Can We Build Software Faster and Better and Cheaper?

Tim Menzies, Oussama El-Rawas
CS&EE, West Virginia University, USA
tim@menzies.us, oelrawas@mix.wvu.edu

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

Barry Boehm
Computer Science Dept.
Uni. S. California, USA
boehm@sunset.usc.edu

ABSTRACT

"Faster, Better, Cheaper" (FBC) was a development philosophy adopted by the NASA administration in the mid to late 1990s. that lead to some some dramatic successes such as Mars Pathfinder as well as a number highly publicized mission failures, such as the Mars Climate Orbiter & Polar Lander.

The general consensus on FBC was "Faster, Better, Cheaper? Pick any two". According to that view, is impossibly to optimize on all three criteria without compromising the third. This paper checks that view using an AI search tool called STAR. We show that FBC is indeed feasible and produces similar or better results when compared to other methods However, for FBC to work, there must be a balanced concern and concentration on the quality aspects of a project. If not, "FBC" becomes "CF" (cheaper and faster) with the inevitable lose in project quality.

Categories and Subject Descriptors

B.4.8 [**Programming techniques**]: PerformanceModeling and prediction; I.6.4 [**Computing Methodologies**]: Model Validation and Analysis

Keywords

software engineering, predictor models, COCOMO, Faster Better Cheaper, simulated annealing, software processes

1. INTRODUCTION

Previously, PROMISE researchers have used *induction* (summary 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 predictive models are generated, they are used within some business context. While we need more papers on predictive model *generation*, it may be time to ask the PROMISE community to write more papers on predictive model *usage*.
- Automatic induction assumes the existence of data. Many domains are *data starved* where it is hard to obtain local data.
- More generally, we ask the question: why do we persist on generating new predictive models all the time? If there is any generality in software engineering it should be possible to reuse predictive models. We should at least experiment with this approach rather than always assuming the best predictive model is a new model.

Hence, we explore predictive model *use* and *reuse* rather than predictive model *generation*. Model reuse is complicated by extreme data starvation: the *tuning variance* problem arises if there is insufficient data to generate precise tunings. Our STAR tool [9, 23, 27] uses AI methods to conduct large scale "what-if" queries over software predictive models. If parts of a predictive models are not known with certainty, STAR seeks stable conclusions within the space of possible values.

In previous publications [24], we had explored the benefits to using STAR for the purpose of presenting a concise strategy to software project managers. This paper applies STAR to assess the infamous "Faster, Better, Cheaper" (FBC) development practices used at NASA in the 1990s. 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 $\frac{1}{4}th$ of that cost (e.g. Mars Pathfinder). However, subsequent failures (Mars Climate Orbiter and Polar Lander; the Columbia Shuttle disaster) lead to much criticism of FBC.

STAR is a natural tool for assessing FBC, since it uses COCOMO based software engineering models that have already been usesd within NASA. These COCOMO models represent FBC in the following manner:

- Faster is represented by the Months model, which estimates the total development months needed for a software project.
- Better is represented by the Defects model, which estimates the number of delivered defects per KLOC (thousand lines of code).

PROMISE '09 Vancouver, Canada USA

- Cheaper is represented by the Effort model, which estimates the effort that is needed for a software project in person-months, and hence can be used to estimate the cost of the development of the project.

Note that for all the models, lower is better. The tool combines estimates with utility weights $\{f, b, c\}$ (short for Faster, Better, Cheaper):

$$score = \frac{\sqrt{f.M^2 + b.D^2 + c.E^2}}{\sqrt{f + b + c}} \quad (1)$$

This *score* value represents the Euclidean distance to the normalized values of the predictions of development effort "E"; total development months "M"; and "D", the number of delivered defects per KLOC. This is the utility function that is used in order to assess any given set of "policies" that might be presented to be implemented in a given software project. Given that we normalize the predictions min..max to 0..1 then Equation 1 has the range one to zero and *lower* scores are *better*. STAR searches for the minimal set of project changes that most reduces this *score*.

By adjusting the various values of (f, b, c), we can compare the effects of methodologies that emphasize different project goals:

- BF = "better, faster" i.e. $c = 0$ and $b = f = 1$;
- BC = "better, cheaper" i.e. $f = 0$ and $b = c = 1$;
- CF = "cheaper, faster" i.e. $b = 0$ and $f = c = 1$;
- FBC = "faster, better, cheaper" i.e. $b = f = c = 1$.

This paper reports the results of such a comparison. After a discussion of related work, we will review the sad history of "Better, Faster, Cheaper" at NASA. We will then describe the case studies will be explore, as well as the STAR systems. In the subsequent results section we will offer the surprising observation 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,

> *FBC produces results that are within 5% of the minimum estimates generated by any of BF,BC,CF.*

Our results suggest that (a) FBC is in fact a viable development methodology but (b) what really went wrong with FBC at NASA was that it was used as a front for CF; i.e. built it faster and cheaper, while almost disregarding quality considerations.

Our more general conclusion is that we can make more *use* and *reuse* of predictive models. If stable conclusions exist across the tuning space, then AI tools like STAR can offer conclusions from predictive models, even when their is insufficient data for tuning. This is an powerful aspect of AI tools like STAR since we often lack the data required to precisely tune an imported model to a local domains. It is hoped that that this paper will inspire more papers at future PROMISE conferences that explore not just model *generation*, but also the *reuse* of those generated model in new domains. To facilitate that discussion, we have made all our code openly available[1] and we are building a web interface to an executable version of STAR[2]. That interface will be on-line by the time of the PROMISE'09 conference.

[1] `http://unbox.org/wisp/tags/STAR/2.1/`
[2] `http://nova.unbox.org`

	domain experts	no domain experts
data starved (little or no data)	build predictive models manually	reuse predictive models from other sites, without tuning; sample across range of possible options. E.g. STAR.
data poor (small amounts of data)	build predictive models manually; tune with local data; e.g. [10]	reuse predictive models from other sites; tune with local data; E.g. Boehm's local calibration of COCOMO models [3].
data rich (much data)	build predictive models automatically e.g. [6, 14, 19, 21, 26].	

Figure 1: Predictive model construction options.

- Metrics-guru Norman Fenton study data collection [11] for many years. Recently, he has despaired of that approach. At a keynote address PROMISE 2007[3], 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 [19].

Figure 2: Evidence for data starvation in SE.

2. RELATED WORK

Figure 1 places STAR in the context of related work. Predictive models can be build automatically or manually. Automatically generated predictive models can be automatically validated using many methods including (e.g.) the cross-validation procedures used in much data mining research [37]. Manually generated predictive models, built using human expertise, can be remarkably effective: see Fenton's PROMISE'07 paper reported a validation study of one such predictive model [10]).

As shown top right of Figure 1, STAR is appropriate in domains that lack both domain experts and data. Many domains are data starved (see Figure 2) in which case there is neither sufficient data to automatically learn predictive models .

In the completely opposite circumstance, a domain is data rich. In such domains, automatic methods can learn predictive models. There are many examples of work in this

area including [6, 14, 19, 21, 26]. We recommend automatic methods for data rich domains since manual modeling can be very expensive (for example, Fenton manually built his Bayes nets over a two year period [10]).

In the middle case of Figure 1, small amounts of local data can be used to tune either:

- Local predictive models either built manually; or
- Predictive models imported from other sites.

For example, Boehm et at. [5] 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:

$$effort = A \cdot KSLOC^{B+0.01 \cdot \sum_j \beta_j SF_j} \cdot \prod_i \alpha_i EM_i \quad (2)$$

The particular effort multipliers and scale factors recommended by Boehm et al. are shown in Figure 3. While Boehm et al offer default values for the Equation 2 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[4]. linear and exponential tuning parameters.

A problem that has been under-explored in the literature is *tuning variance*. In data starved domains, there is insufficient data to produce precise tunings. For example, At PROMISE 2005, we have reported very large tuning variance in the post-tuning values of α and β [25]. Baker [2] offers a similar finding. After thirty 90% random samples of that data, the A, B ranges found during tuning were surprisingly wide:

$$(2.2 \leq A \leq 9.18) \wedge (0.88 \leq B \leq 1.09) \quad (3)$$

We are not the only research group to be concerns about tuning variance. At PROMISE 2007, Korte & Port [17] explore the variance of automatically learned effort predictors. They comment that this variance is large enough to confuse standard methods for assessing different predictive model generators.

Since 2005 [6, 20], we have been trying to reduce tuning variance. using feature subset selection (FSS). However, despite years of work, we now report that FSS reduces but does not tame the variance of A, B, α, β.

Having failed to tame tuning variance, we have been exploring a new approach. The STAR tool [9, 23, 27]. that we describe below checks for stable conclusions within the space of possible tunings.

3. "FASTER, BETTER, CHEAPER"

This paper applies STAR to an analysis of FBC. In the 1990s, 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 new view, being pushed at NASA by Goldin, that "it's ok to fail" [33], 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 [18].

Some within NASA, like 30 year veteran Frank Hoban, supported these policies [18] who 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 generation FBC missions, were successes. This fueled enthusiasm to apply FBC across all of NASA to further reduce spending per mission as well cutting the work force by one 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% [15], 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 [29, 38].

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 damaged morale leading to a string of early retirements of veteran scientists, skilled engineers and managers [15].

Despite this, additional projects were planned including Mars Climate Orbiter and 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 [35]. This was much less than the previous Pathfinder mission (which itself cost slightly less than $300 million) and a huge reduction from the previous Viking Mars missions (cost about $935 million in 1974 Dollars, equivalent to $3.5 billion in 1997 dollars). The success of these missions would've strengthen FBC within NASA and JPL, and been seen to break new ground in terms of mission completion with the reduced staff and budget [12].

Both of these missions failed. Using a single contractor had weakened quality assurance and caused loss of vehicle. These flaws where software issues that could have easily been rectified if they had been discovered on the ground (e.g. a failure to convert from imperial to metric units, causing the loss of the Climate Orbiter [28]). The Mars Program Independent Assessment Team Report [38] 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 un-

[4]We will use uppercase B to denote the COCOMO linear tuning variable of Equation 2 and lower b to denote the business utility associated with defect predictions of Equation 1

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

der way. This mission was 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" [8, 31].

4. CASE STUDIES

Despite the above, this paper advocates FBC using the case studies of Figure 4. These studies represent the 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.

	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 4: 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 3.

Figure 4 describes the details of flight, OSP, and OSP2. Note that Figure 4 does not mention all the features listed in Figure 3 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 4, values are picked at random from the full range of Figure 3.

One aspect to note from Figure 4 is the number of open options *not* specified in the description of the projects. Some of the features in Figure 4 are known precisely (see all the features with single *values*). But many of the features in Figure 4 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. In fact, our case studies can be ranked

$$flight > OSP > OSP2$$

according to the options open to a project manager:

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

As we shall, the number of open options in a project will have an important impact on our results.

5. STAR

STAR uses Figure 4 as the inputs to a Monte Carlo simulation over a set of software models. STAR contains the COCOMO effort E estimator [5] but also the COCOMO development months M estimator [5, p29-57], and COQUALMO D defects estimator [5, p254-268], These estimator generate the $\{E, M, D\}$ variables used by Equation 1 in the introduction.

We base our analysis on COCOMO and COQUALMO for several reasons. These are mature models which have been developed, refined, and constrained over a very long period of time. The range of tuning options explored by STAR are taken from 30 years of modeling experience and regression studies of hundreds of projects [4]. COCOMO and COQUALMO have been selected and tested by a large community of academic and industrial researchers led by Boehm (this large group has meet annually since 1985). Unlike other models such as PRICE TRUE PLANNING [30], SLIM [32], or SEER-SEM [13], the COCOMO family of models are fully described in the literature. Also, at least for the effort model, there exist baseline results [7]. Further, we work extensively with government agencies writing software. Amongst those agencies, these models are frequently used to generate and justify budgets.

But the most important reason we use COCOMO & COQUALMO is that the space of possible tunings within these models is well defined. Hence, it is possible to explore the space of possible tunings. Recall from Equation 2 that the COCOMO model includes $\{A, B, \alpha, \beta\}$ tuning values. Many of these variables are shared with the COQUALMO defect predictor which also 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 3. 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 3.

Second, STAR's simulated annealer[5] seeks constraints on the project options P that most reduce the score of Equation 1 (for examples of P, see Figure 4). A particular subset of $P' \subseteq P$ is scored by using P' as inputs to the COCOMO and COQUALMO. When those predictive 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 predictive models exhibit a *keys effect* where a small number of variables control the rest [22]. 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, after simulated annealing, STAR takes a third step.

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

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

Equation 4 is a poor ranking heuristic since it is distracted by low frequency ($freq$) evidence. For example, note how the probability ($Prob$) of E belonging to the best class is moderately high even though its support is very low; i.e. $Prob(best|E) = 0.66$ but $freq(E|best) = 0.01$. To avoid such unreliable low frequency evidence, we augment Equation 4 with a support term. In Equation 5, likelihood ($like$) is chosen as our support term. Support should *increase* as the frequency of a range *increases*, i.e. $like(x|best)$ is a valid support measure since it does exactly so. High support would indicate a higher number of examples that "support" that E can be part of the best set. STAR1 hence ranks ranges via

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

After ranking members of P', STAR then imposes the top i-th ranked items of P' on the predictive 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 local tuning data. The following is a description that further details the manner in which STAR operates:

1. *SAMPLE:* To sample the ranges from the models, STAR runs the simulated annealer K_1 times. Note that here, we sample across the ranges of all the attributes. While most of the time we sample randomly across the range, we also have a heuristic optimization called extreme sampling. This form of sampling works in the following manner: for x% (x is set to 5 by default), STAR samples only the extremums of the attributes.
2. *DISCRETIZE:* The data seen in the K_1 samples is then discretized into $D = 10$ bins. Discretization converts a continuous range into a histogram with n break points $b_1 \ldots b_n$ where $(\forall i < j : b_i \leq b_j)$. After discretization, many observations can fall into the same range between b_i and b_{i+1} at frequency counts c_i. This study used equal width discretization; i.e.
$$\forall i, j : (b_i - b_{i-1}) = (b_j - b_{j-1})$$
3. *CLASSIFY:* The ranges are then classified into those seen in BEST% *best* or *rest*.
4. *RANK:* The ranges are then ranked in increasing order using Support-Based Bayesian Ranking using Equation 5.
5. *PRUNE:* Also called the back select stage. STAR runs K_2 experiments with the models where the top ranked ranges $1..X$ ranges are pre-set and the remaining ranges can be selected at random.
6. *REPORT:* STAR returns the $1..X$ settings that optimize the best for the fitness function being used according to the weights applied to effort, defects, development time, and threats. These settings are determined by iterating back from the minimum point achieved towards the first point that is statistically similar to the minimum point. This statistical difference is tested via a standard t-test.

To run our experiments, we had to apply our engineering judgment to set the parameters. The following are the default values:

$$K_1 = 10,000, K_2 = 1,000, D = 10, BEST = 10\%$$

Previously [23] we have shown that this approach (that does not use local tuning) generates estimates very similar to those generated by "LC" method proposed by Boehm (that does tune the model to local data) [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 constraining P and not T.

[5]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 [16].

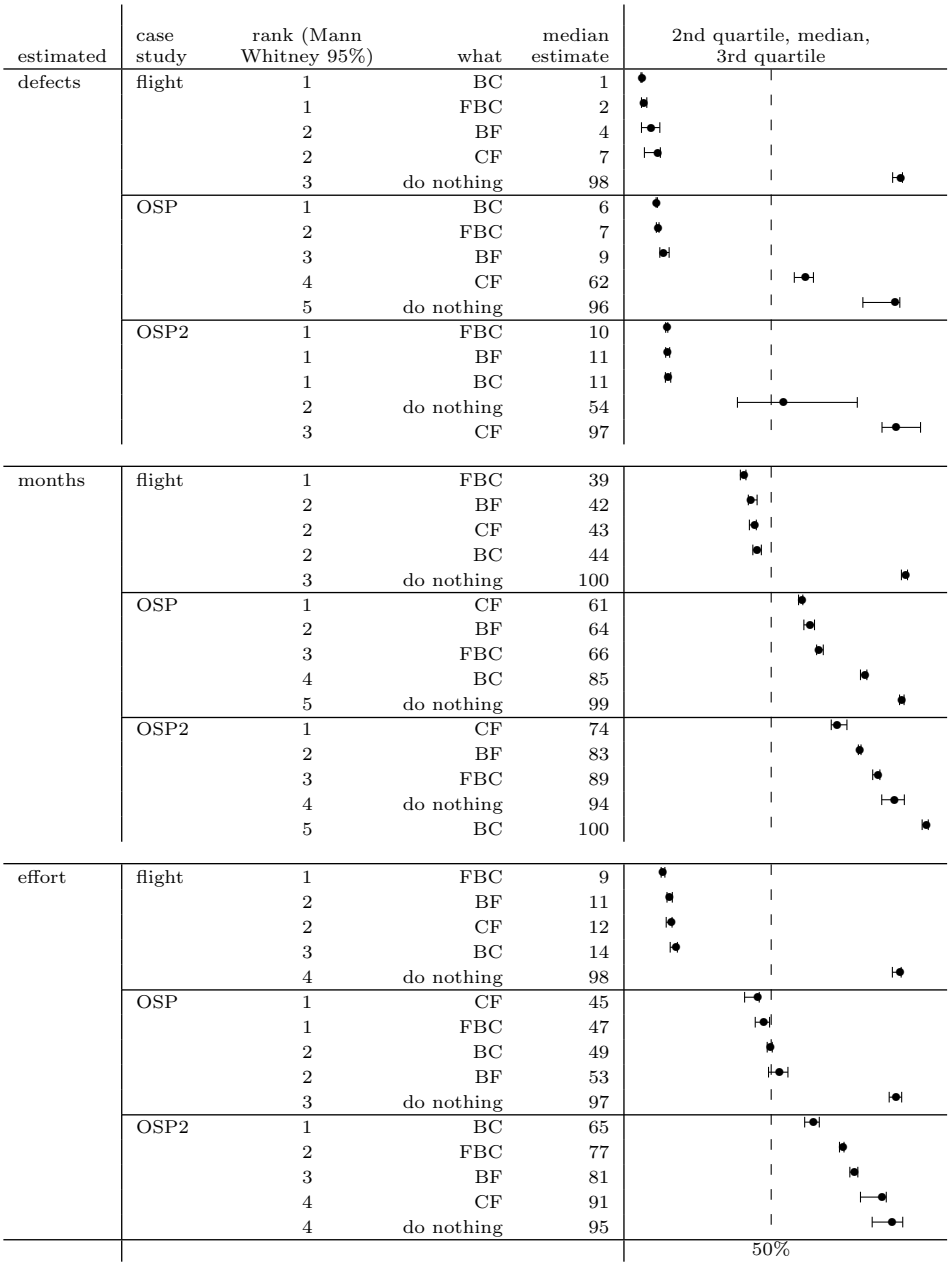

Figure 5: Results

6. RESULTS

6.1 Stability of STAR

Before we delve into the results, we need to show that any results produced by STAR are actually stable despite the stocastic nature of the algortihms used. For this purpose we use two sample projects: one that is highly constrained (OSP) and one that is loosely constrained (flight). Both projects are defined in Figure 4. STAR was run ten times for both, and then the policy results produced were compared. Each of the projects were run through STAR ten times while using the *all* strategy, where all the model features are included in the search to produce policies. Figure 6 and Figure 7 below presents the percentage of times that a certain policy appears: the higher the percentage, the more that is indicative of the stability of that policy.

As we can see, there is a high rate of stability with respect to the policies being produced by STAR for these two sample projects. Any degree of instability that is indicated is a normal occurrence given that the core algorithm used in STAR is a meta-heuristic search algorithm, and also given that we are actively varying the internal parameters of the models used.

6.2 Format of Results

Figure 5 shows the defects, months, and effort estimates seen imposing the *policy* learned by STAR:

Policy	% Used	Policy	% Used
acap = 5	100	stor = 4	70
apex = 5	100	data = 3	60
flex = 6	100	time = 3.5	60
ltex = 4	100	data = 2.5	50
pcon = 5	100	cplx = 3.5	40
plex = 4	100	*peer* = 6	40
pmat = 3	100	stor = 3.5	40
rely = 5	100	pmat = 2.5	30
resl = 6	100	data = 2	20
site = 6	100	docu = 1.5	20
team = 6	100	pvol = 2.5	20
ett = 6	90	pvol = 4	20
pcap = 5	90	rely = 4.5	20
prec = 6	90	ruse = 2.5	20
aa = 6	80	pvol = 3	10
time = 4	80		

Figure 6: Stability of the policies produced for running the flight project.

Policy	% Used	Policy	% Used
aa = 6	100	pcon = 2.5	60
acap = 3	100	prec = 1.5	60
apex = 3	100	team = 2.5	60
ett = 6	100	tool = 2.5	60
flex = 5	100	apex = 2.5	40
ltex = 4	100	time = 3	40
pcon = 3	100	aa = 5.5	30
peer = 6	100	acap = 2.5	30
pmat = 4	100	stor = 3.5	30
prec = 2	100	docu = 4	20
resl = 3	100	ett = 5.5	20
team = 3	100	ruse = 2	20
tool = 3	100	sced = 2.5	20
cplx = 5.5	90	docu = 2.5	10
cplx = 5	80	ltex = 3.5	10
pmat = 3.5	80	resl = 2.5	10
sced = 2	80	sced = 3	10
ruse = 2.5	70	time = 4	10
time = 3.5	70		

Figure 7: Stability of the policies produced for running the OSP project.

- The results have divisions for defects, months, and effort.
- Division are sub-divided into results for flight, OSP, and OSP2.

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 1:

- Results in each division are normalized 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.

6.3 Observations and Recommendations

Three aspects of Figure 5 deserve our 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. This result argues for the use of tools like STAR.

Secondly, STAR is most useful when applied to projects with many project options. As evidence of this, note how in Figure 5 that as we move from flight to OSP 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 decreases.

Thirdly, the goal of faster *and* better *and* cheaper is not overly ambitions. Prior to running these experiments, we believed that to optimize for three criteria, it may be sometimes necessary to accept non-minimal results for one of the criteria. However, contrary to our expectations, we observe that FBC achieves the best (lowest) median results in the case of:

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

In the remaining cases, FBC is statistically indistinguishable from the best (lowest) median result in the case of

- defects for flight systems and
- effort for OSP

Lastly, in the remaining cases, FBC's median is within 5% of the best (lowest) median result in the case of

- defects for OSP
- months for OSP

That is, FBC achieves minimum (or very close to minimum) values. in $\frac{7}{9}$ of Figure 5's division, Hence, we endorse STAR's default setting of $b = f = c = 1$; i.e. try to optimize on all three criteria.

7. CONCLUSION

Our results show that FBC can be achieved with minimal compromises on individual criteria. How can we reconcile this result with NASA very negative experience with FBC?

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 failures. That is, FBC was mostly a technical success, but a PR failure [36].

However, 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 5 shows the disastrous effects of CF:

- In the case of OSP CF's defects where $\frac{62}{9} = 689\%$ worse than the worst policy seen using any of BC, BF, or BFC.
- In the case of OSP2 CF's defects where $\frac{97}{11} = 881\%$ worse than the worst policy seen using any of BC, BF, or BFC.

Some of 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 [38]. This forced projects to use inexperienced managers which caused management mix ups and human error.

Like Spear [34] , 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 [33, 34].

To Spear's recommendation we would add that when applying FBC, never surrender the quest of "better". Observe how, in Figure 5, 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 $b = 1$.

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. Safe and simple software cost analysis. *IEEE Software*, pages 14–17, September/October 2000. Available from http://www.computer.org/certification/beta/Boehm_Safe.pdf.

[5] 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.

[6] 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.

[7] S. Chulani, B. Boehm, and B. Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transaction on Software Engineerining*, 25(4), July/August 1999.

[8] 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.

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

[10] N. Fenton, M. Neil, W. Marsh, P. Hearty, L. Radlinski, and P. Krause. Project data incorporating qualitative factors for improved software defect prediction. In *PROMISE'09*, 2007. Available from http://promisedata.org/pdf/mpls2007FentonNeilMarshHeartyRadlinskiKrause.pdf.

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

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

[13] R. Jensen. An improved macrolevel software development resource estimation model. In *5th ISPA Conference*, pages 88–92, April 1983.

[14] Y. Jiang, B. Cukic, T. Menzies, and N. Bartlow. Comparing design and code metrics for software quality prediction. In *Proceedings of the PROMISE 2008 Workshop (ICSE)*, 2008. Available from http://menzies.us/pdf/08compare.pdf.

[15] S. Key. Columbia, the legacy of "better, faster, cheaper"?, July 2003. http://www.space-travel.com/reports/Columbia_The_Legacy_Of_Better__Faster__Cheaper.html.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science,*

Number 4598, 13 May 1983, 220, 4598:671–680, 1983.

[17] 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.

[18] leonard david. nasa report: too many failures with faster, better, cheaper, march 2000. `http://www.space.com/businesstechnology/business/spear_report_000313.html`.

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

[20] 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`.

[21] T. Menzies, Z. Chen, D. Port, and J. Hihn. Simple software cost estimation: Safe or unsafe? In *Proceedings, PROMISE workshop, ICSE 2005*, 2005. Available from `http://menzies.us/pdf/05safewhen.pdf`.

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

[23] 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`.

[24] T. Menzies, O. Elrawas, J. Hihn, M. Feathear, B. Boehm, and R. Madachy. The business case for automated software engineerng. In *ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 303–312, New York, NY, USA, 2007. ACM. Available from `http://menzies.us/pdf/07casease-v0.pdf`.

[25] 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`.

[26] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang. Implications of ceiling effects in defect predictors. In *Proceedings of PROMISE 2008 Workshop (ICSE)*, 2008. Available from `http://menzies.us/pdf/08ceiling.pdf`.

[27] 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`.

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

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

[30] R. Park. The central equations of the price software cost model. In *4th COCOMO UsersÃŢ Group Meeting*, November 1988.

[31] 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.

[32] L. Putnam and W. Myers. *Measures for Excellence*. Yourdon Press Computing Series, 1992.

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

[34] 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.

[35] 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`.

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

[37] I. H. Witten and E. Frank. *Data mining. 2nd edition*. Morgan Kaufmann, Los Altos, US, 2005.

[38] 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.