Can We Build Software Faster <u>and</u> Better <u>and</u> Cheaper?

Tim Menzies, Oussama El-Rawas
Lane Department of CS & EE
Morgantown, WV, USA
tim@menzies.us,
oelrawas@mix.wvu.edu

Jairus Hihn
Jet Propulsion Laboratory
California
USA
jairus.hihn@jpl.nasa.gov

ABSTRACT

Once a model is constructed (e.g. via data mining), we have to use it. Once a model is constructed (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it. Once a model is constructred (e.g. via data mining), we have to use it.

Categories and Subject Descriptors

B.4.8 [**Programming techniques**]: PerformanceModeling and prediction; I.6.4 [**Computing Methodologies**]: Model Validation and Analysis; G.3 [**General**]: Probability and Statistics—*statistical computing*

Keywords

COCOMO, Faster Better Cheaper, software process control

1. INTRODUCTION

Previously, PROMISE researchers have used *induction* (summarization of data into a model) to generate predictor models for software engineering. While useful, induction has several drawbacks:

- It only explores half the story. As Murray Cantor observed in his PROMISE 2008 keynote, once models are generated, they are used within some business context. While we need more papers on induction, it may be time to ask the PROMISE community to write more papers on model usage.
- Automatic induction assumes the existence of data. As discussed in Figure 1, many domains are *data starved* and it may be difficult to obtain that local data.
- More generally, we ask the question: why do we persist in building new models all the time? If there is any generality in software engineering it should be possible to reuse models. We should at least experiment with this approach rather than always assuming the best model is a new model.

For all the above reasons, we are motivated to explore model *use* and *reuse* rather than model *(re)generation.* Accordingly, in this paper, we see what conclusions we can draw from reusing models that predict for effort, defect, and software development time. Our focus will be an assessment of infamous "Faster, Better, Cheaper" development practices used at NASA in the 1990s. Within the NASA community, "Faster, Better, Cheaper" (FBC) is a strongly depre-

- David Raffo spent two years tuning and validating one process model to one site [25].
- Metrics-guru Norman Fenton spent years advocating careful data collection [9]. Recently, he has despaired of that approach. At a keynote address PROMISE 2007[1], Fenton shocked his audience by saying:

  > *"....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."*

- The COCOMO experience is similar to that of Fenton. After 26 years of trying, the COCOMO team has collected less than 200 sample projects for the COCOMO database. Also, even after two years of effort we were only able to add 7 records to a NASA-wide software cost metrics repository [16].

Figure 1: Evidence for data starvation in SE.

PROMISE '09 Vancouver, Canada USA

cated management practice (to say the least). FBC was advocated in the 1990s by the then-administrator of NASA, Daniel Goldin, as a method for reducing the expenditure of NASA. FBC was in-line with the direction that the Clinton administration's approach of doing more for less. FBC was initially successful: projects that usually cost over a billion were implemented at $\frac{1}{4}th$ that cost (e.g. Mars Pathfinder). However, subsequent failures (Mars Climate Orbiter and Polar Lander; the Columbia Shuttle disaster) caused a wealth of criticism of FBC.

When reusing models, one technical challenge is the *tuning problem in data starved domains*. When a model is reused from another site, it is often *tuned* to the local site. For example, Boehm et at. [4] advocate a certain functional form for generating software development effort estimates. In that form, the development effort is linear on a set of effort multipliers EM_i and exponential on a set of scale factors SF_j:

$$\begin{array}{rcl} effort & = & A \cdot KSLOC^E \cdot \prod_i \alpha_i EM_i \\ E & = & B + 0.01 \cdot \sum_j \beta_j SF_j \end{array} \quad (1)$$

The particular effort multipliers and scale factors recommended by Boehm et al. are shown in Figure 2. While Boehm et al offer default values for the Equation 1 variables, linear regression on local data can tune the α_i, β_j values to the particulars of a local site. Also, if there is insufficient data for a full tuning of α, β, then a coarse grain tuning can be achieved by just adjusting the A, B linear and exponential tuning parameters.

In data started domains, there is insufficient data to produce precise tunings. For example, At PROMISE 2005, we have reported very wide ranges in the post-tuning values of α and β [20]. Baker [2] offers a similar finding. After thirty 90% random samples of that data, the A, B ranges found during tuning were surprisingly wide:

$$(2.2 \leq A \leq 9.18) \wedge (0.88 \leq B \leq 1.09) \quad (2)$$

Elsewhere we have been partially successful in reducing these wide ranges with feature subset selection (FSS) [5,17]. However, despite years of work, we now report that FSS reduces but does not sufficiently narrow the ranges of A, B, α, β[2]

Having failed to generate narrow tunings, we have been exploring a new approach The STAR tool [8, 19, 21]. tool checks for stable conclusions within a wide range of possible tunings. As shown below, we can find stable conclusions using a combination of simulated annealing and Bayesian sensitivity analysis.

STAR is an excellent candidate for exploring issues related to "Faster, Better, Cheaper". STAR combines estimates for effort E, development time in months M and number of delivered defects D in an equation that weights each estimates utilities $\{f, c, b\}$ (short for $\underline{f}$aster, $\underline{c}$heaper, $\underline{b}$etter[3].

$$score = \frac{\sqrt{f.M^2 + b.D^2 + c.E^2}}{\sqrt{f + b + c}} \quad (3)$$

This *score* value models the Euclidean distance to minimum values for all the $\{M, D, E\}$ predictions. If we normalize the predictions min..max to 0..1 then Equation 3 has the range one to zero and *lower* scores are *better*.

In this study, we run STAR in three modes:

- BF, or {*better, faster*}, where $c = 0$ and $b = f = 1$ (so we are ignoring development cost);
- BC, or {*better, cheaper*}, where $f = 0$ and $b = c = 1$ (so we are ignoring delivery time);
- CF, or {*faster, cheaper*}, where $b = 0$ and $f = c = 1$ (so we are ignoring delivered defects).
- FBC, or {*faster, better, cheaper*}, where $b = f = c = 1$ (so we are trying to achieve all goals).

The surprise result of this paper is that, contrary to the prevailing wisdom at NASA, FBC is not necessarily a disastrous development methodology. In fact, in $\frac{7}{9}$ of our studies, it produces results within 1% of the minimum estimates generated by any of BF,BC,FBC. We speculate that what really went wrong with FBC at NASA is that it was used as a front for FC; i.e. built it faster and cheaper, but forget any considerations of quality.

The rest of this paper is structures as follows. XXXX

Before beginning, we digress to make one point. This paper is only commenting on FBC as a *software development* practice. STAR only models software development. so we cannot comment on the more general field of systems engineering. In future work, we plan to (a) implement hardware development models inside STAR; then (b) make an informed comment on NASA's use of FBC in the 1990s for building complex combinations of software *and* hardware devices.

2. HISTORY

The main approach to implementing FBC within NASA was to down size projects and reduce their cost and complexity, concentrating on producing missions in volume. Reducing funding naturally meant that less verification and testing was possible within budget and schedule constraints. The reasoning behind this however was to be able to produce a larger volume of unmanned missions, which would counteract the expected higher rate of mission failure. This would, optimally, yield more successful missions as well as more scientific data produced by these projects. Another focus in this policy was allowing teams to take acceptable risks in projects to allow for cost reduction, and possibly using new technology that could reduce cost while possibly providing more capabilities. This was accompanied by the the new view that was being pushed at NASA by Goldin that "it's ok to fail" [26], which was rather misunderstood. This new policy was meant to eliminate huge budget missions of the past, that upon possible failure would yield large losses. Project cost used to routinely exceed the $1 billion mark, while the first FBC project, the Mars Pathfinder, was completed for a fraction of the cost, netting at about $270 million [15].

Some within NASA, such as 30 year veteran Frank Hoban, supported these policies [15]. Some viewed these new policies as a necessary break from traditional policies that were very risk averse. The additional cost reduction, accompanied by the additional risk, was to allow for a path to cheap and commercial space flight. Even given the reduced funding, the Mars Pathfinder mission, along with other first gen-

[2] We are aware of only one other report, by Korte & Port at PROMISE 2007 [14] of these large post-tuning ranges. It is an open question why other researchers have not reported these large ranges. Perhaps these large post-tuning ranges have been missed since researchers rarely check for this effect.

[3] We will use uppercase B to denote the COCOMO linear tuning variable of Equation 1 and lower b to denote the business utility associated with defect predictions of Equation 3

	Definition	Low-end = {1,2}	Medium ={3,4}	High-end= {5,6}
Defect removal features				
execution-based testing and tools (etat)	all procedures and tools used for testing	none	basic testing at unit/ integration/ systems level; basic test data management	advanced test oracles, assertion checking, model-based testing
automated analysis (aa)	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 (pr)	all peer group review activities	none	well-defined sequence of preparation, informal assignment of reviewer roles, minimal follow-up	formal roles plus 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 risks 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 35%	35% - 90%	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 $(\frac{frequency\ of\ major\ changes}{frequency\ of\ minor\ changes})$	$\frac{12\ months}{1\ month}$	$\frac{6\ months}{2\ weeks}$	$\frac{2\ weeks}{2\ days}$
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

Figure 2: Features of the COCOMO and COQUALMO models used in this study.

eration FBC missions, were successes. This fueled enthusiasm to apply FBC across all of NASA and further reduce spending per mission, as well as reduce NASA expenditure by reducing the work force by a third. FBC was extended to be applied on manned space missions as well, where funding was also reduced. Coming into a space shuttle program that was starting to age and in need of updates, the new policies imposed cuts in funding from 48% of the NASA budget to 38% [11], further straining that program. Further more, a single prime contractor (Lockheed Martin) was used for missions in another bid to reduce cost and managerial complexity [23, 31].

This produced opposition within NASA, where traditionally issues pertaining to the shuttle were designated LOVC (Loss of Vehicle and Crew) and given priority over all other issues, including cost. However the cost cuts and layoffs that ensued were too much for teams, and caused a blow to morale. In addition there was a progressive loss of veteran scientists, engineers and managers who had accepted offers for early retirement that were extended to them [11].

Despite this, additional projects were on the way in the form of the Mars Climate Orbiter and the Mars Polar Lander. These two projects were more aggressive implementations of FBC, especially when it came to the Faster-Cheaper part of those policies. Costs of the Orbiter and the Lander were brought down to $125 million and $165 million respectively [28]. This was much lower that the previous Pathfinder mission, which itself cost slightly less than $300 million. The success of these missions would've furthered the FBC mantra within NASA and JPL, and would've been seen as breaking new ground in terms of mission completion with the kind of staff and cost reductions they had compared to previous missions, even the Pathfinder [10].

Given its early success in terms of mission delivery, FBC started being more aggressively applied to missions in NASA. One product of this were the above mentioned Mars Climate Orbiter and Polar Lander. Each cost about 40% less than the previous Pathfinder mission, which is extraordinary given that Pathfinder had been touted as a money saver by NASA when compared to previous missions like Viking, which cost about $935 million in 1974 Dollars (equivalent to $3.5 billion in 1997 dollars). Both of these missions however failed. Using a single contractor had weakened quality assurance and resulted in flaws that caused the loss of these two Mars missions. These flaws had been software flaws that could have easily been rectified if they had been discovered on the ground. One of these flaws was a failure to convert from imperial to metric units, causing the loss of the Climate Orbiter [22]. The Mars Program Independent Assessment Team Report [31] found that these missions were under-staffed, under-funded by at least 30%, and too tightly scheduled.

Elsewhere, across the Atlantic in the UK, another Mars mission to deliver a lander, designated the Beagle 2, was under way. This mission way also developed cheaply, applying the same concepts in design and implementation that NASA was at the time using. The lander however was declared lost after not establishing contact after separation from the mars express vehicle [1].

One other failure that FBC was blamed for was the Columbia Shuttle disaster in 2003. This was post-Goldin, at a point where NASA had realized the excessive cost cutting and staff reducing policies needed to be changed. After that disaster, critics quickly pointed the finger to these missions being under funded due to FBC. There were many calls, especially politically, for throwing FBC "in the waste basket" [7, 24].

3. CASE STUDIES

This paper assess the value of "Faster, Better, Cheaper" using the three cases studies of Figure 3. These cases studies represent the same class of NASA flight software, at increasing levels of specificity:

- *Flight* is a general description of flight software at NASA's Jet Propulsion Laboratory.
- *OSP* is a specific flight system: the GNC (guidance, navigation, and control) component of NASA's 1990s *Orbital Space Plane*;
- *OSP2* is a later version of OSP;

In the sequel, the following will be important. Our case studies can be ranked

$$flight > OSP > OSP2$$

according to how many open options they offer a project manager:

- In the case of flight systems, the description is very general and managers have many options.

	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
				site	6
JPL flight software	rely	3	5	tool	2
	data	2	3	sced	3
	cplx	3	6		
	time	3	4		
	stor	3	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	KSLOC	7	418		

Figure 3: Three case studies. Numeric values $\{1, 2, 3, 4, 5, 6\}$ map to $\{verylow, low, nominal, high, veryhigh, extrahigh\}$. The terms in column 2 come from Figure 2.

- In the case of OSP2, most of the project options are pre-determined and project managers have very little opportunity to effect the course of a project.
- OSP is an early version of OSP2 and, measured in terms of the number of open options, falls in between flight and OSP2.

Figure 3 describes the details of flight, OSP, and OSP2. Note that Figure 3 does not mention all the features listed in Figure 2 inputs. For example, our defect predictor has inputs for use of *automated analysis*, *peer reviews*, and *execution-based testing tools.* For all inputs not mentioned in Figure 3, values are picked at random from the full range of Figure 2.

The important thing to note from Figure 3 is the number of open options *not* specified in the description of the projects. Some of the features in Figure 3 are known precisely (see all the features with single *values*). But many of the features in Figure 3 do not have precise values (see all the features that *range* from some *low* to *high* value). Sometimes the ranges are very narrow (e.g., the process maturity of JPL ground software is between 2 and 3), and sometimes the ranges are very broad. The broader the range of options, the more freedom a manager has to adjust with the internals of their project. As we shall see below, this ability to make project adjustments will critically effect our results.

4. STAR

STAR uses the cases studies of Figure 3 as inputs to a Monte Carlo simulation. STAR contains the COCOMO effort E estimator [4] but also the COCOMO development months M estimator [4, p29-57], and COQUALMO D defects estimator [4, p254-268], These estimator generate the $\{E, M, D\}$ variables of Equation 3.

While STAR's effort and months models share the same $\{A, B, \alpha, \beta\}$ values, the defect model has a separate set of tuning variables, which we will call γ. Using 26 years of publications about COCOMO-related models, we inferred the minimum and maximum values yet seen for $\{A, B, \alpha, \beta, \gamma\}$. For example, the A, B min/max values come from Equation 2. We use the variable T to store the range of possible values for these tuning variables.

STAR runs as follows. First, a project P is specified as a set of min/max ranges to the input variables of STAR's models:

- If a variable is known to be exactly x, then then $min = max = x$.
- Else, if a variable's exact value is not known but the range of possible values is known, then min/max is set to the smallest and largest value in that range of possibilities.
- Else, if a variable's value is completely unknown then min/min is set to the full range of that variable in Figure 2.

Second, STAR's simulated annealer[4] seeks constraints on P that most reduce the score of Equation 3. A particular subset of $P' \subseteq P$ is scored by using P' as inputs to the COCOMO and COQUALMO. When those models run, variables are selected at random from the min/max range of possible tunings T and project options P. In practice, the majority of the variables in P' can be removed without effecting the score; i.e. our models exhibit a *keys effect* where a small number of variables control the rest [18]. Finding that minimal set of variables is very useful for management since it reveals the *least* they need to change in order to *most* improve the outcome. Hence, simulated annealing, STAR takes a third step.

In STAR's third step, a Bayesian sensitivity analysis finds the the smallest subset of P' that most effects the output. The scores seen during simulated annealing are sorted into the 10% best and the 90% rest. Members of P' are then ranked by their Bayesian probability of appearing in *best*. For example, after $K = 10,000$ runs of the simulated annealing, the output scores are divided into 1,000 lowest 10% *best* solutions and 9,000 *rest*. The range $rely = vh$ might appears 10 times in the *best* solutions, but only 5 times in the *rest*. Hence:

$$\begin{aligned}
E &= (reply = vh) \\
P(best) &= 1000/10000 = 0.1 \\
P(rest) &= 9000/10000 = 0.9 \\
freq(E|best) &= 10/1000 = 0.01 \\
freq(E|rest) &= 5/9000 = 0.00056 \\
like(best|E) &= freq(E|best) \cdot P(best) = 0.001 \\
like(rest|E) &= freq(E|rest) \cdot P(rest) = 0.000504 \\
P(best|E) &= \frac{like(best|E)}{like(best|E) + like(rest|E)} = 0.66
\end{aligned} \quad (4)$$

Previously [6] we have found that Equation 4 is a poor ranking heuristic since it is distracted by low frequency evidence. For example, note how the probability of E belonging to the best class is moderately high even though its support is very low; i.e. $P(best|E) = 0.66$ but $freq(E|best) = 0.01$. To avoid such unreliable low frequency evidence, we augment Equation 4 with a support term. Support should *increase* as the frequency of a range *increases*, i.e. $like(x|best)$ is a valid support measure. STAR1 hence ranks ranges via

$$P(best|E) * support(best|E) = \frac{like(x|best)^2}{like(x|best) + like(x|rest)} \quad (5)$$

After ranking members of P', STAR then imposes the top i-th ranked items of P' on the model inputs, then running the models 100 times. This continues until the scores seen using $i + 1$ items is not statistically different to those seen using i (t-tests, 95% confidence). STAR returns items $1..i$ of P' as the *least* set of project decisions that *most* reduce effort, defects, and development time. We call these returned items the *policy.*

Note that STAR constraints the project options P but never the tuning options T. That is, the *policy* generated by STAR contains parts of the project options P that most improve the score, despite variations in the tunings T. This approach has the advantage that it can reuse COCOMO models without requiring data for local tuning.

Previously [19] we have shown that this approach, that does not use local tuning, generates estimates very similar to those generated after using local tuning via the "LC" method proposed by Boehm and in widespread use in the COCOMO community [3]. We have explained this effect as follows. Uncertainty in the project options P and the tuning options T contribute to uncertainty in the estimates generated by STAR's models. However, at least for the COCOMO and COQUALMO models used by STAR, the uncertainty created by P dominates that of T. Hence, any uncertainty in the output can be tamed by constrain P and not T.

5. RESULTS

5.1 Format of Results

Figure 4 shows the defects, months, and effort estimates seen imposing the *policy* learned by simulated annealing and Bayesian sensitivity analysis.

- The results have separate divisions for defects, months, and effort.
- Each division is sub-divided into the results for flight, OSP, and OSP2 results.

[4] Simulated annealers randomly alter part of the some *current* solution. If this *new* solution scores better than the current solution, then *current* = *new*. Else, at some probability determined by a temperature variable, the simulated annealer may jump to a sub-optimal *new* solution. Initially the temperature is "hot" so the annealer jumps all over the solution space. Later, the temperature "cools" and the annealer reverts to a simple hill climbing search that only jumps to new better solutions. For more details, see [12].

estimated	case study	rank (Mann Whitney 95%)	what	median estimate	2nd quartile, median, 3rd quartile
defects	flight	1	BC	1	
		1	FBC	2	
		2	BF	4	
		2	CF	7	
		3	do nothing	98	
	OSP	1	BC	6	
		2	FBC	7	
		3	BF	9	
		4	CF	62	
		5	do nothing	96	
	OSP2	1	FBC	10	
		1	BF	11	
		1	BC	11	
		2	do nothing	54	
		3	CF	97	
months	flight	1	FBC	39	
		2	BF	42	
		2	CF	43	
		2	BC	44	
		3	do nothing	100	
	OSP	1	CF	61	
		2	BF	64	
		3	FBC	67	
		4	BC	85	
		5	do nothing	99	
	OSP2	1	CF	74	
		2	BF	83	
		3	FBC	89	
		4	do nothing	94	
		5	BC	100	
effort	flight	1	FBC	9	
		2	BF	11	
		2	CF	12	
		3	BC	14	
		4	do nothing	98	
	OSP	1	CF	45	
		1	FBC	47	
		2	BC	49	
		2	BF	53	
		3	do nothing	97	
	OSP2	1	BC	65	
		2	FBC	77	
		3	BF	81	
		4	CF	91	
		4	do nothing	95	

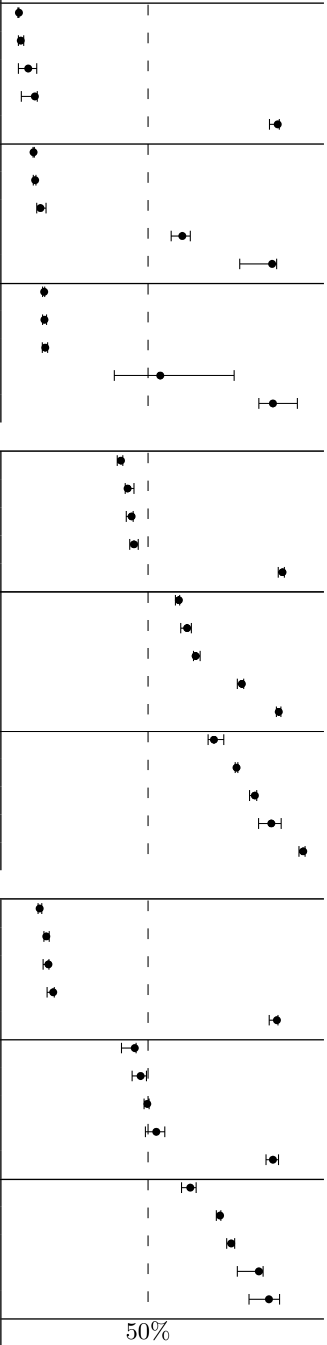

Figure 4: Results

Within each sub-division, the rows are sorted by median scores. The "Do nothing" row comes from Monte Carlo simulations over the project range P, without any restrictions.

The *rank* results shown in column three show the results of a statistical comparison of each sub-division. Two rows have the same rank if there is no statistical difference in their distributions. We use Mann-Whitney for this comparison for the following reasons:

- The random nature of Monte Carlo simulations, the inputs to each run are not paired;
- Ranked tests make no, possibly inappropriate, assumption about normality of the results.

Each row shows results from 100 calls to Equation 3:

- Within each division, the results are normalized to run 0..100, min..max.
- Each row shows the 25% to 75% quartile range of the normalized scores collected during the simulation.
- The median result is shown as a black dot.

All the performance scores (effort, months, defects) get *better* when the observed scores get *smaller*; i.e. move over the left.

5.2 Observations and Recommendations

Three aspects of Figure 4 deserve out attention. Firstly, it is almost always true that some optimizations on any pair or triple from "Faster, Better, Cheaper" can reduce defects *and* months *and* effort from the levels seen in the baseline "do nothing" scenario. Hence, we recommend the widespread use of tools like STAR.

Secondly, we also recommend we advise applying tools like STAR as early as possible in the life cycle of a project while there still exists a wide range of process options. Note in Figure 4 that as we move from flight to OPS to OSP2, the median performance scores get worse. In order to explain this effect, we repeat remarks made above: our case studies are sorted in decreasing order of "number of open options" (there is much that can be adjusted within the general description of flight systems; fewer adjustments options are possible in OSP; and even fewer adjustments are possible in OSP2). As we *decrease* the number of open options, our ability to find "fixes" to the current project also decrease.

Thirdly, the goal of faster *and* better *and* cheaper is not overly ambitions. Pre-experimentally, we agreed with the standard view of "Faster, Better, Cheaper? Pick any two". We believed that when optimizing for three criteria, it may be sometimes necessary to accept non-minimal results for one of the criteria. However, contrary to our expectations, in the case of:

- defects for OSP2,
- months for flight systems, and
- effort for flight systems

FBC achieves the best (lowest) median results. Also, for:

- defects for flight systems and
- defects for OSP2 and
- effort for OSP

FBC is statistically indistinguishable from the best (lowest) median result. Lastly, in the case of

- defects for OSP

FBC's median is within 1% of the best (lowest) median result. That is, in $\frac{7}{9}$ of the divisions of Figure 4, FBC achieves either minimum or very close to minimum values. Hence, we still endorse STAR's default setting of $b = f = c = 1$; i.e. try to optimize on all three criteria.

5.3 Discussion

These results appear to contradict the historical record. Our last paragraph concluded that "Faster, Better, Cheaper" can usually be achieved with minimal compromises on individual criteria. How are we to reconcile this result with the NASA experience, described above?

One way to explain the very large failures within the FBC program is to speculate that, sometimes, FBC was a front for CF (i.e.. cheaper and faster, as the expense of quality). Figure 4 shows the disastrous effects of CF. CF leads to poor defect results in the case of OSP and the *worst* defect results in the case of OSP2.

Also, one thing forgotten about FBC is that, usually, it worked. Despite all the criticism against it, FBC successful/partially successful in 136 of the total of 146 missions launched during the period that Goldin was administrator. This would be called an overall success if it hadn't been for the largely publicized failure. That is, FBC was mostly a technical success, but a PR failure [30].

Some the decisions made under the banner of FBC are questionable; i.e. staff reductions leading to loss in veteran engineers and managers to retirement and causing experienced managerial staff to be stretched too thin given tight scheduling [31]. This forced projects to use inexperienced managers which caused management mix ups and human error.

Like Spear [27] , we would endorse FBC, but under the condition that it is better managed. Tony Spear, a JPL veteran engineer from 1962 to 1998, testified to the possible effectiveness of FBC. Despite mentioning problems with FBC (a fixation on cost, causing cost cuts that were too much for 2^{nd} generation FBC projects), he recommended *not* to discard it. Rather, he argued for a more focused way of implementing it, concentrating on aspects such as building and retaining talent, taking advantage of advancements in technology such as the Internet, and advancing methods used in project development and verification [26, 27].

To Spear's recommendation we would add that when applying FBC, never surrender the quest of "better". Observe how, in Figure 4, whenever we optimize for "Better" using $b = 1$, we always reduce defects by about an order of magnitude over the baseline "do nothing" result. The only time that our optimizer does not reduce defects is when the "better" utility is set to zero (see the CF detect results). Hence, we recommend always setting $b = 1$.

6. RELATED WORK

This paper explores a unique solution to data starvation: sample across the space of possible model tunings. Elsewhere, we have explored solving local data starvation using imported data. With Turhan et al. [29], we have compared the performance of *defect prediction models* built from local data or imported data. Imported data, we found, can have impractically high false alarm rates when applied to local data. However, we also found that, with the right *relevancy filtering*, we can sometimes solve the data drought problem using imported data. For example, building a Bayes classifier from the 10 training instances nearest each test instance can reduce that false alarm rate by 300%.

In other work, Kitchenham et.al. [13] study *effort estimation models* built from local or imported data. They conduct a systematic review of ten projects, comparing estimates using historical data within the same company or imported from another. In no case was it better to use data from other sites, and sometimes importing such data yielded significantly worse estimates.

Note that the Kitchenham et al. result is the complete reverse of the Turhan et al. That is, the results of [29] may not always apply and local data starvation can not always we solved with data from another site. Hence, this work.

7. CONCLUSION

BFC not a bad thing

STAR rules

more process options than you know

Time to turn PROMISE from model generation to model conclusion. One thing we are very interested in is the kind of discussion this paper inspires. Will our conclusions be rejected because they are “just” based on COCOMO? If our models are bad, where are ones that are better? Is there some better model that a large community endorses as a valid source of insight into software engineering? Is there no generality in software engineering? We look forward to a lively discussion on these issues.

8. REFERENCES

[1] Beagle 2 mission profile. http://solarsystem.nasa.gov/missions/profile.cfm?MCode=Beagle_02.

[2] D. Baker. A hybrid approach to expert and model-based effort estimation. Master’s thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007. Available from https://eidr.wvu.edu/etd/documentdata.eTD?documentid=5443.

[3] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[4] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

[5] Z. Chen, T. Menzies, and D. Port. Feature subset selection can improve software cost estimation. In *PROMISE’05*, 2005. Available from http://menzies.us/pdf/05/fsscocomo.pdf.

[6] R. Clark. Faster treatment learning, Computer Science, Portland State University. Master’s thesis, 2005.

[7] K. Cowig. Nasa responds to the columbia accident report: Farewell to faster - better - cheaper, September 2003. http://www.spaceref.com/news/viewnews.html?id=864.

[8] O. El-Rawas. Software process control without calibration. Master’s thesis, 2008. Available from http://unbox.org/wisp/var/ous/thesis/thesis.pdf.

[9] N. E. Fenton and S. Pfleeger. *Software Metrics: A Rigorous & Practical Approach (second edition)*. International Thompson Press, 1995.

[10] M. hardin. Mars climate orbiter nearing sept. 23 arrival, September 1999. JPL Universe, Vol. 29, No. 19.

[11] S. Key. Columbia, the legacy of ”better, faster, cheaper”?, July 2003. http://www.space-travel.com/reports/Columbia__The_Legacy_Of_Better__Fas%ter__Cheaper.html.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[13] B. A. Kitchenham, E. Mendes, and G. H. Travassos. Cross- vs. within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, pages 316–329, May 2007.

[14] M. Korte and D. Port. Confidence in software cost estimation results based on mmre and pred. In *PROMISE ’08: Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 63–70, 2008.

[15] leonard david. nasa report: too many failures with faster, better, cheaper, march 2000. http://www.space.com/businesstechnology/business/spear_report_000313.ht%ml.

[16] T. Menies, K. Lum, and J. Hihn. The deviance problem in effort estimation. In *PROMISE, 2006*, 2006. Available from http://menzies.us/06deviations.pdf.

[17] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, November 2006. Available from http://menzies.us/pdf/06coseekmo.pdf.

[18] T. Menzies, D.Owen, and J. Richardson. The strangest thing about software. *IEEE Computer*, 2007. http://menzies.us/pdf/07strange.pdf.

[19] T. Menzies, O. Elrawas, B. Barry, R. Madachy, J. Hihn, D. Baker, and K. Lum. Accurate estimates without calibration. In *International Conference on Software Process*, 2008. Available from http://menzies.us/pdf/08icsp.pdf.

[20] T. Menzies and A. Orrego. Incremental discreatization and bayes classifiers handles concept drift and scaled very well. 2005. Available from http://menzies.us/pdf/05sawtooth.pdf.

[21] T. Menzies, S. Williams, O. El-waras, B. Boehm, and J. Hihn. How to avoid drastic software process change (using stochastic statbility). In *ICSE’09*, 2009. Available from http://menzies.us/pdf/08drastic.pdf.

[22] NASA. Mars climate orbiter mishap investigation board phase i report. November 1999.

[23] T. o. F. B. C. NASA watch. Faster - better - cheaper under fire. http://www.nasawatch.com/fbc.html.

[24] I. F. O. PROFESSIONAL and A.-C. TECHNICAL ENGINEERS. Ifpte report on the effectiveness of nasa’s workforce & contractor policies, March 2003. http://www.spaceref.com/news/viewsr.html?pid=10275.

[25] D. Raffo. Modeling software processes quantitatively and assessing the impact of potential process changes of process performance, May 1996. Ph.D. thesis, Manufacturing and Operations Systems.

[26] T. Spear. Nasa fbc task final report, March 2000. mars.jpl.nasa.gov/msp98/misc/fbctask.pdf.

[27] T. Spear. Testimony on nasa fbc task before the subcommittee on science, technology, and space, March 2000. www.nasawatch.com/congress/2000/03.22.00.spear.pdf.

[28] D. Tuite. Better, faster, cheaperâĂŤpick any two: That old mantra used to be a touchstone for development. but does it still ring true?, March 2007. `http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=14997`.

[29] B. Turhan, T. Menzies, A. Bener, and J. Distefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 2009. Available from `http://menzies.us/pdf/08ccwc.pdf`.

[30] M. Turner. Faster, cheaper, and more ... metric?, August 2003. http://www.spacedaily.com/news/oped-03zz.html.

[31] T. Young, J. Arnold, T. Brackey, M. Carr, D. Dwoyer, R. Fogleman, R. Jacobson, H. Kottler, P. Lyman, and J. Maguire. Mars program independent assessment team report. *NASA STI/Recon Technical Report N*, pages 32462–+, Mar. 2000.