
  - Vote. Record majority view

Sampling

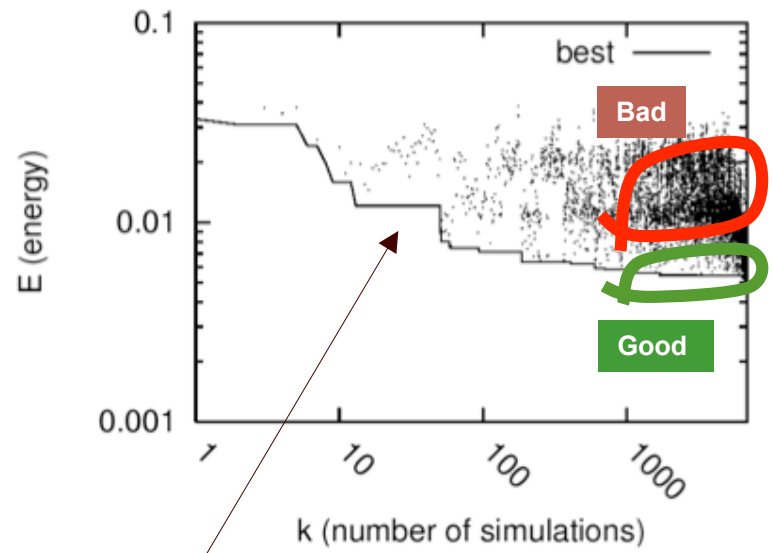
- E.g. effort = mx + b
- Two kinds of unknowns
  - Unknowns in project ranges
    - E.g. range of “x”
  - Unknowns in internal ranges
    - E.g. range of {“m”, “b”}
- Standard practice:
  - Use historical data to constrain {“m”,”b”}
- Here: Monte Carlo over range of { “x” , “m”, “b” }
  - Learn values for “x” that reduce effort
  - As a side-effect, reduce variance
  - Not need for tuning 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		



Search for stable conclusions

- Using simulated annealing, Monte Carlo simulated annealing across intersection of
  - A particular project type
  - Space of possible tunings
- Rank options by frequency in **good**, not **bad**
- For r options
  - Try setting the 1 ≤ x ≤ R top ranked options
  - Simulate (100 times) to check the effect of options 1 .. x
- Smile if
  - Reduced median and variance in defects/ efforts/ time/ threats

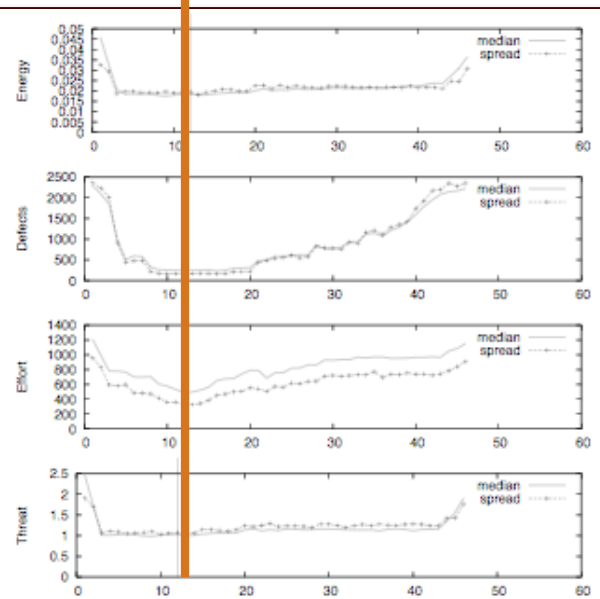


Sample run
(after 10,000 runs, little improvement)

JPL flight systems (GNC)

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

project	feature	ranges low	ranges high	values feature	values 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		

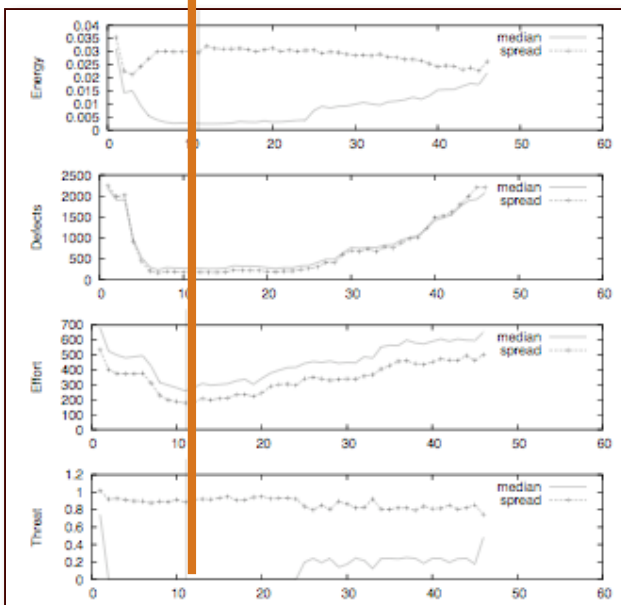


X	feature = range
1	cplx = 3
2	ruse = 2
3	peer reviews = 6
4	execution testing and tools = 6 ▲▲
5	docu = 2
6	flex = 3
7	automated analysis = 6 ▲▲
8	resl = 5
9	stor = 3
10	data = 2
11	flex = 5
12	flex = 4

JPL ground systems (GNC)

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

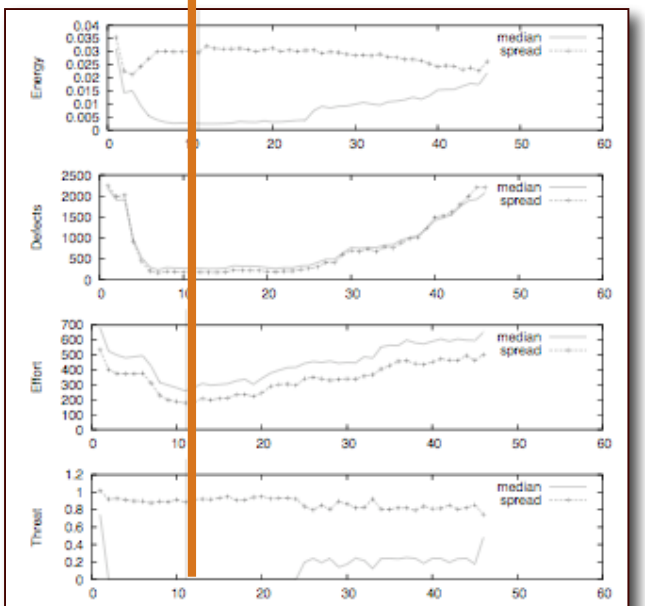
project	ranges feature	low	high	values feature	setting
Flight:	rely	3	5	tool	2
	data	2	3	sced	3
	cplx	3	6		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	Ksloc	7	418		
Ground:	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4		
	time	3	4		
	stor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



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

Assessment criteria

- Minimal values found for:
  - Defects
  - Months
  - Effort
- Number of decisions required to find those minimums
  - In this case, 10 (ruse appears twice)



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

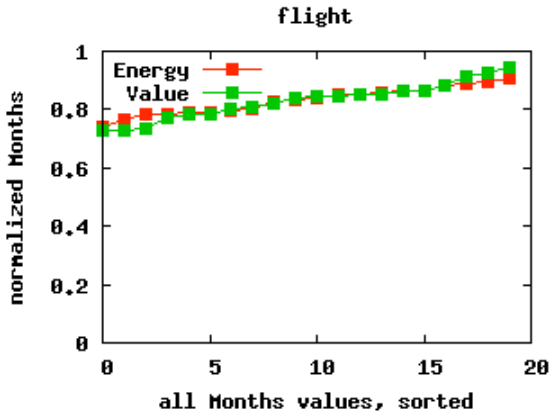
Results

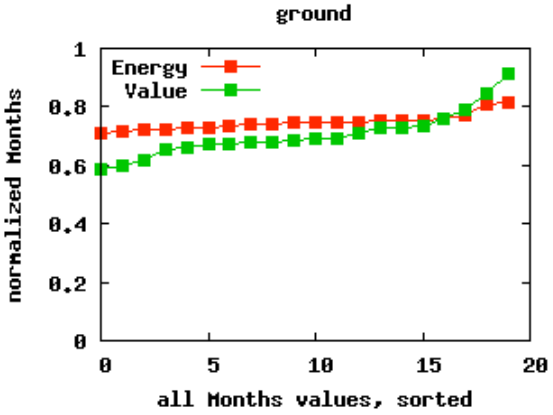
And the winner is…

Value does not take more time

- Months = calendar time
- Results from 20 trials
  - Normalized min..max = 0 .. 100
- Good news

  - Tell the world

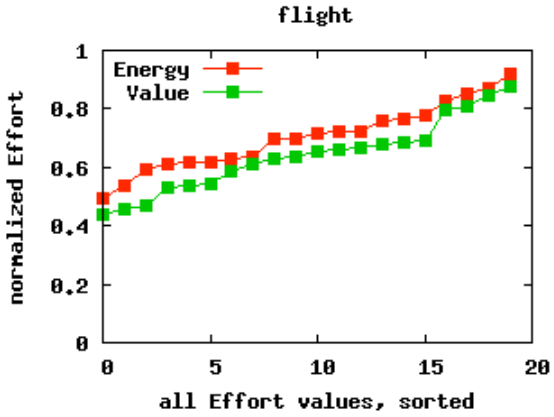


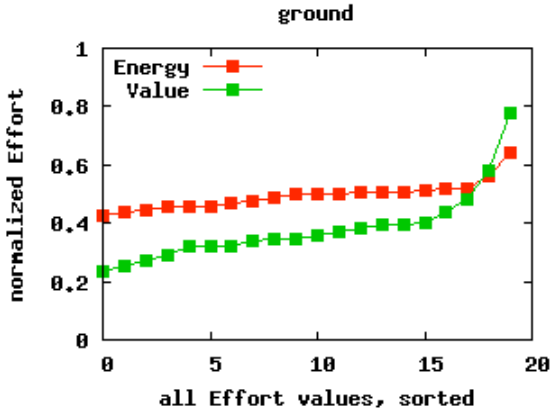


Value takes more effort

- Effort = staff months
- Results from 20 trials
  - Normalized min..max = 1..100
- Yawn!

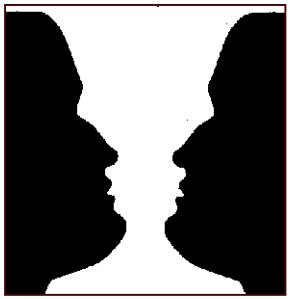
  - No surprises here
  - Better products take more time



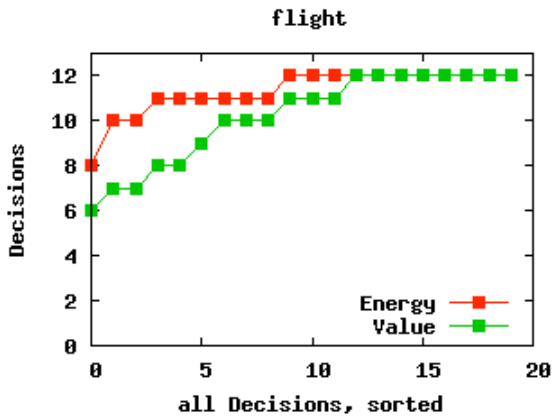


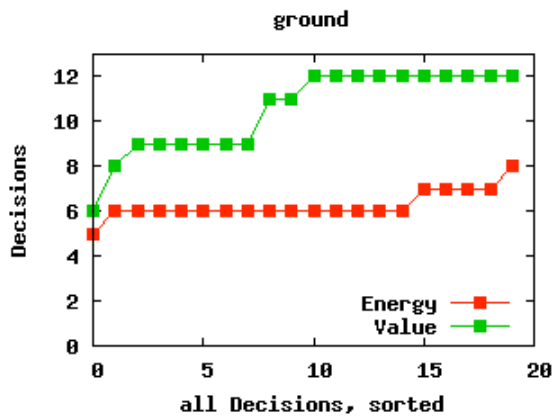
Value (is , isn't) harder to control

- Results from 20 runs
- Counts project variables that the AI search has decided to change
  - E.g. acap, pcap, pmat, etc
- Ambiguous results



- Flight systems
  - Same, or fewer decisions for value
- Ground systems
  - More decisions for value

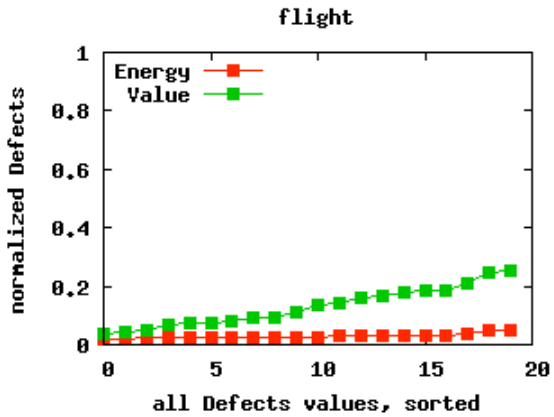


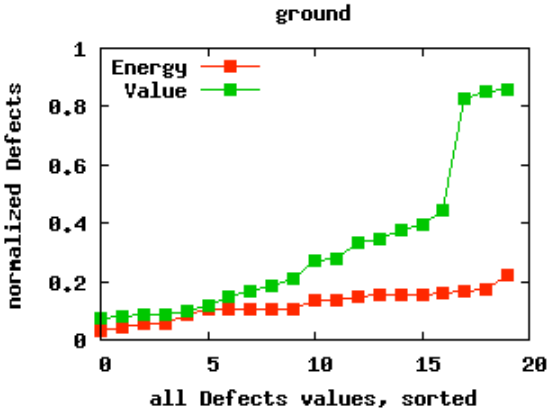


More value = more defects

- Defects per 100/KLOC
- Results from 20 trials
  - Normalized min..max 0..100
- More defects in value-based approach
- Whatever

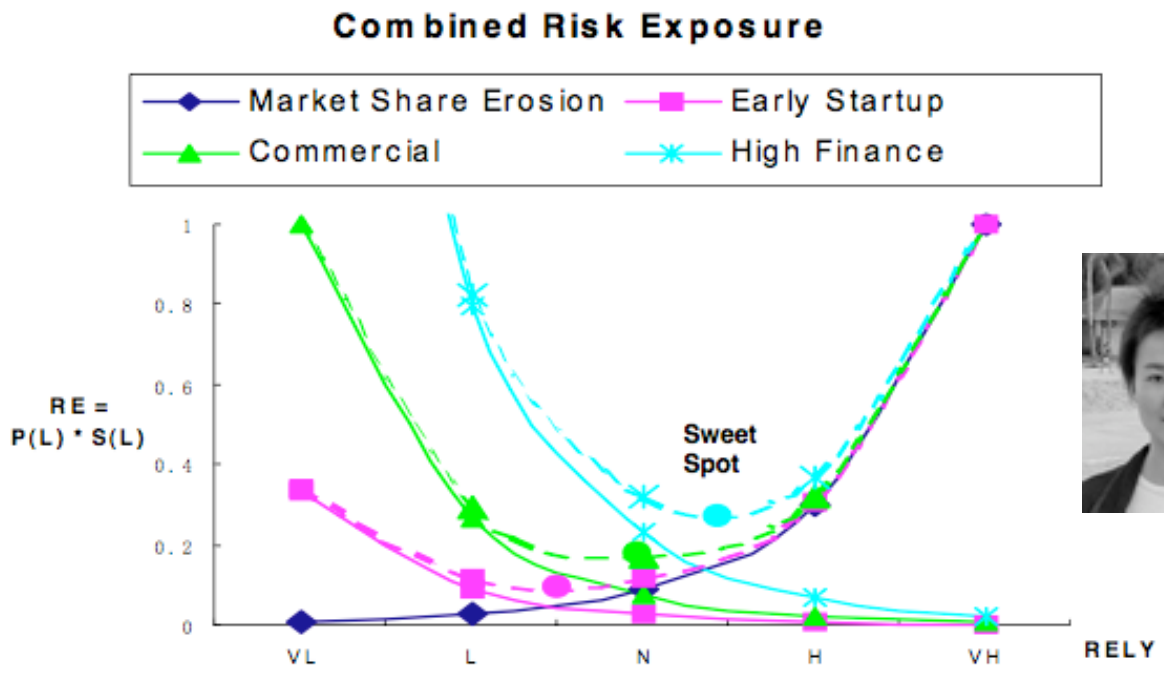
  - More to life than defect reduction
- Cautionary tale to our colleagues in automated software engineering
  - Where defect removal is king
  - And all else is secondary





Note: we are not the first to say value ≠ defects

- From [Huang06]
- Infinitely increasing software reliability is not necessarily the best plan



Conclusion

So what?

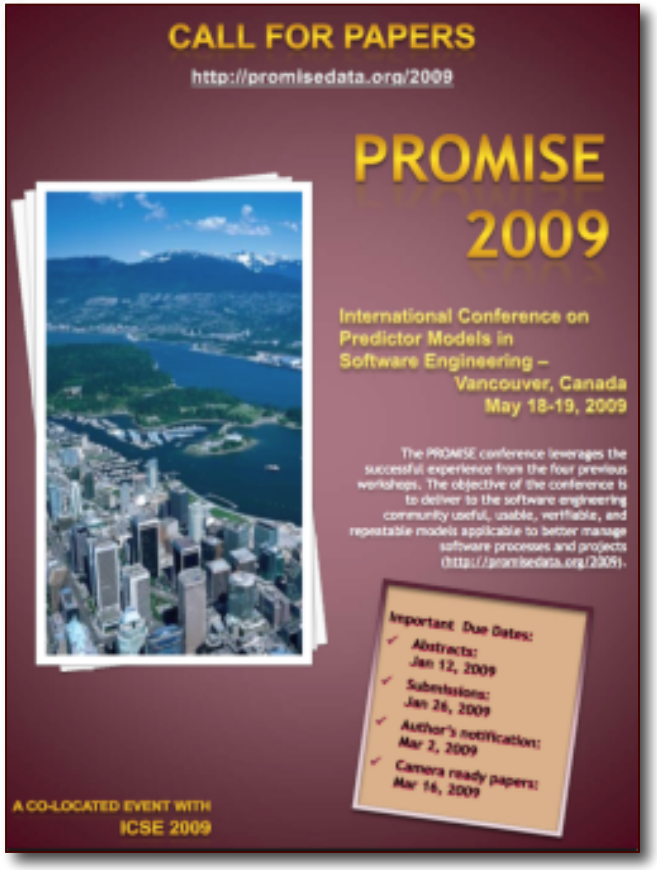
Conclusion

- Is value-based SE “ganz falsch”? (not even wrong)
  - Hard to tell, if we have a data drought
  - So seek stability in samples of the possibilities
- On sample, using 2 target functions and 2 systems:
  1. Value does not take more time (good news!)
  2. Value takes more effort (yawn)
  3. Value (<u>is</u> , <u>isn’t</u>) harder to control (huh?)
  4. More value = more defects (say what?)
- Clearly, not true for all value propositions
  - But are there classes of systems with repeated patterns of value propositions?
  - For those “value patterns”:
    - Under what conditions do 1,2,3,4 apply

Sound bites

- Come to PROMISE '09
- Value-based SE:
  - not even wrong?
- Data drought leading to conclusion uncertainty
  - Seek stability over samples
- On sampling some systems, we see
  1. Value does not take more time
  2. Value takes more effort
  3. Value (<u>is</u> , <u>isn't</u>) harder to control
  4. More value = more defects
- Community challenge:
  - when does 1,2,3,4 hold?

PROMISE '09



- www.promisedata.org/2009
- Reproducible SE results
- Papers:
  - and the data used to generate those papers
  - www.promisedata.org/data
- Keynote speaker:
  - Barry Boehm, USC
- Motto:
  - Repeatable, refutable, improvable
  - Put up or shut up