Can We Build Software Faster and Better and Cheaper?

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

ABSTRACT

In this paper we seek to further support the viability of FBC, by analyzing its behavior in best and worst case decision making scenarios and comparing that to the other derivate "you pick two" policies. More generally, we seek to utilize the same stochastic AI tool previously used in order to determine the best and worst case scenarios for certain project test cases, all while circumventing model tuning. We find that our tool, STAR, is able to reliably determine worst case scenarios, without the need for model fine tuning. We also provide further support to the viability of FBC by observing the policy's success and failure in various test cases.

Keywords

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

1. INTRODUCTION

"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. It was later on blamed for several project failures at NASA.

Previously **[INSERT CITATION HERE]** we have explored n using STAR and other AI search methods in reusing USC COCOMO based models in the absence of sufficient data for tuning these models to the particular projects and project templates that we being studied. In these studies we've explored the our ability to make good decisions that would improve our test project by reducing the effort, defects and development time of a project.

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.

.

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, as well as the STAR system. Previous results presented in [27] suggested that (a) FBC is in fact a viable development methodology and (b) suggested that when applied properly, can indeed produce similar or better results compared to the "you pick two" policies in a variety of project cases.

Building on that study, this paper presents this new study of FBC using the Previously used AI tool "STAR" **[INSERT CITATION HERE]**. It is found that, even if FBC is applied properly to a project, project managers need to be weary of making bad decisions. Also, bad decisions that are made while applying FBC affect every aspect of the project, increasing the damage done to a project. In addition to all that, we show that STAR is able to reliably show which combinations of policies are the worst for a certain project, showing that not only are we capable of getting the best policies, but also the worst of them using stochastic AI methods applied to data starved software engineering models with medium granularity.

Previous papers by Menzies et. al explored the validity of the general consensus on "Faster, Better, Cheaper? Pick any two". It was found that "Faster, Better, Cheaper" (FBC) is indeed feasible when maintaining a balanced concern and concentration on the quality aspects of a project. This paper extends previous publications by the authors **[INSERT CITATION HERE]** in several ways:

-
-
-

The following sections are organized in the following fashion:

2. "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" [36], 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 [9].

Some within NASA, like 30 year veteran Frank Hoban, supported these policies [9] 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% [17], 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 [38, 39].

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 [17].

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 [37]. 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 [29]). The Mars Program Independent Assessment Team Report [39] found that these missions were under-staffed, underfunded 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 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 [28].

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, 14]. It was these criticisms that turned NASA away from FBC towards policies concentrating on two of the three aspects of FBC.

3. BACKGROUND

STAR is a natural tool for assessing policies produced under FBC, since it uses COCOMO based software engineering models that have already been used 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).
- 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) using the following formula:

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

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. $b = f = c = 1$.

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 11. 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[1]. 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

[1]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

reported very large tuning variance in the post-tuning values of α and β [24]. 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 [19] 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 [10, 22, 25]. that we describe below checks for stable conclusions within the space of possible tunings.

4. STAR

STAR uses Figure 1 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 [31], SLIM [33], or SEER-SEM [16], 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 11.

Second, STAR's simulated annealer[2] seeks constraints on the project options P that, normally, most reduce the score of Equation 1 (for examples of P, see Figure 1). For finding the worst policies, STAR had to be slightly adapted, where lookied for the policies that most increased Equation 1. 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 [21]. 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:

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

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

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

After ranking members of P', STAR imposes the top i-th ranked items of P' as model inputs, then runs 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*.

[2]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 suboptimal 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 [18].

Note that STAR constrains the project options P but not the tuning options T. Hence, STAR's generated *policy* contains subsets of the project options P that most improve the score, despite variations in the tunings T. This approach means we can reuse COCOMO models without using 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 [22] 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. CASE STUDIES AND METHODS

We use two categories of projects in this paper to study the failure of of FBC. The one category includes three projects of various sizes. Designated *Small*, *Medium*, and *Large*, they corresponded to projects of size ranges [7,13] kloc, [70,130] kloc, and [700,1300] kloc respectively. These projects are open ended and are used to study whether there is an influence of project size purely on our study. Open ended here meaning that there are no project limitations that are predetermined but that the only limitation is the default limits of the models and the size of the projects.

	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		
	pvol	2	4		
	acap	3	5		
	apex	2	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	KSLOC	7	418		
JPL ground software	rely	1	4	tool	2
	data	2	3	sced	3
	cplx	1	4		
	time	3	4		
	stor	3	4		
	pvol	2	4		
	acap	3	5		
	apex	3	5		
	pcap	3	5		
	plex	1	4		
	ltex	1	4		
	pmat	2	3		
	KSLOC	11	392		

Figure 1: 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 11.

The second category are the NASA project templates that we've previously used, and are presented in Figure 1. These studies represent the NASA software, at increasing levels of specificity:

- *Flight* is a general description of flight software at NASA's Jet Propulsion Laboratory.
- *Ground* is a general description of ground 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.

Large		Medium		Small	
Policy	% Used	Policy	% Used	Policy	% Used
acap=5.0	100	acap=5.0	100	pcap=5.0	100
apex=5.0	100	apex=5.0	100	acap=5.0	100
flex=6.0	100	auto=6.0	100	apex=5.0	100
ltex=5.0	100	flex=6.0	100	flex=6.0	100
pcap=5.0	100	ltex=5.0	100	ltex=5.0	100
pcon=5.0	100	pcap=5.0	100	pcon=5.0	100
plex=5.0	100	pcon=5.0	100	plex=5.0	100
pmat=6.0	100	plex=5.0	100	prec=6.0	100
prec=6.0	100	pmat=6.0	100	sced=1.0	100
resl=6.0	100	prec=6.0	100	sced=1.5	100
site=6.0	100	resl=6.0	100	site=6.0	100
team=6.0	100	sced=1.0	100	team=6.0	100
sced=1.0	90	sced=1.5	100	auto=6.0	90
auto=6.0	80	site=6.0	100	pmat=6.0	90
sced=1.5	80	team=6.0	100	resl=6.0	90
ett=6.0	60	peer=6.0	80	cplx=1.5	80
peer=6.0	60	rely=5.0	80	ett=6.0	80
rely=5.0	60	cplx=1.5	70	docu=1.0	70
		tool=5.0	70	cplx=1.0	60
		data=2.5	60	data=2.5	60
		docu=1.5	60	peer=6.0	60
		ett=6.0	60	sced=2.0	60
		docu=1.0	50	docu=1.5	50
				pvol=2.0	50
				pvol=2.5	50
				rely=5.0	50
				time=3.5	50
				tool=5.0	50

Figure 3: The most common policies selected as good policies to apply. Note that policies picked less than 50% of the time have been ommitted.

Figure 1 describes the details of flight, ground, OSP, and OSP2. Note that Figure 1 does not mention all the features listed in Figure 11 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 1, values are picked at random from the full range of Figure 11.

One aspect to note from Figure 1 is the number of open options *not* specified in the description of the projects. Some of the features in Figure 1 are known precisely (see all the features with single *values*). But many of the features in Figure 1 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

$$ground \equiv flight > OSP > OSP2$$

according to the options open to a project manager:

- In the case of flight and ground systems, the description is very general and managers have many options.
- In the case of OSP2, most of the project options are predetermined 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.

The results of our study are presented in the following sections. We present two separate results:

1. The first of these results show the stability of our AI methods in producing results in this study, in addition to presenting the ability of these policies to control the projects that they are selected for. A summary of the best and worst selected policies are also presented.
2. The second result presents the potential for FBC to succeed or fail and compares that potential with that of other "you-pick-two" strategies in the form of quartile charts and Mann-Whitney rankings.

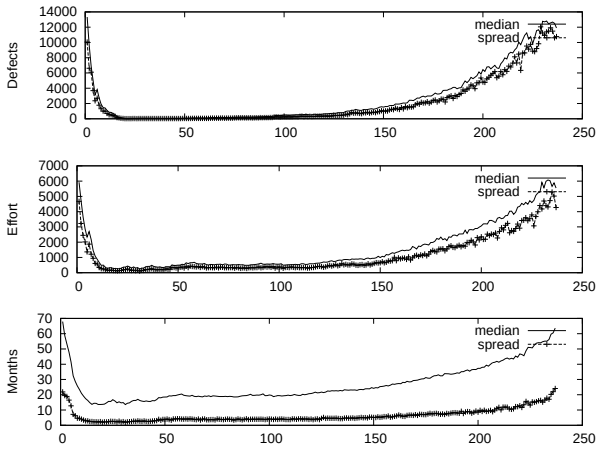


Figure 6: An example of the results generated for large projects while trying to find the best policies (FBC applied).

6. RESULTS

6.1 Stability, Control and Validity

In order to determine the stability of STAR in choosing the worst policies, three open ended projects, of different sizes, were created. We ran the three of those in STAR using FBC and its "pick two" variants, with each case running ten times to account for the random variation in the results. This was repeated twice: one with the intention of choosing the best policies, and the other the worst policies. This was repeated for our NASA projects as well, except each case was repeated twenty times. Figures 2 through 5 present the stability results. These figures present the percentage of times that policies appear in the set of chosen policies, whether looking for the best or the worst policies.

The results shown confirm previous results concerning the stability in selecting the best policies, and also suggest that this stability is also available when choosing the worst policies. This is indicated by the large amount of policies that are chosen at a high percentage rate. However, the policy generation is not equally stable across the board. In the case of the open ended projects, the results presented in Figure 2 and Figure 3 for the *Small* project seem to be less stable, generating a larger range of policies compared to *Medium* and *Large*. This makes sense in small open ended projects, as the project would be more sensitive to the small details within a project, which is reflected in the models as well. Take the case of the COCOMO II model: Equation 2 shows that when the size (KLOC) of the project decreases, the effort multipliers will inherently have a larger relative effect on the effort of a project. On the other hand, if the size was large, the scale factors would have a much larger effect, meaning that the effort multipliers that would be included in the policies need to have a large effect on the project to "make the grade".

In the case of stability for the NASA projects, we see a different stability result in Figure 4 and Figure 5, where the more specific

Large		Medium		Small			
Policy	% Used	Policy	% Used	Policy	% Used	Policy	% Used
acap=1.0	100	acap=1.0	100	acap=1.0	100	auto=2.0	80
apex=1.0	100	apex=1.0	100	apex=1.0	100	data=4.5	80
auto=1.0	100	auto=1.0	100	auto=1.0	100	pvol=4.5	80
ett=1.0	100	ett=1.0	100	ett=1.0	100	time=5.5	80
flex=1.0	100	flex=1.0	100	ltex=1.0	100	auto=1.5	70
ltex=1.0	100	ltex=1.0	100	pcap=1.0	100	docu=4.5	70
pcap=1.0	100	pcon=1.0	100	pcon=1.0	100	stor=5.5	70
pcon=1.0	100	peer=1.0	100	peer=1.0	100	cplx=5.5	60
peer=1.0	100	plex=1.0	100	plex=1.0	100	ett=1.5	60
plex=1.0	100	pmat=1.0	100	prec=1.0	100	ett=3.0	60
pmat=1.0	100	prec=1.0	100	rely=1.0	100	time=5.0	60
prec=1.0	100	rely=1.0	100	resl=1.0	100	auto=3.0	50
rely=1.0	100	resl=1.0	100	sced=4.0	100	data=4.0	50
resl=1.0	100	sced=4.5	100	sced=4.5	100	ett=2.0	50
site=1.0	100	sced=5.0	100	sced=5.0	100	ett=2.5	50
team=1.0	100	site=1.0	100	site=1.0	100	peer=2.0	50
sced=4.5	90	team=1.0	100	team=1.0	100	plex=1.5	50
tool=1.0	80	tool=1.0	100	flex=1.0	90	ruse=5.5	50
sced=5.0	50	pcap=1.0	90	pmat=1.0	90		
		sced=4.0	70	tool=1.0	90		

Figure 2: The most common policies selected as bad policies to apply. Note that policies picked less than 50% of the time have been ommitted.

flight		ground		OSP		OSP2	
Policy	% Used	Policy	% Used	Policy	% Used	Policy	% Used
acap=3.0	100	acap=3.0	100	docu=2.0	100	ltex=2.0	100
auto=1.0	100	pmat=2.0	100	ett=1.0	100	auto=1.0	100
plex=1.0	100	auto=1.0	100	ltex=2.0	100	ett=1.0	100
ltex=1.0	100	pcon=1.0	100	acap=2.0	100	peer=1.0	100
pcon=1.0	100	ltex=1.0	100	apex=2.0	100	prec=3.0	100
site=1.0	100	site=1.0	100	pcon=2.0	100	pmat=4.0	95
pcap=3.0	100	flex=1.0	100	auto=1.0	100	docu=3.0	95
resl=1.0	100	resl=1.0	100	prec=1.0	100	sced=2.0	75
ett=1.0	100	ett=1.0	100	resl=1.0	100	pmat=4.5	50
flex=1.0	100	pcap=3.0	100	tool=2.0	100		
rely=3.0	100	team=1.0	100	team=2.0	100		
apex=2.0	100	apex=3.0	100	peer=1.0	100		
team=1.0	100	prec=1.0	100	pmat=1.0	100		
prec=1.0	100	peer=1.0	100	sced=1.0	80		
peer=1.0	100	plex=1.0	95	flex=2.0	75		
pmat=2.0	90	rely=1.0	55	cplx=5.0	60		
stor=3.0	50						

Figure 4: The most common policies selected as bad policies to apply for NASA projects. Note that policies picked less than 50% of the time have been omitted.

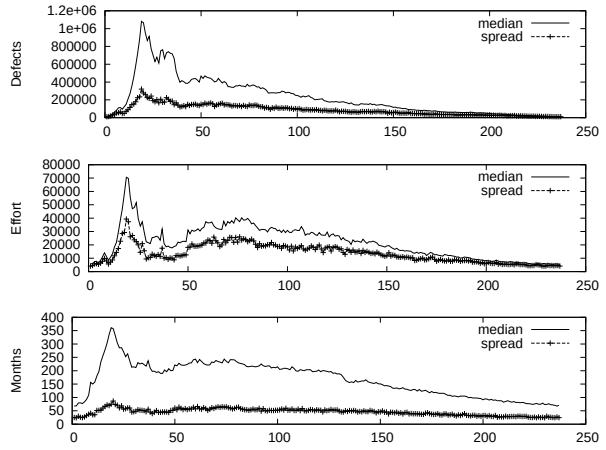


Figure 7: An example of the results generated for large projects while trying to find the worst policies (FBC applied).

a project is the more restricted and stable are its results, for both searching for good and bad policies. The reason to this is that more specified projects tend have a smaller search space, causing smaller policy sets to created by STAR.

We also need to study whether STAR is able to control projects and guide the to the end that it intends, whether good or bad. Figure 6 and Figure 7 are examples of the STAR runs, both conducted for large projects with an FBC focused evaluation function. The results from these runs are clumped with the results from other runs in order to obtain clearer overall results. The first corroborates results from previous papers we've published previously ([23]), and choses the best policies for reducing overall defects, effort and development time. Note here that since the projects being used are open ended, this causes larger policies to me generated. We can see however that even for such open ended projects, STAR is able to control the project well, demonstrated by the large "dip" in the graphs of Figure 6. On the other hand, Figure 7 shows the policies that are selected by star to be the worst in one of the runs.

The purpose of this experiment is determining the policies that need to be avoided, while at the same time showing how bad a project can be mishandled in the worst case scenario. This scenario is demonstrated by the spike that appears in the graphs of

flight		ground		OSP		OSP2	
Policy	% Used	Policy	% Used	Policy	% Used	Policy	% Used
pcon=5.0	100	plex=4.0	100	sced=1.0	100	peer=6.0	100
prec=6.0	100	resl=6.0	100	apex=3.0	100	auto=6.0	100
pmat=3.0	100	flex=6.0	100	team=3.0	100	pmat=5.0	100
ltex=4.0	100	team=6.0	100	peer=6.0	100	ett=6.0	100
site=6.0	100	prec=6.0	100	ltex=4.0	100	ltex=5.0	100
flex=6.0	100	acap=5.0	100	auto=6.0	100	prec=5.0	100
acap=5.0	100	pcon=5.0	100	resl=3.0	100	docu=4.0	95
resl=6.0	100	ltex=4.0	100	prec=2.0	100	sced=2.5	70
plex=4.0	100	apex=5.0	100	flex=5.0	100		
apex=5.0	100	site=6.0	100	acap=3.0	100		
team=6.0	100	pcap=5.0	100	pcon=3.0	100		
pcap=5.0	95	pmat=3.0	95	ett=6.0	100		
data=3.0	90	time=3.5	85	pmat=4.0	100		
time=3.5	90	ett=6.0	75	tool=3.0	90		
stor=4.0	85	stor=3.5	65	sced=1.5	65		
pmat=2.5	80	data=2.5	65	cplx=6.0	60		
data=2.5	80	stor=4.0	60	apex=2.5	60		
time=4.0	70	data=3.0	60	prec=1.5	60		
rely=5.0	70	rely=4.0	60	tool=2.5	60		
peer=6.0	70	time=4.0	60	cplx=5.5	60		
stor=3.5	65	pmat=2.5	60	acap=2.5	55		
ett=6.0	55	peer=6.0	55	time=3.5	55		
auto=6.0	50	auto=6.0	55	pcon=2.5	50		
				team=2.5	50		

Figure 5: The most common policies selected as good policies to apply for NASA projects. Note that policies picked less than 50% of the time have been ommitted.

Attributes		Ratings: Large						Medium						Small					
		1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Scale Factors (COCOMO II)	prec	△					▲	△					▲	△					▲
	flex	△					▲	△					▲	△					▲
	resl	△					▲	△					▲	△					▲
	team	△					▲	△					▲	△					▲
	pmat	△					▲	△					▲	△					▲
Effort Multipliers (COCOMO II)	rely	△				▲		△				▲		△				▲	
	data									▲						▲	△	△	
	cplx								▲					▲	▲				△
	ruse																		△
	docu							▲	▲					▲	▲			△	
	time																▲	△	△
	stor																		△
	pvol														▲	▲		△	
	acap	△				▲		△				▲		△				▲	
	pcap	△				▲		△				▲		△				▲	
	pcon	△				▲		△				▲		△				▲	
	apex	△				▲		△				▲		△				▲	
	plex	△				▲		△				▲		△	△			▲	
	ltex	△				▲		△				▲		△				▲	
	tool	△						△				▲		△				▲	
	site	△					▲	△					▲	△					▲
	sced	▲	▲			△		▲	▲		△	△		▲			△	△	
Defect Removal (COQUALMO)	auto	△					▲	△					▲	△	△	△			▲
	peer	△					▲	△					▲	△	△				▲
	ett	△					▲	△					▲	△	△	△			▲

Figure 8: Policy ranges open ended: This table shows the ranges of policies for the open ended size projects. The gray filled cells represent the allowed range, and "▲" and "△" the policies chosen while searching for good and bad policies respectively.

Figure 7. The fact that STAR is able to produce such a clear spike when searching for the worst policies demonstrates that it is producing a set of policies which produce definitive worst results for that project.

The stability results presented in figures 2 to 5 are presented in a different manner in figures 8 to 10. Figure 8 presents the stable policies fort the open ended projects. The shaded cells indicated the range for the particular attribute in that row, which in this case are the default values indicated in the COCOMO II model. Also "▲" and "△" show the locations of the best and worst stable policy respectively. Figure 9 and Figure 10 represent the stable policies for the NASA projects. These stable results have been made to overlap with the results of medium size projects, since most of the NASA projects fall in that category. The light gray cells indicate the model ranges that aren't within the project range, while the dark gray cells indicate the ranges of attributes within the project. "▲" and "△" represent the open ended medium-sized project's stable results, while "■" and "□" represent the NASA project's stable results, best and worst respectively.

We ask the question of whether an AI tool that covers the full ranges of attributes is necessary or not. Other works such as the work of [25] seem to suggest that the nature of the models in use al-

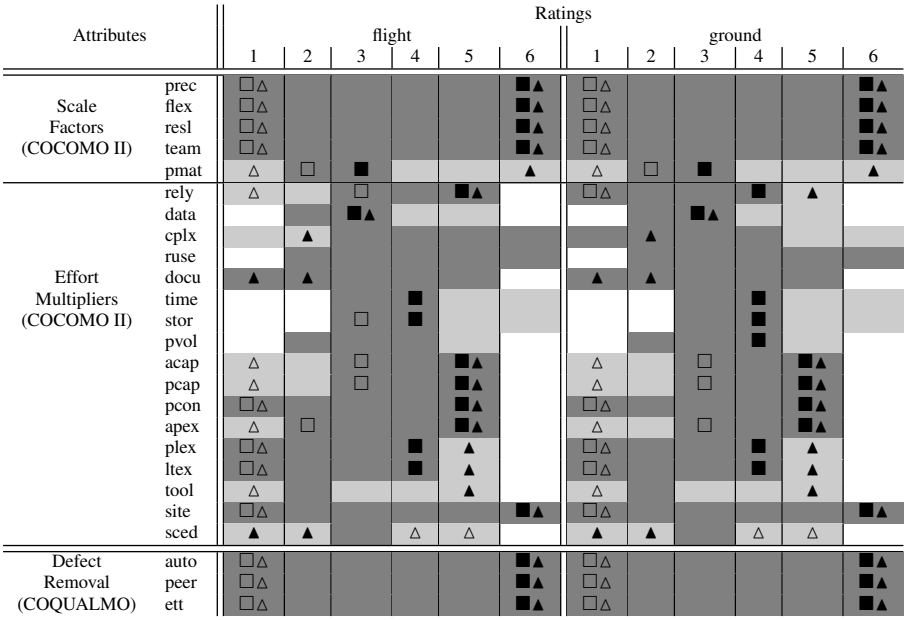


Figure 9: Policy ranges part 1: This table shows the ranges for flight and ground. The dark gray filled cells represent the allowed range, and "■" and "□" the policies chosen while searching for good and bad policies respectively. This table is overlapped over the results for medium projects from Figure 10.

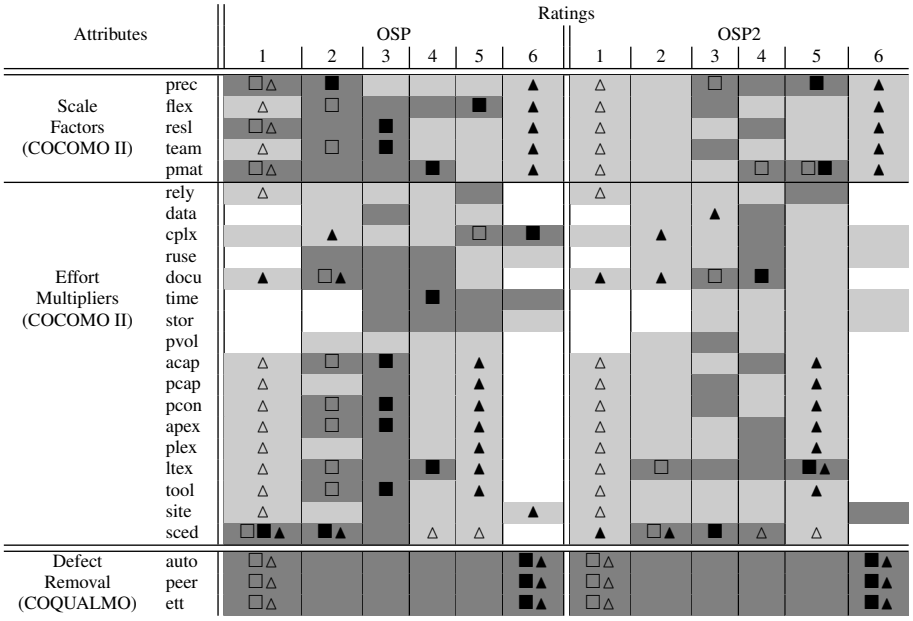


Figure 10: Policy ranges part 2: This table shows the ranges for OSP and OSP2. The dark gray filled cells represent the allowed range, and "■" and "□" the policies chosen while searching for good and bad policies respectively. This table is overlapped over the results for medium projects from Figure 10.

low us to simply overlook the ranges of the attributes while simply concentrating on the maxima and minima of the attributes. Looking at Figure 8, we can see that there is support for this operating assumption made by SEESAW [25, 26] in that most of the attributes (mainly those attributes that are proportional to COCOMO II Effort) seem to to propagate to either extreme. Note however that there is also counter evidence to that is presented in these tables. The obvious evidence is that fact that there seems to be policies that aren't on the extreme, especially for the medium and small open-ended projects. This seems to suggest that there is a need to explore the full ranges of the attributes in order to obtain the best policies. Another piece of evidence is the fact that there are several attributes that aren't set in the medium and large projects. This indicates that sometimes the whole range of an attribute can be ignored, includ-

ing the extremes. So not only is there a need to evaluate the whole of the range to evaluate what value of the attribute is needed, but also to evaluate whether a value needs to be set for that attribute.

6.2 Success/Failure Potential of FBC

In this section, we will present the our observations concerning the potential for FBC to succeed and to fail compared to other "you pick two" policies mentioned in §2. These observations are based on the quartile charts and Mann-Whitney ranking presented in figures 14 through 13. These charts show the normalized median results, as well as the second and third quartiles, accompanied by the ranking of each of the policy schemes. Some of the results presented here reiterate results that were mentioned in O. El-Rawas' thesis [10].

We Start with the results for applying the best policies presented in Figure 12 and Figure 13.The former present the quartile charts and rankings for FBC and its derivate policies for the open ended projects, and the latter for the NASA projects. When observing these figures, we can see that FBC does well in general, especially compared to "do nothing'. "do nothing" is a baseline result that is produced for each of the projects without applying any policies to them, and is supposed to represent how the project would end up given its current shape and assuming all future decisions are randomly chosen. The results presented in the aforementioned figures show the FBC rarely out does its derivate policies, however it does manage to stay close in terms of results, ranking close to the top according to the Mann-Whitney rankings, and rarely ranking last behind the "you pick two" policies. This supports the conclusion presented in [10]: "BFC is a 'jack of all trades, master of none'. It works well in most instances, and provides a compromise solution in balancing out model estimates." This result in addition suggests that FBC is a viable set of priorities to follow that does not have a catastrophic cost side effects. Furthermore, FBC is capable of avoiding model and project specific catastrophic side effects. Some examples of such side effects include the following:

- For the open-ended projects in Figure 12, as well as OSP and OSP2 in Figure 13, BC fails badly with regards months (development time), even though it ranks first and has the lowest normalized median estimates for defects in all the cases but one. Infact, BC does worse than "do nothing" with OSP2 months, which is to say that development time would've been better off in that case had we done nothing.
- For OSP and OSP2 in Figure 13, CF fails badly with defects introduced, despite the fact that it ranks at the top with months and effort in 3 of the 4 projects.

One interesting occurrence that we notice is that BF doesn't seem to fail catastrophically. In this case, we argue that this is due to the nature of the relationship between the effort and the months models. More specifically, it is due to the fact that effort (or rather the vast majority of the effort calculation) is included in the derivation equations of months. This infers that by reducing months we are subsequently reducing effort, but reducing effort doesn't necessarily reduce months. In other words, BF in this case is acting as a pseudo-FBC due to this inter-model dependence.

Having discussed the results and analysis when applying the best policies, we move on to the case where we apply the worst policies. The reason this is done is to find the worst case scenario when applying a priority policy such as FBC, and the results are presented in Figure 14 and Figure 15. "do nothing", as presented previously, is a baseline that is supposed to represent how the project would end up given its current shape and assuming all future decisions are randomly chosen. Since we are applying the worst policies, leaving a project as is will naturally be better than applying such policies. Hence the result where "do nothing" does best in the vast majority of cases presented in this paper. By finding the worst scenario for FBC and the "you pick two" policies, we try to find the potential damage that following these policies could cause on a software project. The following observations can be made about the figure results:

- In the case of CF, CF seems to have the least damage potential of all the policies when it comes to defects. In one case it even has less potential damage pertaining to defects than "do nothing". This would make sense since CF has no control over defects of a project. Another such case is effort for OSP2, which can be justified by the fact OSP2 is a very well defined and restricted project, meaning that CF isn't able to control it as much as in the cases of flight and ground for effort.
- In the case of BC, it exhibits lower damage with regards to months for all the open ended projects, OSP and OSP2. we can use the same argument as CF to justify this, i.e. BC does not deal with or attempt to control the development time of a project.
- BF doesn't seem to exhibit the behavior of both CF and BC. This can be justified by the same reasoning as above, where the relationship between the effort and months models allows BF to have an effect that is close to that of FBC.

In general the results for the worst case seem to reflect those for the best case. This seems to infer that exploring this maximum potential damage also tells about the control that a policy has over certain aspects of the project's performance. In addition to this, the results presented here indicate that the failure behavior of FBC is very similar to the derivate policies, ranking mid-pack most of the time. This implies that, despite the larger feature coverage of FBC, the potential for FBC to fail isn't any higher. This reinforces our previous conclusion concerning the viability of FBC, where the use of FBC doesn't seem to threaten a software project with a larger failing potential in the case that bad decisions are made.

7. DISCUSSION AND VALIDITY

The results presented in §6.2 of this paper can be argued against in various ways. All the analysis in this paper is done under the assumption that COCOMO and its related models are models that are able to provide us with close estimates, and we truly know the tuning limits of this model. The fact is the limits we impose are based on a limited dataset of projects sourced from a limited set of development environments, mainly NASA. The other issue here that comes up is that these models are in some cases related, bringing the study of prioritizing development time (faster), defects introduction (better), or person-months effort (cheaper) into question since the models that measure these aspects aren't independent of each other. Or perhaps this is saying something more profound: that we cannot solely prioritize one or two aspects of a project while ignoring the third without considering the relationships between the models. Either way, we run into the issue of dealing with models which may or may not accurately represent aspects of a software project

8. FUTURE WORK

Given the above discussion concerning the validity of the models, one possible future direction could be one where we attempt to

work independent of models, and move closer to case base reasoning (CBR) approaches. One way to do this could be using Figure 9 and Figure 10 and deriving a distance measure between a given project and one that's in the database with known real world or simulated values. This would allow us to circumvent the process of simulating the given project under certain models. What we have done so far for evaluating a project is to:

1. randomly setting undetermined settings in a project to a value that is within project parameters
2. running a simulation of the project
3. repeating 1 and 2 several thousand times in order to account for random variance, and obtaining the median values.

By attempting to change our approach to one closer to CBR, we could break free of our model dependence, while at the same time reducing the amount of computation and verification needed potentially giving us significant speedups, as well as additional dataset exposure.

9. RELATED WORK

Much of the related work on uncertainty in software engineering uses a Bayesian analysis. For example, Pendharkar et.al. [32] demonstrate the utility of Bayes networks in effort estimation while Fenton and Neil explore Bayes nets and defect prediction [11] (but unlike this paper, neither of these teams links defect models to effort models). We elect to take a non-Bayesian approach since most of the industrial and government contractors we work with use parametric models like COCOMO.

The process simulation community (e.g. Raffo [35]) studies models far more elaborate than COCOMO or COQUALMO. While such models offer more detailed insight into an organization, the effort required to tune them is non-trivial. For example, Raffo spent two years tuning one such model to one particular site [34].

Other related work is the search-based SE approach advocated by Harmon [13]. Search-Based Software Engineering (SBSE) uses optimization techniques from operations research and meta-heuristic search (e.g. simulated annealing and genetic algorithms) to hunt for near optimal solutions to complex and over-constrained software engineering problems. The SBSE approach can and has been applied to many problems in software engineering (e.g. requirements engineering [15]) but most often in the field of software testing [1]. Harmon's writing inspired us try simulated annealing to search the what-ifs in untuned COCOMO models [23]. However, we found that SEESAW ran much faster and produced results with far less variance than simulated annealing. In addition the the above related work, a recent ICSP'09 paper by Orrego et. al [30] has explored the merits of software reuse in the context of NASA projects using the same USC models that we use in this paper. For that study the same stochastic AI methods were used and showed that software reuse doesn't always benefit a project, recommending a case by case evaluation of the appropriateness of software reuse.

10. CONCLUSION

Having discussed the concept of "Faster, Better, Cheaper" and its history, we presented a stochastic AI tool which we used to run several simulation in the absence of proper model tuning. We have shown that the results produced by this tool are stable and reliable, and proceeded to use this tool to find best and worst case scenarios with the use of FBC.

With the help of quartile charts, we were able to show that FBC is a viable option that doesn't exhibit catastrophic consequences with regards to any of the models used in this paper. We also found that other "you pick two" policies exhibit such consequences. In addition, they have varying behavior depending on the aspect of the project being discussed and the project itself, while FBC has a more uniform response.

Our final recommendation is that FBC is indeed a viable option if it is implemented properly, that despite having lost credibility in software development circles.

APPENDIX

A. REFERENCES

[1] J. Andrews, F. Li, and T. Menzies. Nighthawk: A two-level genetic-random unit test data generator. In *IEEE ASE'07*, 2007. Available from http://menzies.us/pdf/07ase-nighthawk.pdf.

[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] L. David. NASA report: Too many failures with faster, better, cheaper, march 2000. http://www.space.com/businesstechnology/business/spear_report_000313.html.

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

[11] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999. Available from http://citeseer.nj.nec.com/fenton99critique.html.

[12] M. hardin. Mars climate orbiter nearing sept. 23 arrival, September 1999. JPL Universe, Vol. 29, No. 19, http://www.jpl.nasa.gov/info/universe/un990917.pdf.

[13] M. Harman and J. Wegener. Getting results from search-based approaches to software engineering. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 728–729, Washington, DC, USA, 2004. IEEE Computer Society.

[14] IFPTE. IFPTE report on the effectiveness of nasa's workforce & contractor policies, March 2003. http://www.spaceref.com/news/viewsr.html?pid=10275.

[15] O. Jalali, T. Menzies, and M. Feather. Optimizing requirements decisions with keys. In *Proceedings of the PROMISE 2008 Workshop (ICSE)*, 2008. Available from http://menzies.us/pdf/08keys.pdf.

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

[17] 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.

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

[19] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] T. Menzies, S. Williams, O. El-rawas, 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.

[26] T. Menzies, S. Williams, O. Elrawas, D. Baker, B. Boehm, J. Hihn, K. Lum, and R. Madachy. Accurate estimates without local data? *Software Process Improvement and Practice (to appear)*, 2009.

[27] T. Menzies, S. Williams, O. ElRawas, B. Boehm, and J. Hihn. How to avoid drastic software process change (using stochastic stability). In *International Conference on Software Engineering*, 2009. to be published in ICSE'09.

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

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

[30] A. Orrego, T. Menzies, and O. El-Rawas. On the relative merits of software reuse. In *International Conference on Software Process*, 2009. Available from http://menzies.us/pdf/09reuse.pdf.

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

[32] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger. A probabilistic model for predicting software development effort. *IEEE Trans. Softw. Eng.*, 31(7):615–624, 2005.

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

[34] 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.

[35] D. Raffo and T. Menzies. Evaluating the impact of a new technology using simulation: The case for mining software repositories. In *Proceedings of the 6th International Workshop on Software Process Simulation Modeling (ProSim'05)*, 2005.

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

[37] D. Tuite. Better, faster, cheaper-pick 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`.

[38] N. watch. Faster - better - cheaper under fire, 2003. `http://www.nasawatch.com/fbc.html`.

[39] 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.

B. QUARTILE CHARTS

	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

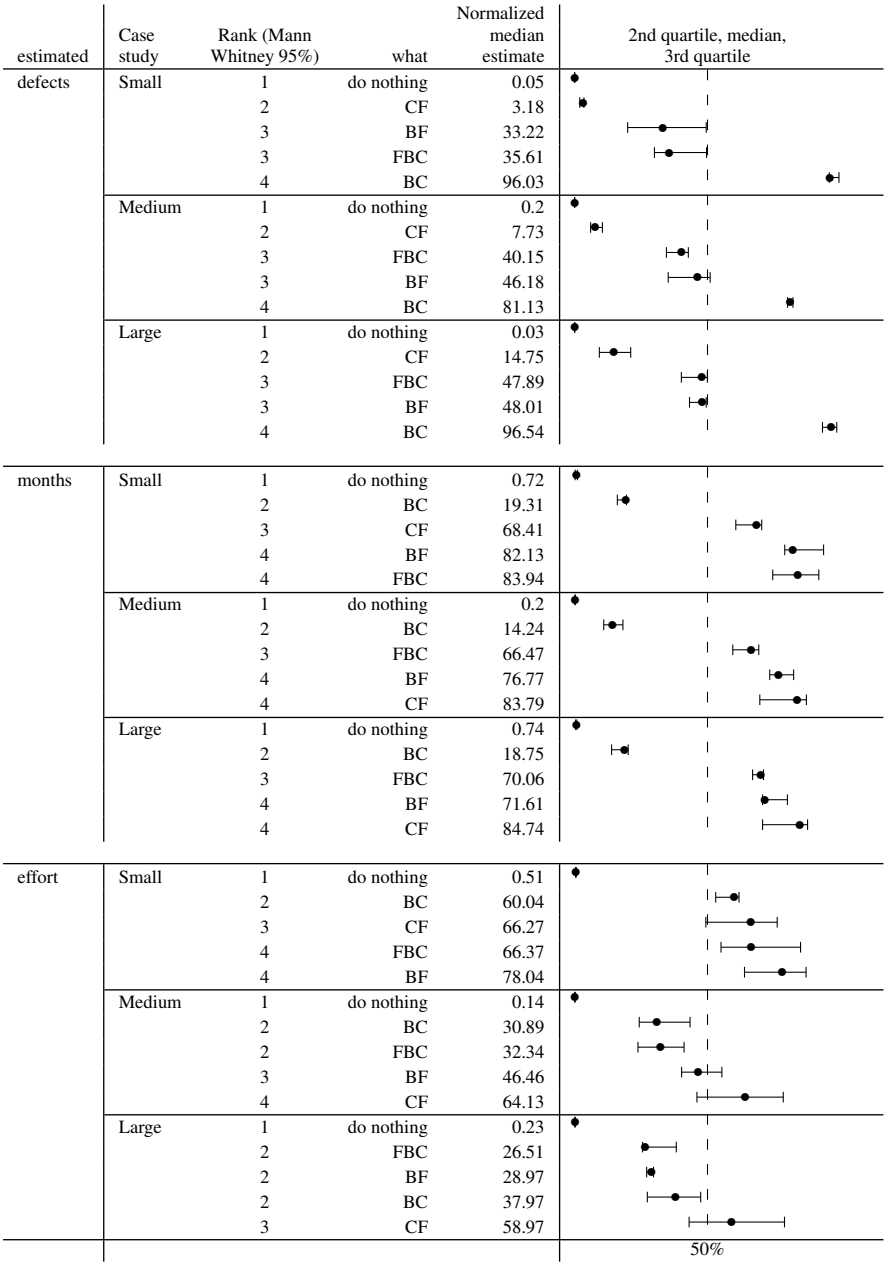
Figure 11: Features of the COCOMO and COQUALMO models used in this study.

estimated	Case study	Rank (Mann Whitney 95%)	what	Normalized median estimate	2nd quartile, median, 3rd quartile
defects	Small	1	BC	0.14	
		1	CF	0.22	
		1	BF	0.37	
		1	FBC	0.37	
		2	do nothing	82.61	
	Medium	1	BC	0.08	
		2	BF	0.16	
		3	FBC	0.28	
		4	CF	1.14	
		5	do nothing	96.02	
	Large	1	BC	0.01	
		2	BF	0.38	
		2	CF	0.43	
		2	FBC	0.85	
		3	do nothing	95.62	
months	Small	1	CF	2.01	
		1	BF	3.02	
		1	FBC	3.02	
		2	BC	32.66	
		3	do nothing	64.32	
	Medium	1	CF	1.8	
		1	BF	2.52	
		1	FBC	3.6	
		2	BC	40.65	
		3	do nothing	94.96	
	Large	1	CF	2.57	
		1	BF	5.48	
		2	FBC	7.53	
		3	BC	38.36	
		4	do nothing	97.09	
effort	Small	1	CF	0.74	
		2	FBC	1.47	
		3	BF	2.33	
		4	BC	4.3	
		5	do nothing	77.03	
	Medium	1	CF	0.81	
		1	BF	0.91	
		2	FBC	1.92	
		3	BC	2.67	
		4	do nothing	96.76	
	Large	1	CF	1.58	
		2	BF	2.77	
		3	BC	2.9	
		4	FBC	3.83	
		5	do nothing	95.14	
					50%

Figure 12: Results for applying the best policies for the different target functions of FBC. Note that the mean values are normalized.

estimated	Case study	Rank (Mann Whitney 95%)	what	Normalized median estimate	2nd quartile, median, 3rd quartile
defects	flight	1	BC	0.33	
		2	FBC	1.43	
		2	CF	1.7	
		2	BF	3.11	
		3	do nothing	91.11	
	ground	1	BC	0.34	
		2	BF	0.42	
		3	FBC	0.47	
		3	CF	0.67	
		4	do nothing	87.75	
	OSP	1	BC	1.18	
		2	FBC	2.9	
		3	BF	3.28	
		4	CF	58.36	
		5	do nothing	96.64	
	OSP2	1	FBC	0.34	
		1	BF	0.67	
		2	BC	4.61	
		3	CF	49.69	
		4	do nothing	62.71	
months	flight	1	CF	11.2	
		1	BF	11.62	
		1	FBC	11.62	
		2	BC	16.6	
		3	do nothing	95.02	
	ground	1	CF	8.46	
		2	BF	11.44	
		2	FBC	11.44	
		3	BC	15.42	
		4	do nothing	95.52	
	OSP	1	CF	16.85	
		2	FBC	22.47	
		2	BF	23.03	
		3	BC	65.17	
		4	do nothing	95.75	
	OSP2	1	BF	28.12	
		2	CF	36.46	
		3	FBC	43.71	
		4	do nothing	65.58	
		5	BC	83.33	
effort	flight	1	FBC	5.27	
		1	CF	5.27	
		1	BF	5.34	
		2	BC	7.89	
		3	do nothing	91.94	
	ground	1	CF	3.78	
		2	FBC	5.2	
		2	BF	5.59	
		3	BC	7.36	
		4	do nothing	91.06	
	OSP	1	CF	12.2	
		2	BC	20.59	
		3	BF	27.21	
		3	FBC	30.28	
		4	do nothing	97	
	OSP2	1	BC	6.1	
		2	FBC	29.14	
		3	CF	35.15	
		3	BF	36.7	
		4	do nothing	74.13	
					50%

Figure 13: Results for applying the best policies for the different target functions of FBC on NASA project templates. Note that the mean values are normalized.



estimated	Case study	Rank (Mann Whitney 95%)	what	Normalized median estimate	2nd quartile, median, 3rd quartile
defects	Small	1	do nothing	0.05	
		2	CF	3.18	
		3	BF	33.22	
		3	FBC	35.61	
		4	BC	96.03	
	Medium	1	do nothing	0.2	
		2	CF	7.73	
		3	FBC	40.15	
		3	BF	46.18	
		4	BC	81.13	
	Large	1	do nothing	0.03	
		2	CF	14.75	
		3	FBC	47.89	
		3	BF	48.01	
		4	BC	96.54	
months	Small	1	do nothing	0.72	
		2	BC	19.31	
		3	CF	68.41	
		4	BF	82.13	
		4	FBC	83.94	
	Medium	1	do nothing	0.2	
		2	BC	14.24	
		3	FBC	66.47	
		4	BF	76.77	
		4	CF	83.79	
	Large	1	do nothing	0.74	
		2	BC	18.75	
		3	FBC	70.06	
		4	BF	71.61	
		4	CF	84.74	
effort	Small	1	do nothing	0.51	
		2	BC	60.04	
		3	CF	66.27	
		4	FBC	66.37	
		4	BF	78.04	
	Medium	1	do nothing	0.14	
		2	BC	30.89	
		2	FBC	32.34	
		3	BF	46.46	
		4	CF	64.13	
	Large	1	do nothing	0.23	
		2	FBC	26.51	
		2	BF	28.97	
		2	BC	37.97	
		3	CF	58.97	
					50%

Figure 14: Results for applying the worst policies for the different target functions of FBC. Note that the mean values are normalized.

estimated	Case study	Rank (Mann Whitney 95%)	what	Normalized median estimate	2nd quartile, median, 3rd quartile
defects	flight	1	do nothing	0.03	
		2	CF	12.87	
		3	BF	70.13	
		4	FBC	73.81	
		5	BC	90.87	
	ground	1	do nothing	0.14	
		2	CF	14.37	
		3	BF	48.39	
		4	FBC	63.22	
		5	BC	88.48	
	OSP	1	do nothing	0.75	
		2	CF	4.77	
		3	BF	78.16	
		4	FBC	87.71	
		5	BC	96.52	
	OSP2	1	CF	2.3	
		1	do nothing	2.48	
		2	BC	84.92	
		2	BF	85	
		2	FBC	85.39	
months	flight	1	do nothing	1.49	
		2	BC	61.92	
		3	FBC	77.98	
		4	BF	80.46	
		4	CF	85.43	
	ground	1	do nothing	1.13	
		2	BC	45.4	
		3	FBC	67.35	
		3	BF	69.61	
		4	CF	86.49	
	OSP	1	do nothing	2.62	
		2	BC	9.61	
		3	FBC	53.28	
		4	BF	67.25	
		5	CF	75.98	
	OSP2	1	BC	8.41	
		2	do nothing	39.25	
		3	BF	57.94	
		3	FBC	58.88	
		4	CF	91.59	
effort	flight	1	do nothing	0.5	
		2	BC	38.75	
		3	FBC	54.81	
		4	BF	59.81	
		4	CF	64.41	
	ground	1	do nothing	0.32	
		2	BC	22.83	
		2	FBC	46.86	
		3	BF	48.53	
		4	CF	73.68	
	OSP	1	do nothing	1.22	
		2	FBC	56.59	
		2	BC	58.23	
		2	CF	65.18	
		3	BF	78.63	
	OSP2	1	do nothing	10.98	
		1	CF	15	
		2	FBC	53.83	
		3	BF	54.52	
		4	BC	63.19	
					50%

Figure 15: Results for applying the worst policies for the different target functions of FBC on NASA project templates. Note that the mean values are normalized.