

A second look at Faster, Better, Cheaper

Oussama El-Rawas · Tim Menzies

Received: 5 August 2010 / Accepted: 1 October 2010
© Springer-Verlag London Limited 2010

Abstract “Faster, Better, Cheaper” (*FBC*) was a systems development methodology used by NASA in the 1990s. While usually a deprecated practice, we find that, with certain caveats, it is a viable approach. To determine this we utilized a stochastic AI tool to determine the behavior of *FBC* for several case studies. In these case studies we compare results of using *FBC* with that of other optimization policies. In our tests, *FBC* is as advantageous a policy to use with projects as other policies, while avoiding their apparent downfalls.

Keywords Software engineering · Predictor models · COCOMO · Faster Better Cheaper · Simulated annealing · Software processes

1 Introduction

“Faster, Better, Cheaper” (*FBC*) was a systems development philosophy adopted by the NASA administration in the mid- to late-1990s that led to 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 other project failures at NASA.

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 one-fourth of that cost (e.g. Mars Pathfinder). However, subsequent failures (Mars Climate Orbiter and Polar Lander; the Columbia Shuttle disaster) led to much criticism of *FBC*. This failure had led to a suggestion by many experts that only two aspects of *FBC* can be optimized for at the same time. The phrase “Faster, Better, Cheaper: pick any two” arose from this suggestion, presenting the idea that *FBC* is not a viable optimization scheme.

We ask the question of whether *FBC* can simultaneously achieve all three of its objectives. Previous papers by El-Rawas [11] and Menzies et al. [23] 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. For that study we used a stochastic AI tool we named STAR. In this paper we seek to pose three separate questions regarding the use of our tool (STAR) and *FBC*:

- Is STAR stable enough to rely on?
- How appropriate is STAR for conducting these studies?
- Do previous results presented in [11, 23] hold? Does using additional case studies produce similar results, further validating the use of *FBC*?

Specifically we wish to test three assertions:

- [H1] The stochastic nature of STAR does not render it unusable and unreliable for the purposes of decision making.
- [H2] There are instances where the possible best case value for an attribute is not linear on the attribute settings. This suggests the need to explore the whole range of possibilities using tools like STAR.

O. El-Rawas · T. Menzies (✉)
LCSEE Department, West Virginia University,
Morgantown, WV, USA
e-mail: tim@menzies.us

O. El-Rawas
e-mail: orawas@gmail.com

[H3] Not considering all aspects of a software project can be dangerous. This supports *FBC* as a viable policy to follow.

We will start out by presenting *FBC*, followed by a brief background section. This will be followed by a description of STAR, our stochastic tool, followed by presenting the case studies and methods used. Finally, we present the results obtained regarding the questions posed above.

2 “Faster, Better, Cheaper”

In the 1990s, the main approach to implementing *FBC* within NASA was to downsize 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” [39], 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 [10].

Some within NASA, like 30-year veteran Frank Hoban, who viewed these new policies as a necessary break from traditional policies that were very risk averse, supported these policies [10]. 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, was a success. 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% [18], further straining that program. Furthermore, a single prime contractor (Lockheed Martin) was used for missions in another bid to reduce cost and managerial complexity [41, 42].

This move 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 [18].

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 [40]. 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 have strengthened *FBC* within NASA and JPL, and been seen to break new ground in terms of mission completion with the reduced staff and budget [13].

Both of these missions failed. Using a single contractor had weakened quality assurance and caused loss of vehicle. These flaws were 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 [31]). The Mars Program Independent Assessment Team Report [42] found that these missions were under-staffed, under-funded by at least 30%, and too tightly scheduled.

Elsewhere, across the Atlantic in the UK, another Mars mission to deliver a lander, designated the Beagle 2, was under way. This mission 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 [30].

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” [9, 15]. These criticisms turned NASA away from *FBC* towards policies concentrating on two of the three aspects of *FBC*.

Having shown the history of the rise of infamy of *FBC*, we will proceed to briefly present some background knowledge relating to the studies conducted. This will include the software engineering models and the tool that we use in our study (STAR). We will also briefly present the case studies being used.

3 Background

3.1 Software models

The software models used in this paper are USC COCOMO based. The COCOMO models represent *FBC* in the following manner:

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

$$score = \frac{\sqrt{f \cdot M^2 + b \cdot D^2 + c \cdot E^2}}{\sqrt{f + b + c}} \tag{1}$$

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 Eq. 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*. Hence, STAR can be used as a tool for assessing policies produced under *FBC* because of its ability to use the *score* function to combine score from different aspects of a software project.

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. $b = f = c = 1$.

Regarding the COCOMO II model, Boehm et al. [6] advocated 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 :

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

The particular effort multipliers and scale factors recommended by Boehm et al. are shown in Table 1. While Boehm et al. offer default values for the Eq. 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 are 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 reported very large tuning variance in the post-tuning values of α and β [29]. Baker [3] offers a similar finding. After thirty 90% random samples of that data, the A, B ranges found during tuning were surprisingly wide:

$$(2.2 \leq A \leq 9.18) \wedge (0.88 \leq B \leq 1.09) \tag{3}$$

We are not the only research group to be concerned about tuning variance. At PROMISE 2007, Korte and Port [20] 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 [7,28], 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 [11,24,26] that we describe below checks for stable conclusions within the space of possible tunings.

3.2 STAR

STAR uses Table 2 as the inputs to a Monte Carlo simulation over a set of software models. STAR contains not only the COCOMO effort *E* estimator [6] but also the COCOMO development months *M* estimator [6, pp. 29–57], and COQUALMO *D* defects estimator [6, pp. 254–268]. These estimators generate the $\{E, M, D\}$ variables used by Eq. 1 in the introduction.

We use COCOMO & COQUALMO since the space of possible tunings within these models is well defined. Hence, it is possible to explore this tuning space. Recall from Eq. 2

¹ We will use uppercase *B* to denote the COCOMO linear tuning variable of Eq. 2 and lower *b* to denote the business utility associated with defect predictions of Eq. 1.

Table 1 Features of the COCOMO and COQUALMO models used in this study

	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 ($\frac{\text{frequency of major changes}}{\text{frequency of minor changes}}$)	$\frac{12 \text{ months}}{1 \text{ month}}$	$\frac{6 \text{ months}}{2 \text{ weeks}}$	$\frac{2 \text{ weeks}}{2 \text{ days}}$
rely	Required reliability	Errors are slight inconveniences	Errors are easily rectifiable	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

Table 2 The four NASA case studies

Project	Ranges			Values		Project	Ranges			Values	
	Feature	Low	High	Feature	Setting		Feature	Low	High	Feature	Setting
OSP: Orbital space plane	prec	1	2	data	3	OSP2	prec	3	5	flex	3
	flex	2	5	pvol	2		pmat	4	5	resl	4
	resl	1	3	rely	5		docu	3	4	team	3
	team	2	3	pcap	3		ltex	2	5	time	3
	pmat	1	4	plex	3		sced	2	4	stor	3
	stor	3	5	site	3		KSLOC	75	125	data	4
	ruse	2	4							pvol	3
	docu	2	4							ruse	4
	acap	2	3							acap	4
	pcon	2	3							pcap	3
	apex	2	3							pcon	3
	ltex	2	4							apex	4
	tool	2	3							plex	4
	sced	1	3							tool	5
	cplx	5	6							cplx	4
KSLOC	75	125						site	6		
JPL flight software	rely	3	5	tool	2	JPL ground software	rely	1	4	tool	2
	data	2	3	sced	3		data	2	3	sced	3
	cplx	3	6				cplx	1	4		
	time	3	4				time	3	4		
	stor	3	4				stor	3	4		
	acap	3	5				acap	3	5		
	apex	2	5				apex	2	5		
	pcap	3	5				pcap	3	5		
	plex	1	4				plex	1	4		
	ltex	1	4				ltex	1	4		
	pmat	2	3				pmat	2	3		
	KSLOC	7	418				KSLOC	11	392		

Numeric values {1, 2, 3, 4, 5, 6} map to {very low, low, nominal, high, very high, extra high}

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 Eq. 3. We use the variable T to store the range of possible values for these tuning variables (see previous section for the space of tunings).

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 $\min = \max = x$.

- 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/\max is set to the full range of that variable in Table 1.

Second, STAR’s simulated annealer² seeks constraints on the project options P that, normally, most reduce the score of Eq. 1 (for examples of P , see Table 2). For finding the

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 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 [19].

worst policies, STAR had to be slightly adapted to look for the policies that most increased Eq. 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 [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 $rel_y = vh$ appears ten times in the *best* solutions, but only five times in the *rest*, then

$$\begin{aligned}
 E &= (reply = vh) \\
 Prob(best) &= 1,000/10,000 = 0.1 \\
 Prob(rest) &= 9,000/10,000 = 0.9 \\
 freq(E|best) &= 10/1,000 = 0.01 \\
 freq(E|rest) &= 5/9,000 = 0.00056 \\
 like(best|E) &= freq(E|best) \cdot Prob(best) = 0.001 \\
 like(rest|E) &= freq(E|rest) \cdot Prob(rest) = 0.000504 \\
 Prob(best|E) &= \frac{like(best|E)}{like(best|E)+like(rest|E)} = 0.66
 \end{aligned} \tag{4}$$

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 Eq. 4 with a support term. In Eq. 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

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

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 are 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 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 are then discretized into $D = 10$ bins. Discretization converts a continuous range into a histogram with n break points $b_1 \dots b_n$ 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 Eq. 5.
5. **PRUNE:** Also called the back select stage. STAR runs K_2 experiments with the models where the top-ranked ranges 1.. X 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:

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

Previously [24] 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) [4]. 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 .

3.3 Case studies and methods

We use two categories of projects in this paper to study *FBC*. The one category includes three projects of various sizes. Designated *Small*, *Medium*, and *Large*, they corresponded to projects of size ranges $7 \leq \text{KLOC} \leq 13$, $70 \leq \text{KLOC} \leq 130$, and $700 \leq \text{KLOC} \leq 1,300$, 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 means 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.

The second category are the NASA project templates that we have previously used and are presented in Table 2. 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*.

Table 2 describes the details of flight, ground, *OSP*, and *OSP2*. Note that Table 2 does not mention all the features listed in Table 1 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 Table 2, values are picked at random from the full range of Table 1.

One aspect to note from Table 2 is the number of open options *not* specified in the description of the projects. Some of the features in Table 2 are known precisely (see all the features with single *values*). But many of the features in Table 2 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.

Having presented the models, the tool (STAR) and the case studies used in this paper, we will proceed to present our results. These results aim to answer the questions that

we posed previously in the introduction regarding STAR and its use, as well as regarding previous results.

4 Results

Our results are presented in this section in two stages. In stage one we apply some sanity checks to ensure that our tool is adequate and not unnecessarily engineered. In stage two (Sect. 4.3), we return to our assessment of *FBC* policies. Note that for these two stages, we use two distinct categories of projects, as mentioned in the preceding section:

- Three open-ended projects of various sizes: *Small*, *Medium*, and *Large*.
- The NASA project templates that we have previously used, and are presented in Table 2. These projects are derived from actual NASA projects that have used COCOMO-based models for effort estimation. These projects include *flight*, *ground*, *OSP*, and *OSP2*.

In diversifying the types of case studies used, we aim at observing results that can span across multiple project types.

4.1 Stability, control, and validity

STAR is in essence a stochastic tool that randomly generates examples to learn from. Given this random nature, we posed the question of whether STAR can produce fairly stable results. In order to determine the stability of STAR in choosing policies, three open-ended projects, of different sizes, were created. We ran the three of those in STAR using *FBC*, 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. All the projects used are briefly described in Sect. 3.

Tables 3 through 6 present the stability results. The results show that STAR seems to be quite stable in choosing both the best policies and the worst policies to apply. 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 (small, medium, and large), the results presented in Tables 3 and 4 for the *Small* project seem to be less stable, generating a larger range of policies compared with *Medium* and *Large*. This makes sense in small open-ended projects, as the projects 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.

Table 3 The most common policies selected as *bad policies* to apply to the generic *open-ended* projects

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		

Note that policies picked less than 50% of the time have been omitted

- 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 Tables 5 and 6, where the more specific a project is the more restricted and stable are its results, for both searching for good and bad policies. The reason for this is that more specified projects tend to have a smaller search space, causing smaller policy sets to be created by STAR.

Having shown the stability of STAR, we will move on to briefly study the attribute ranges that STAR selects. This will help explore whether STAR is too elaborate and exhaustive, given the general assumption that the software models are linear.

4.2 Range analysis

We ask the question of whether an AI tool that covers the full range of attributes is necessary or not. The alternative might be a much simpler method that only explores the extremes. Other works such as the work of [26] seem to suggest that the nature of the models in use allow us to simply overlook the ranges of the attributes while simply concentrating on the maxima and minima of the attributes.

Figure 1 presents the stable policies for the open-ended projects. The shaded cells indicate 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. Figures 2 and 3 represent the stable policies for the NASA projects. These stable results have been made to overlap 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 are not 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 have used a tool called SEESAW [26,27] to explore the maxima and minima of attributes, avoiding intermediate attribute values. Looking at Fig. 1, we can see that there is support for this operating assumption made by SEESAW in that most of the attributes (mainly those attributes that are proportional to COCOMO II Effort) seem to propagate to either extreme. Note, however, that there is also counter evidence to that presented in these tables. The obvious evidence is the fact that there seems to be policies that are not on the extreme, especially for the medium and small open-ended projects. Not only that, but such evidence occurs also in the specific projects in Figs. 2 and 3:

Table 4 The most common policies selected as *good* policies to apply to the generic open-ended projects

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

Note that policies picked less than 50% of the time have been omitted

- In Fig. 2, notice that there are policies that are ignored when analyzing these projects for the worst case scenarios, such as *cplx* and *ruse*.
- In Fig. 3 there are two instances where a policy is set at a median value. In *OSP*, *time* is set to *high* (in the table referred to as “4”), even though its range extends from *nominal* to *extremely high*. Also, for *OSP2*, *sced* at one point is set to *nominal*, a setting that is not an extreme for that attribute.

This seems to suggest that there is a need to explore the full range of the attributes, using tools like STAR, in order to obtain the best policies. Another piece of evidence is the fact that there are several attributes that are not set in the medium and large projects. This indicates that sometimes the whole range of an attribute can be ignored, including 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. In addition, this suggests that, even though the COCOMO II models are supposed to be linear, there might be non-linear behavior in them which causes this behavior.

4.3 The viability of *FBC*

The previous two sections were “sanity checks” that we were using the right tool for this analysis. Having passed these checks, we can now return to the main point of this paper: the analysis of the viability and utility of *FBC*. In this section we will compare the performance of *FBC* with other “pick any two” policies mentioned in Sect. 2. The observations we will make are based on the quartile charts and Mann–Whitney ranking [21] presented in Figs. 4, 5 and 6. In our

Table 5 The most common policies selected as *bad policies* to apply for NASA projects

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						

Note that policies picked less than 50% of the time have been omitted

results, a row gets a lower rank if its median value is higher than the other rows (i.e. lower is better), in addition to being statistically significantly different.

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. The first and fourth quartiles are not included in the charts in order to negate the effect of outliers. Two results that we will present here are that

- *FBC* does well most of the time. Even though it does not get the absolute best results always, it is able to compete with the other policies in.
- *FBC* does not fail catastrophically by “cutting corners”. This will be made more clear in our analysis.

Note that some of the results presented here reiterate results that were mentioned in O. El-Rawas’ thesis [11], and that the study method used in that document for studying *FBC* is also used in this section.

For Figs. 4, 5 and 6, the first presents the quartile charts and rankings for *FBC* and its derivate policies for the open-ended projects, and the latter two for the NASA projects. “Derivate policies”, in this context, means those that are geared to optimizing for two of the software project assessment factors [e.g. Cheaper, Faster (*FC*)] rather than the triple goals of *FBC* (Faster, Better, Cheaper).

When observing these figures, we can see that *FBC* does well in general, especially compared with “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 Figs. 4 through 6 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 all of the “pick any two” policies. This result in addition suggests that *FBC* is a viable policy to follow that does not have a large 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 Fig. 4, as well as OSP and OSP2 in Fig. 5, *BC* fails badly with regard to months (development time), even though it ranks first and has the lowest normalized median estimates for defects in all the cases but one. In fact, *BC* does worse than “do nothing” with OSP2 months, which is to say that development time would have been better off in that case had we done nothing.
- For OSP and OSP2 in Fig. 5, *CF* (Cheaper, Faster) fails badly with defects introduced, despite the fact that it ranks at the top with months and effort in three of the four projects.

Table 6 The most common policies selected as *good policies* to apply for NASA projects

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		

Note that policies picked less than 50% of the time have been omitted

One interesting occurrence that we notice is that BF does not 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 does not necessarily reduce months. One example of that are the results for *BC* for OSP2. These show *BC* ranking first for effort in Fig. 6, while simultaneously ranking even worse than “do nothing” for months (development time). Hence, we can state that BF in this case is acting as a pseudo-*FBC* due to this inter-model dependence.

With the above results and observations, we are able to assert that *FBC* is indeed a viable policy to follow. They also show the possible disastrous outcome of neglecting any one aspect of a project, at least for development time (months) and quality (defects). In fact, we can speculate that the reason *FBC* failed historically at NASA was because it was not implemented fully, as demonstrated by the deterioration in quality that caused the failure of major missions.

5 Threats to validity

The results and methods presented in this paper have several threats to their validity which would invalidate them. For instance, all the analyses in this paper are done under the assumption that COCOMO and its related models (COQUALMO) are models that are able to provide us with close estimates. We also assume that we truly know the tuning limits of these models. These assumptions threaten the construct validity, where undermining these models would invalidate this study.

However, 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 [5]. 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 [33], SLIM [35],

Fig. 1 Policy ranges open ended: This figure shows the ranges of policies for the open-ended size projects. The *gray filled cells* represent the allowed range, and *filled triangle* and *open triangle* the policies chosen while searching for good and bad policies, respectively

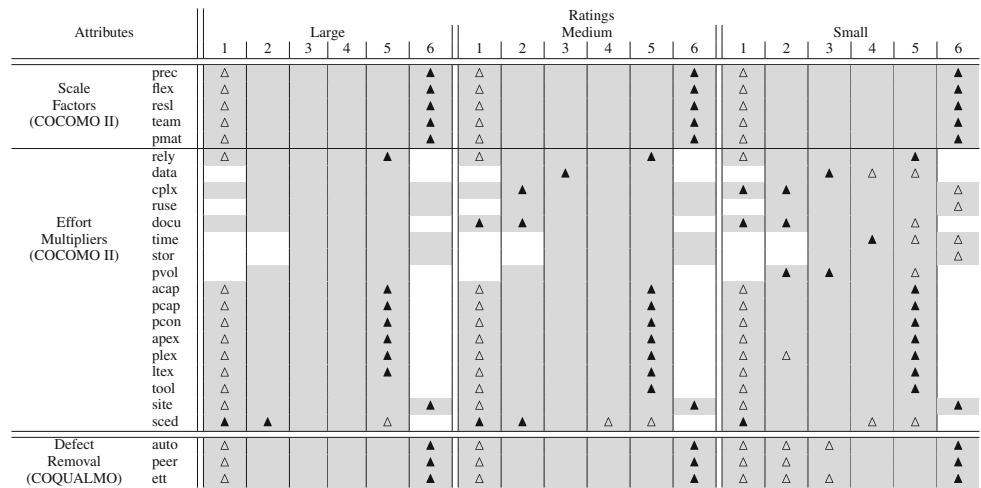


Fig. 2 Policy ranges part 1: This figure shows the ranges for flight and ground. The *dark gray filled cells* represent the allowed range, and *filled square* and *open square* the policies chosen while searching for good and bad policies, respectively. This figure is overlapped over the results for medium projects from Fig. 3

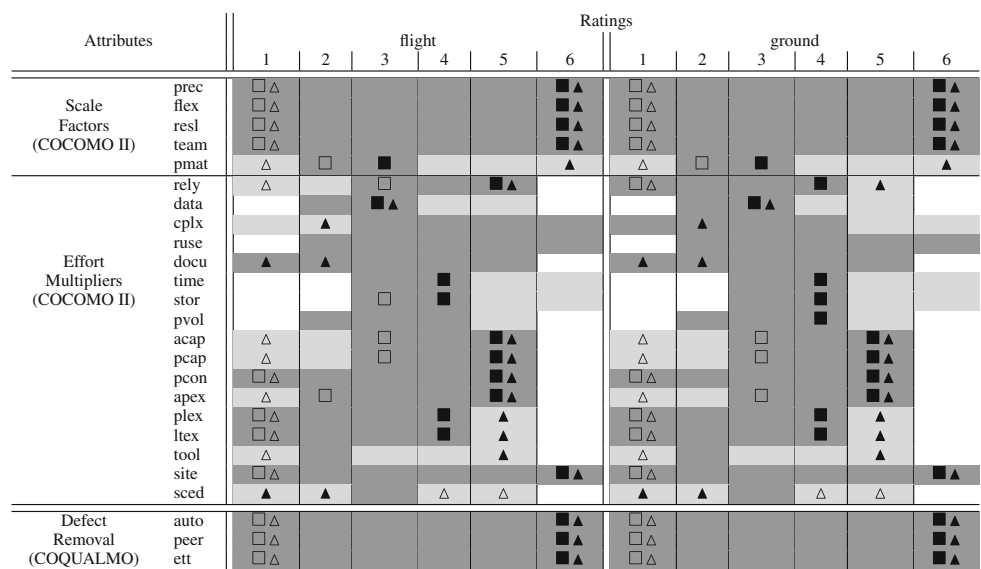


Fig. 3 Policy ranges part 2: This figure shows the ranges for OSP and OSP2. The *dark gray filled cells* represent the allowed range, and *filled square* and *open square* the policies chosen while searching for good and bad policies, respectively. This figure is overlapped over the results for medium projects from Fig. 3

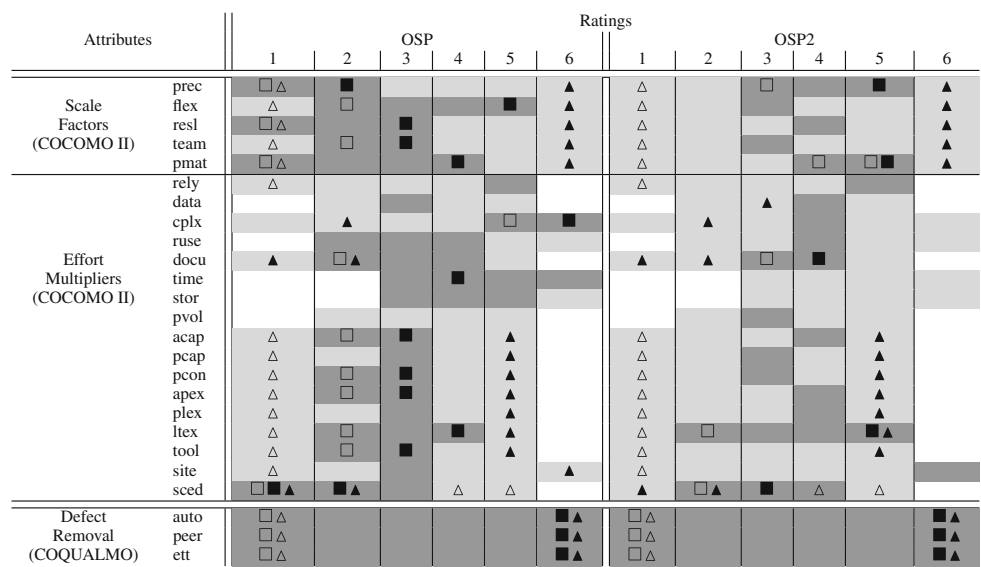
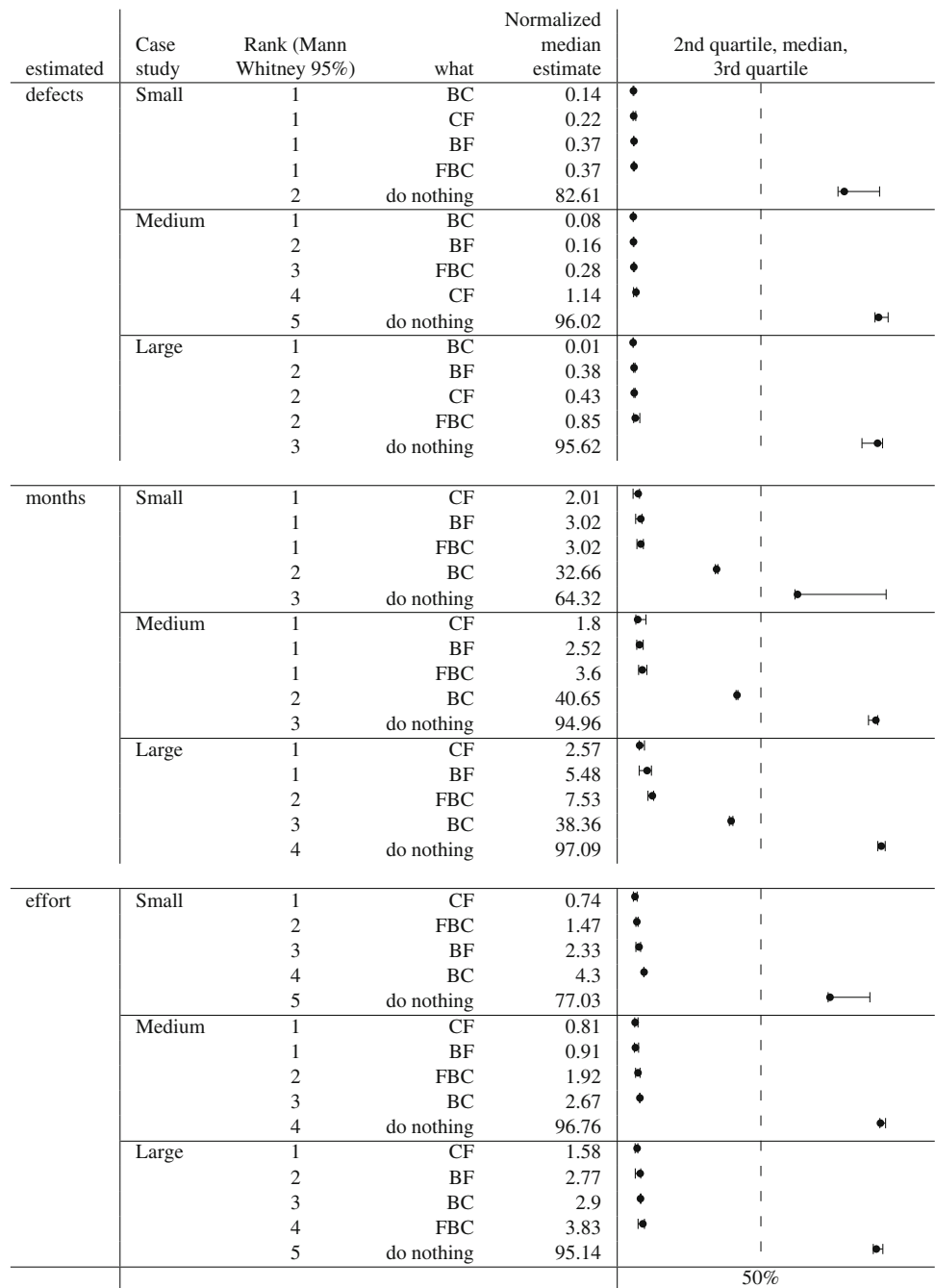


Fig. 4 Results for applying the best policies for the different target functions of FBC. Note that the mean values are normalized



or SEER-SEM [17], the COCOMO family of models are fully described in the literature. Also, at least for the effort model, there exist baseline results [8]. Further, we work extensively with government agencies writing software. Amongst those agencies, these models are frequently used to generate and justify budgets.

The internal validity is threatened by basing this study on a limited dataset of real-world projects sourced from a limited set of development environments, mainly NASA. We included other open-ended size-based project templates as well. However, it could be argued that seven distinct project

templates are not enough to conduct our study. It is important to recall here that these project templates are in fact used to generate super sets of thousands of related projects. These super sets are in turn used by our search-based tool (STAR) for the study. This is similar to the method used by Shepperd and Kadoda [38] to generate projects, the only distinction being that, while we use seven templates, only one was used in that study.

In addition to the above arguments, we have some evidence that the above potential threats to validity do not compromise our conclusions. In fact, we argue that these

Fig. 5 Results for applying the best policies for the different target functions of FBC on NASA project templates. Note that the mean values are normalized (part1)

estimated defects	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	◆-	

conclusions are supported by historical data, where among the 146 missions launched during the *FBC*-Goldin era at NASA, 136 missions actually succeeded [1] (that is a success rate of about 93%).

We argue that the publicized catastrophic failures of *FBC* were due to not using that policy as intended, but rather using it as *CF* (Cheaper, Faster). This falls in line with our results: using *CF* obviously compromised quality in well-defined and constrained projects (in this case *OSP* and *OSP2*).

6 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 based reasoning* (CBR) approaches. One way to do this could be using Figs. 2 and 3 and deriving a distance measure between a given project and one that is 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 set undetermined settings in a project to a value that is within project parameters
2. run a simulation of the project
3. repeat 1 and 2 several thousand times in order to account for random variance, and obtain the median values.

Fig. 6 Results for applying the best policies for the different target functions of FBC on NASA project templates. Note that the mean values are normalized (part2)

estimated effort	Case study	Rank (Mann Whitney 95%)	what	Normalized median estimate	2nd quartile, median, 3rd quartile
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	■ ■	

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.

7 Related work

Much of the related work on uncertainty in software engineering uses Bayesian analysis. For example, Pendharkar et al. [34] demonstrate the utility of Bayes networks in effort estimation while Fenton and Neil explore Bayes nets and defect prediction [12] (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 and Menzies [36]) 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 2 years tuning one such model to one particular site [37].

Other related work is the search-based SE approach advocated by Harmon and Wegener [14]. 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 [16]) but most often in the field of software testing [2]. Harmon’s writing inspired us to try simulated annealing to search the what-ifs in untuned COCOMO models [25]. However, we found that SEESAW ran much faster and produced results with far less variance than simulated annealing. In addition to the above-related work, a recent ICSP’09 paper by Orrego et al. [32] 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 does not always benefit a project, recommending a case-by-case evaluation of the appropriateness of software reuse.

8 Conclusion

This paper provides a resolution to the apparent perception that *FBC* (Faster, Better, Cheaper) is not a viable software systems development policy. In fact, we argue that *FBC* is indeed a viable policy: this is supported by historical data, where among the 146 missions launched during the *FBC*-Goldin era at NASA, 136 missions actually succeeded [1] (that is a success rate of about 93%), in addition to being demonstrated in our study. We argue that the publicized catastrophic failures of *FBC* were due to not using that policy as intended, but rather using it as *CF* (Cheaper, Faster).

We started out this study by stating three statements that we wanted to establish through this study:

- [H1] STAR, despite its stochastic nature, is stable enough. More generally, the stochastic nature of a tool does not render it unusable.
- [H2] There are instances where the possible best case value for an attribute in the COCOMO models is not on the extremes of its range. This suggests the need to explore the whole range.
- [H3] Not considering all aspects of a software project can be dangerous. This supports *FBC* as a viable policy to follow.

Having discussed the concept of “Faster, Better, Cheaper” and its history, we presented a stochastic AI tool which we used to run several simulations in the absence of proper model tuning. Following this, we presented our studies. Concerning [H1], we presented a study that showed that STAR is able to choose certain solutions very consistently, making STAR a stable tool to use. We moved on to [H2] to show that the effort in exploring the entire attribute range is worth it. We found that there are instances where intermediate values are chosen, which support [H2].

Having established the testbed being used, we moved on to [H3]. By presenting a study done on several projects, and by comparing *FBC* to other policies, we were able to uncover downfalls to non-fully inclusive policies. In fact, these downfalls were serious enough at times that doing nothing to the project would have been better. In doing so, we showed that not only is *FBC* viable, but that it would be reckless not to take into consideration all aspects of a software project when it is being developed.

Our final recommendation is that *FBC* should be considered a viable policy, despite its current lack of credibility, if it is implemented by regarding all the factors in *FBC* (Faster, Better, Cheaper) as equally important.

References

- NASA reexamines faster, better, cheaper strategy. http://findarticles.com/p/articles/mi_qa3622/is_200007/ai_n8887228. Accessed June 2000
- Andrews JH, Li FCH, Menzies T (2007) Nighthawk: a two-level genetic-random unit test data generator. In: IEEE ASE'07. <http://menzies.us/pdf/07ase-nighthawk.pdf>
- Baker D (2007) A hybrid approach to expert and model-based effort estimation. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University. <https://eidr.wvu.edu/etd/documentdata.eTD?documentid=5443>
- Boehm B (1981) Software engineering economics. Prentice Hall, Englewood Cliffs
- Boehm B (2000) Safe and simple software cost analysis. In: IEEE software, pp 14–17. http://www.computer.org/certification/beta/Boehm_Safe.pdf. Accessed September/October 2000
- Boehm B, Horowitz E, Madachy R, Reifer D, Clark BK, Steece B, Brown AW, Chulani S, Abts C (2000) Software cost estimation with Cocomo II. Prentice Hall, Englewood Cliffs
- Chen Z, Menzies T, Port D (2005) Feature subset selection can improve software cost estimation. In: PROMISE'05. <http://menzies.us/pdf/05fsscocomo.pdf>
- Chulani S, Boehm B, Steece B (1999) Bayesian analysis of empirical software engineering cost models. IEEE Trans Softw Eng 25(4):573–583
- Cowig K (2003) NASA responds to the columbia accident report: farewell to faster–better–cheaper. <http://www.spaceref.com/news/viewnews.html?id=864>. Accessed September 2003
- David L (2000) NASA report: too many failures with faster, better, cheaper. http://www.space.com/business/technology/business/spear_report_000313.html. Accessed March 2000
- El-Rawas O (2008) Software process control without calibration. Master's thesis, West Virginia University. <http://unbox.org/wisp/var/ous/thesis/thesis.pdf>
- Fenton NE, Neil M (1999) A critique of software defect prediction models. IEEE Trans Softw Eng 25(5):675–689. <http://citeseer.nj.nec.com/fenton99critique.html>
- Hardin M (1999) Mars climate orbiter nearing Sept. 23 arrival, September 1999. JPL Universe, vol 29, no. 19. <http://www.jpl.nasa.gov/info/universe/un990917.pdf>
- Harman M, Wegener J (2004) Getting results from search-based approaches to software engineering. In: ICSE '04: proceedings of the 26th international conference on software engineering, Washington, DC, USA. IEEE Computer Society, pp 728–729
- IFPTE (2003) IFPTE report on the effectiveness of NASA's workforce & contractor policies. <http://www.spaceref.com/news/viewsr.html?pid=10275>. Accessed March 2003
- Jalali O, Menzies T, Feather M (2008) Optimizing requirements decisions with keys. In: Proceedings of the PROMISE 2008 Workshop (ICSE). <http://menzies.us/pdf/08keys.pdf>
- Jensen R (1983) An improved macrolevel software development resource estimation model. In: 5th ISPA conference, pp 88–92, April 1983
- Key S (2003) Columbia, the legacy of “better, faster, cheaper”? http://www.space-travel.com/reports/Columbia_The_Legacy_Of_Better_Faster_Cheaper.html. Accessed July 2003
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
- Korte M, Port D (2008) 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, pp 63–70
- Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60. <http://projecteuclid.org/DPUbs?service=UI&version=1.0&verb=Display&handle=euclid.aoms/1177730491>
- Menzies T, Owen D, Richardson J (2007) The strangest thing about software. IEEE Computer. <http://menzies.us/pdf/07strange.pdf>
- Menzies T, El-Rawas O, Hihn J, Boehm B (2009) Can we build software faster and better and cheaper? In: PROMISE'09. <http://menzies.us/pdf/09bfc.pdf>
- Menzies T, Elrawas O, Barry B, Madachy R, Hihn J, Baker D, Lum K (2008) Accurate estimates without calibration. In: International conference on software process. <http://menzies.us/pdf/08icsp.pdf>
- Menzies T, Elrawas O, Hihn J, Feather M, Boehm B, Madachy R (2007) The business case for automated software engineering. In: ASE '07: proceedings of the twenty-second IEEE/ACM international conference on automated software engineering, New York, NY, USA. ACM, pp 303–312. <http://menzies.us/pdf/07casease-v0.pdf>
- Menzies T, Williams S, El-Rawas O, Boehm B, Hihn J (2009) How to avoid drastic software process change (using stochastic stability). In: ICSE'09. <http://menzies.us/pdf/08drastic.pdf>

27. Menzies T, Williams S, El-Rawas O, Baker D, Boehm B, Hihn J, Lum K, Madachy R (2009) Accurate estimates without local data? *Softw Process Improv Pract* 14:213–225. <http://menzies.us/pdf/09nodata.pdf>. July 2009
28. Menzies T, Chen Z, Hihn J, Lum K (2006) Selecting best practices for effort estimation. *IEEE Trans Softw Eng.* <http://menzies.us/pdf/06coseekmo.pdf>. November 2006
29. Menzies T, Orrego A (2005) Incremental discretization and Bayes classifiers handles concept drift and scaled very well. <http://menzies.us/pdf/05sawtooth.pdf>
30. NASA. Beagle 2 mission profile. http://solarsystem.nasa.gov/missions/profile.cfm?MCode=Beagle_02
31. NASA (1999) Mars climate orbiter mishap investigation board phase I report. November 1999
32. Orrego A, Menzies T, El-Rawas O (2009) On the relative merits of software reuse. In: International conference on software process. <http://menzies.us/pdf/09reuse.pdf>
33. Park R (1988) The central equations of the price software cost model. In: 4th COCOMO Users Group Meeting, November 1988
34. Pendharkar PC, Subramanian GH, Rodger JA (2005) A probabilistic model for predicting software development effort. *IEEE Trans Softw Eng* 31(7):615–624
35. Putnam L, Myers W (1992) Measures for excellence. Yourdon Press Computing Series
36. Raffo D, Menzies T (2005) 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)
37. Raffo DM (1996) Modeling software processes quantitatively and assessing the impact of potential process changes of process performance. PhD thesis, Manufacturing and Operations Systems, May 1996
38. Shepperd M, Kadoda GF (2001) Comparing software prediction techniques using simulation. *IEEE Trans Softw Eng* 27(11):1014–1022
39. Spear T (2000) NASA FBC task final report. <http://mars.jpl.nasa.gov/msp98/misc/fbctask.pdf>. March 2000
40. Tuite D (2007) Better, faster, cheaper-pick any two: that old mantra used to be a touchstone for development, but does it still ring true? <http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=14997>. March 2007
41. NASA watch (2003) Faster–better–cheaper under fire. <http://www.nasawatch.com/fbc.html>
42. Young T, Arnold J, Brackey T, Carr M, Dwoyer D, Fogleman R, Jacobson R, Kottler H, Lyman P, Maguire J (2000) Mars program independent assessment team report. NASA STI/Recon technical report N, pp 32462+, March 2000