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

Managers have more options than they think
## Internal change
**(twiddle current project)**

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

<table>
<thead>
<tr>
<th>OSP: Orbital space plane</th>
<th>ranges:</th>
<th>min</th>
<th>max</th>
<th>fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec</td>
<td>1</td>
<td>2</td>
<td>data</td>
<td>3</td>
</tr>
<tr>
<td>flex</td>
<td>2</td>
<td>5</td>
<td>pvol</td>
<td>2</td>
</tr>
<tr>
<td>resl</td>
<td>1</td>
<td>3</td>
<td>rely</td>
<td>5</td>
</tr>
<tr>
<td>team</td>
<td>2</td>
<td>3</td>
<td>pcap</td>
<td>3</td>
</tr>
<tr>
<td>pmat</td>
<td>1</td>
<td>4</td>
<td>plex</td>
<td>3</td>
</tr>
<tr>
<td>stor</td>
<td>3</td>
<td>5</td>
<td>site</td>
<td>3</td>
</tr>
<tr>
<td>ruse</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>docu</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acap</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcon</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>apex</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l tex</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tool</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>seed</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cplex</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KSLOC</td>
<td>75</td>
<td>125</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Drastic change (massive project reorganization)

From Hoh Peter In And Barry Boehm, 1999

(not all drastic changes, just a sample)
### Drastic changes can be pretty drastic

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

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 * log_10 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

\[ \langle G, M, P, T, S \rangle \]

- \( G \) = A goal function to guide the search
- \( M \) = model
- \( P \) = Project options
- \( T \) = Tuning options
- \( S \) = A search engine to explore subsets \( p \) of \( P \)
\[ \langle G, M, P, T, S \rangle \]

\[ G = \text{goal} \]

- Goal = minimize score

\[ \text{score} = \frac{\sqrt{f \cdot M^2 + b \cdot D^2 + c \cdot E^2}}{\sqrt{f + b + c}} \]

- 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
$\langle G, M, P, T, S \rangle$

$P =$ project options

- AI searches (Min.. Max)
- As project grow, they grow less flexible
- OSP$_2$ < OSP < (flight, ground)
  - Flight: general description
  - OSP: orbital space plan
  - OSP$_2$: OSP v2.

<table>
<thead>
<tr>
<th>OSP: Orbital space plane</th>
<th>ranges:</th>
<th>min</th>
<th>max</th>
<th>fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>flex</td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>resl</td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>team</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>pmat</td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>stor</td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>ruse</td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>docu</td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>acap</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>pcon</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>apex</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>ltx</td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>tool</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>seed</td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>cplx</td>
<td></td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>KSLOC</td>
<td></td>
<td>75</td>
<td>125</td>
<td></td>
</tr>
</tbody>
</table>
\[
\langle G, M, P, T, S \rangle
\]
\(T=\text{tuning options}\)

- 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 \(\{3,1\}\);

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

\[(0.073 \leq m_a^+ \leq 0.21) \wedge (-0.178 \leq m_a^- \leq -0.078)\]

- Similarly for scale factors (and COQUALMO)

- \text{cplx, data, docu, pvol, rely, ruse, stor, time}
- \text{acap, apex, ltex, pcap, pcon, plex, sced, site, tool}
\[ \langle G, M, P, T, S \rangle \]
\[ S = \text{search} \]

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

Mann-Whitney results (95% confidence)
Defect predictions after SEESAW

- Always first, or ties with first
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)
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, OSP1, 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, e.g.
  - 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.