

# Case-Based Reasoning for Reducing Software Development Effort

Adam Brady,  
Tim Menzies  
Lane Department of CS&EE  
West Virginia University, USA  
adam.m.brady@gmail.com,  
tim@menzies.us

Jacky Keung  
NICTA  
University of NSW  
Sydney  
Australia  
Jacky.Keung@nicta.com.au

Oussama El-Rawas,  
Ekrem Kocaguneli  
Lane Department of CS&EE  
West Virginia University, USA  
orawas@gmail.com,  
kocaguneli@gmail.com

## ABSTRACT

How can we best find project changes that most improve project estimates? Prior solutions to this problem required the use of standard software process models that may not be relevant to some new project. Also, those prior solutions suffered from limited verification (the only way to assess the results of those studies was to run the recommendations back through the standard process models).

Combining case-based reasoning and contrast set learning, the  $\mathcal{W}$  system requires no underlying model. Hence, it is widely applicable (since there is no need for data to conform to some software process models). Also,  $\mathcal{W}$ 's results can be verified (using hold-out sets). For example, in the experiments reported here,  $\mathcal{W}$  found changes to projects that greatly reduced estimate median and variance by up to 95% and 83% (respectively).

## 1. INTRODUCTION

Existing research in effort estimations focuses mostly on deriving estimates from past project data using (e.g.) parametric models [6] or case-based reasoning (CBR) [41] or genetic algorithms [22]. That research is curiously silent on how to change a project in order to (say) reduce development effort. That is, that research reports *what is* and not *what should be changed*.

Previously [12,26,28,31], we have tackled this problem using STAR/NOVA, a suite of AI search algorithms that explored the input space of standard software process models to find project options that most reduced the effort estimates. That approach had some drawbacks including (a) the dependency of the data to be in the format of the standard process models; (b) the implementation complexity of the Monte Carlo simulator and the AI search engines; and (c) the lack of an independent verification module (the only way to assess the results of those studies was to run the recommendations back through the standard process models).

This paper describes “ $\mathcal{W}$ ”, a simpler, yet more general, solution to the problem of finding project changes that most improves project

estimates. Combining case-based reasoning and contrast set learning,  $\mathcal{W}$  requires no underlying parametric model. The code is much simpler (hundreds of lines of AWK vs thousands of lines of C & LISP). Also  $\mathcal{W}$  can be applied to more data sets (since there is no requirement for the data sets to be in the format required by the standard software process models). Finally,  $\mathcal{W}$ 's results can be verified (using hold-out sets). For example, in the experiments reported here,  $\mathcal{W}$  found changes to projects that greatly reduced estimate median and variance by up to 95% and 83% (respectively).

The difference between  $\mathcal{W}$  (that finds what to change in order to improve an estimate) and a standard case-based effort estimator (that only generates estimates) is very small. Based on the this experiment, we advise augmenting standard CBR tools with modules like planning sub-systems like  $\mathcal{W}$ .

The rest of this paper is structured as follows. XXXX in Section 2, followed by a brief explanation of CBR in Section 3. We will then present the datasets used in Section 4, followed by a presentation of the new approach, which we call  $\mathcal{W}$  (The Decider) in Section 5. Experimental results and analysis will be presented and discussed in Section 6 followed by future research direction in Section 7. Section 8 discusses applications where the proposed approach may be applicable and Section 9 concludes the paper.

## 2. BACKGROUND

### 2.1 Why Study Effort Estimation?

Generating (and regenerating) project effort estimates is an important and continuous process for project managers. Good estimates allow better utilize resources. Also, by reviewing and improving their estimation process,, a software company can learn and improve from their past experience.

Sadly, we often get estimates wrong. Consider the recent case of NASA's Check-out Launch Control System, which was cancelled when the initial estimate of \$200 million was overrun by an *additional* \$200M [43]. This case is not unique, despite the significant effort put into designing more accurate estimation models, it has been reported that many predictions are wrong by a factor of four or more [5, 19].

In order to conduct software effort estimation, there is a need for models to express and allow the estimation of needed effort and cost. Many software process models have emerged aiming to achieve that task, and there has not emerged a single standardized model that is widely used by the software engineering industry. There several reasons for this including *generality*, *data islands* and *in-*

*stability*. Software models may not be *general* so it can be inappropriate to apply a software model learned in environment one to environment two. Also, many companies prefer to keep cost related data confidential. This *data island* effect has also contributed to the fragmentation of the field by preferring building private models rather than using publicly available models. This multiplicity of software effort models has led to scarcity of the publicly available local data needed for the model based effort estimation. Without sufficient data to build, audit, and tune models, the predictions generated by these models may be highly *unstable*. Baker [4] reports a study that learned values for the  $(a, b)$  (linear, exponential) constants in Boehms' COCOMO software process model [5]. The study was repeated 30 times, each time selecting at random a sample of 90% of the available project data. The learned values for  $(a, b)$  exhibited an alarming variance:

$$(2.2 \leq a \leq 9.18) \wedge (0.88 \leq b \leq 1.09) \quad (1)$$

Such large variations make it possible to misunderstand the effects of changing project options. Suppose some proposed change doubles productivity, but  $a$  moves from 9 to 4.5. The improvement resulting from that change would be obscured by the tuning variance.

## 2.2 Handling Large Variances and Uncertainty

The process simulation community (e.g. Raffo [38]) studies elaborate software process models. While such models offer a 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 [37]. Slow tuning time means that models will exhibit large variances for much of their lifetime.

These large variances defeat standard decision making algorithms. For example, a standard tool in sensitivity analysis [39] is to partially differentiate the equations of a model, then explore the coefficients on the partials. Large variances like Equation 1 guarantee that any such gradient analysis will yield highly variable results.

Miller [33] reports that *feature selection* can reduce the variance of model predictions. While he was talking specifically about least squares regression, his more general point is that the fewer the variables in a model, the less the contribution of variable variance to output variance. Inspired by Miller, we have conducted extensive feature selection experiments with effort data sets [9, 24, 29], with disappointing results. As predicted by Miller, variance is decreased when (say) 20 attributes are reduced to 10. But note in Baker's study that large variances persist for effort data, even after reducing the number of variables to two. Clearly, we need to look further than just feature selection.

The opposite to feature selection is *case selection*. In case selection, data sets of size  $D$  are replaced with a smaller data set of *prototypes*. The smaller the data set, the fewer the values and (hopefully), the smaller the variance. *Prototype generation* should not be confused with nearest neighbor methods: the latter is run on a per-query basis while the former is run once as a pre-processor to all subsequent inferencing. The art of prototype generation is to find or generate, a set of representative examples that best represents and replaces the training cases. Typically, prototype generation removes most of the training data. For example, Chang's prototype generators [7] replaced training sets of size  $T = (514, 150, 66)$  with prototypes of size  $N = (34, 14, 6)$  (respectively). That is, prototypes may be as few as  $\frac{N}{T} = (7, 9, 9)\%$  of the original data. The problem with prototype generation is that

it can be very slow. [22] discusses genetic algorithms that combine feature and case selection. Dr. Li was kind enough to share his code. Using that system, we have learned that searching the  $2^D$  space of possible cases can be very slow. A similar result was reported by Kirsopp and Shepperd [20], who found that case selection was only practical for very small data sets. Hence, at least in this paper, we will eschew case selection.

If we cannot remove variance, we must deal with the uncertainties it presents. Much of the related work on uncertainty in software engineering uses a Bayesian analysis. For example, Pendharkar et.al. [35] demonstrate the utility of Bayes networks in effort estimation while Fenton and Neil explore Bayes nets and defect prediction [13]. We take a different approach here due to the simplicity of our implementation (see below).

Of all the literature, we found the Search-Based Software Engineering (SBSE) research [16] to be closest to this work. 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 [17]) but most often in the field of software testing [3].

## 2.3 STAR and NOVA

The SBSE literature inspired us try simulated annealing to search the what-ifs associated with Equation 1. This led to the STAR system [12, 28]. NOVA was a generalization of STAR that included simulated annealing, and other search engines [26, 31].

STAR/NOVA handled model variance by finding conclusions that were stable across the space of possible predictions. This was accomplished under the assumption that, for a mature effort estimation model, the range of possible tunings was known (this is the case for models like COCOMO). For such models, it is possible for the AI search engines to find conclusions that hold across the space of all tunings. STAR and NOVA were implemented in C++ and LISP respectively.

STAR and NOVA constrain project options  $P$  but not the tuning options  $T$ . Hence, their recommendations contains subsets of the project options  $P$  that most improve the score, despite variations in the tunings  $T$ . This approach meant we could reuse COCOMO models requiring using local tuning data. The following is a description that briefly presents the operation of STAR and NOVA:

1. *SAMPLE*: We sample across the ranges of all the attributes in the model  $n_1$  times. Most of the time we sample randomly across the range, with some heuristics that allow us to concentrate more on the extremes of the range.
2. *DISCRETIZE*: The data seen in the  $n_1$  samples is then discretized into  $D = 10$  bins. Equal frequency bins were used.
3. *CLASSIFY*: The generated top  $n\%$  projects are then classified as *best* or *rest*.
4. *RANK*: The ranges are then ranked in increasing order using Support-Based Bayesian Ranking.
5. *PRUNE*: Also called the back select stage. STAR runs  $n_2$  experiments with the models where the top ranked ranges  $1..X$

study	NOVA
nasa93 flight	72%
nasa93 ground	73%
nasa93 osp	42%
nasa93 ops2	5%

**Figure 1: Improvements in effort estimates** ( $\frac{initial-final}{initial}$ ) **found by NOVA. From [14].**

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. 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.

In practice, STAR/NOVA approach was very effective. Figure 1 shows the large effort reductions found by STAR in three out of four case presented at ASE'09. It is insightful to reflect about STAR/NOVA's failure to find large reductions in the fourth case study (*nasa93 osp2*). In this project, management had already fixed most of the project options. STAR/NOVA failed, in this case, since there was very little left to try and change. This fourth case study lead to one of the lessons learned of STAR/NOVA: apply project option exploration tools as early as possible in the lifecycle of a project. Or, to say that more succinctly: *if you fix everything, there is nothing left to fix* [27].

While a successful reserach prototype, STAR/NOVA has certain drawbacks:

- *Model dependency*: Using stochastic approach, a model to calculate (e.g.) estimated effort is still required. In order to do so, we had to use some software process models to generate te estimates. Hence, the conclusions reached by STAR/NOVA are only as good as this model. That is, if a client doubts the relevance of those models, then the conclusions will also be doubted.
- *Data Dedendency*: STAR/NOVA's AI search algorithms explord some underlying software process model. Hence, it could only process project data in a format compatible with the underlying model. In practice, this limits the scope of the tool.
- *Inflexibility*: It proved to be trickier than we thought to code up the process models, in a manner suitable for Monte Carlo simulation. By our count, STAR/NOVA's models required 22 design decisions to handle certain special cases<sup>1</sup>. Lacking guidance from the literature, we just had to apply "engineering judgement" to make those decisions. While we think we made the right decisions, we cannot rigorously justify them.
- *Performance*: Our stochastic approach conducted several tens of thousands of iterations to explore the search space, with

<sup>1</sup>E.g. one user required the following modification: do not increase the use of automatic tools unless the analyst capability also increases

several effort estimates needed calculated for each iteration. This resulted in a performance disadvantage.

- *Size and Maintainability*: Due to all the above factors, our code base proved difficult to maintain.

While there was nothing in principle about applying our techniques to other software effort models, we believe that the limiting factor on disseminating our technique is the complexity of our implementation. As partial evidence for this, we note that in the three years since we first reported our technquie [28]:

- We have only able to code in one set of software process models (COCOMO), which inherently limited the scope of our study.
- No other research group has applied these techniques.

Therefore, rather than further elaborate a complex code base, we explored a different option, based on Case Based Reasoning (CBR). The proposed new approach had no model restrictions (since it does not use a model) and can accomodate a wide range of data sets (since there are no restrictions of the variables that can be processed). The new option, called  $\mathcal{W}$ , was surprisingly easy to write (a few hundred lines of AWK, rather than the 3000 lines of C++ in STAR or the 5000 lines of LISP in NOVA).

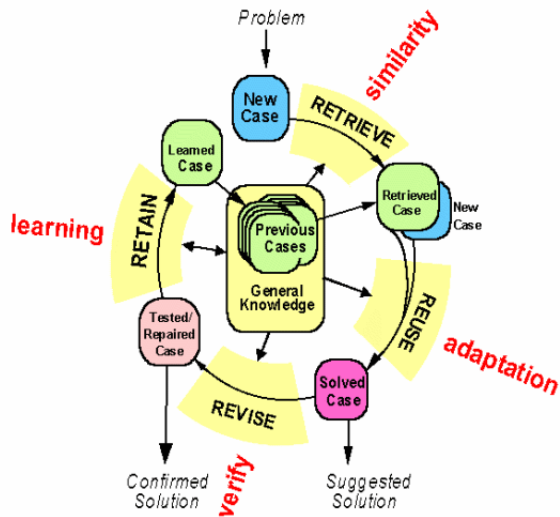
### 3. CASE-BASED REASONING (CBR)

Case based reasoning is a method of machine learning that seeks to emulate human recollection and adaptation of past experiences in order to find solutions to current problems. That is, as humans we tend to base our decisions not on complex reductive analysis, but on an instantaneous survey of past experiences [40]; i.e. we don't think, we remember. CBR is purely based on this direct adaptation of previous cases based on the similarity of those cases with the current situation. Having said that, a CBR based system has no dedicated world model logic, rather that model is expressed through the available past cases in the case cache. This cache is continuously updated and appended with additional cases.

Aamodt & Plaza [1] described a 4-step general CBR cycle, which consisting of:

1. *Retrieve*: Find the most similar cases to the target problem.
2. *Reuse*: Adapt our actions conducted for the past cases to solve the new problem.
3. *Revise*: Revise the proposed solution for the new problem and verifying it against the case base.
4. *Retain*: Retain the parts of current experience in the case base for future problem solving.

Having verified the results from our chosen adapted action on the new case, the new case is added to the available case base. The last step allows CBR to effectively learn from new experiences. In this manner, a CBR system is able to automatically maintain itself. As discussed below,  $\mathcal{W}$  supports *retrieve*, *reuse*, *revise* (and *retain* is a decision left up to the user collecting project data).



**Figure 2: figure**

A diagram describing the steps of CBR (source: <http://www.peerscience.com/Assets/cbrcycle1.gif>).

This 4-stage cyclical CBR process is sometimes referred to as the R4 model [42]. Shepperd [42] considered the new problem as a case that comprises two parts. There is a description part and a solution part forming the basic data structure of the system. The description part is normally a vector of features that describe the case state at the point at which the problem is posed. The solution part describes the solution for the specific problem (the problem description part).

The similarity between the target case and each case in the case base is determined by a similarity measure. Different methods of measuring similarity have been proposed for different measurement contexts. A similarity measure is measuring the closeness or the distance between two objects in an n-dimensional Euclidean space, the result is usually presented in a distance matrix (similarity matrix) identifying the similarity among all cases in the dataset. Although there are other different distance metrics available for different purposes, the Euclidean distance metric is probably the most commonly used in CBR for its distance measures.

Irrespective of the similarity measure used, the objective is to rank similar cases from case-base to the target case and utilize the known solution of the nearest  $k$ -cases. The value of  $k$  in this case has been the subject of debate [18, 41]: Shepperd [41], Mendes [23] argue for  $k = 3$  while Li [22] propose  $k = 5$ .

Once the actual value of the target case is available it can be reviewed and retained in the case-base for future reference. Stored cases must be maintained over time to prevent information irrelevancy and inconsistency. This is a typical case of incremental learning in an organization utilizing the techniques of CBR.

Observe that these 4 general CBR application steps (retrieve, reuse, revise, retain) do not include any explicit model based calculations; rather we are relying on our past experience, expressed through the case base, to estimate any model calculations based on the similarity to the cases being used. This has two advantages:

1. It allows us to operate independently of the models being used. For example, our prior report to this conference [14] ran over two data sets. This this study, based on CBR, uses twice as many data sets.
2. This improves our performance, since data retrieval can be more efficient than calculation, especially given that many thousands of iterations of calculation were needed with our traditional modeling based tool. As evidence of this, despite the use of a slower language,  $\mathcal{W}$ 's AWK code runs faster than the C++/LISP used in STAR/NOVA. It takes just minutes to conduct 20 trials over 13 data sets with  $\mathcal{W}$ . A similar trial, conducted with NOVA or STAR, can take hours to run.

#### 4. FROM CBR TO $\mathcal{W}$

A standard CBR algorithm reports the median class value of some local neighbourhood. The  $\mathcal{W}$  algorithm treats the local neighborhood in a slightly different manner:

- The local neighborhood is divided into *best* and *rest*;
- A *contrast set* is learned that most separates the regions (contrast sets contain attribute ranges that are common in one region, but rare in the other).
- $\mathcal{W}$  then searches for a subset of the contrast set that best selects for (e.g.) the region with lower effort estimates.

The rest of this section details the above process.

##### 4.1 Finding Contrast Sets

Once a contrast set learner is available, it is a simple matter to add  $\mathcal{W}$  to CBR.  $\mathcal{W}$  finds contrast sets using a greedy search, where candidate contrast sets are ranked by the log of the odds ratios. Let some attribute range  $R$  appear at frequency  $N_1$  and  $N_2$  in two regions of size  $R_1$  and  $R_2$ . Let the  $R_1$  region be the preferred goal and  $R_2$  be some undersired goal. The log of the odds ratio, or LOR, is:

$$LOR = \log \left( \frac{N_1/R_1}{N_2/R_2} \right)$$

Note that when  $LOR = 0$ , then a range is occurring with the same probability in each region (such ranges are therefore not useful for selecting on region or another). On the other hand, when  $LOR > 0$ , the range is more common in the preferred region than otherwise. These positive ranges are candidate members of the contrast set that selects for the desired outcome.

It turns out that, for many data sets, the LOR values for all the ranges contain a small number of very large values (strong contrasts) and a large number of very small values (weak contrasts). The reasons for this distribution do not concern us here (and if the reader is interested in this *master-variable* effect, they are referred to [25, 30]). What is relevant is that LOR can be used to rank candidate members of a contrast set.  $\mathcal{W}$  computes the LORs for all ranges, then conducts experiments applying the top  $i$ -th ranked ranges.

For more on LOR, and their use for multi-dimensional data, see [34].

##### 4.2 The Algorithm

CBR systems input a query  $q$  and a set of cases. They return the subset of cases  $C$  that is relevant to the query. In the case of  $\mathcal{W}$ :

```

@project
@attribute ?rely 3 4 5
@attribute tool 2
@attribute cplx 4 5 6
@attribute ?time 4 5 6

```

**Figure 3:  $\mathcal{W}$ 's syntax for describing the input query  $q$ . Here, all the values run 1 to 6.  $4 \leq cplx \leq 6$  denotes projects with above average complexity. Question marks denote what can be controlled- in this case, *rely*, *time* (required reliability and development time)**

- Each case  $C_i$  is an historical record of one software projects, plus the development effort required for that project. Within the case, the project is described by a set of attributes which we assume have been discretized into a small number of discrete values (e.g. analyst capability  $\in \{1, 2, 3, 4, 5\}$  denoting very low, low, nominal, high, very high respectively).
- Each query  $q$  is a set of constraints describing the particulars of a project. For example, if we were interested in a schedule over-run for a complex, high reliability projects that have only minimal access to tools, then those constraints can be expressed in the syntax of Figure 3.

$\mathcal{W}$  seeks  $q'$  (a change to the original query) that finds another set of cases  $C'$  such that the median effort values in  $C'$  are less than that of  $C$  (the cases found by  $q$ ).  $\mathcal{W}$  finds  $q'$  by first dividing the data into two-thirds training and one-third testing. *Retrieve* and *reuse* are applied to the training set, then *revising* is applied to the test set.

1. *Retrieve*: The initial query  $q$  is used to find the  $N$  training cases nearest to  $q$  (using a Euclidean distance measure where all the attribute values are normalize 0..1).
2. *Reuse* (adapt): The  $N$  cases are sorted by effort and divided into the  $K_1$  best cases (with lowest efforts) and  $K_2$  rest cases. For this study, we used  $K_1 = 5, K_2 = 15$ . Then we seek the contrast sets that select for the  $K_1$  best cases with lowest estimates. All the attribute ranges that the user has marked as "controllable" are scored and sorted by LOR. This sorted order  $S$  defines a set of candiate  $q'$  queries that use the first  $i - th$  entries in  $S$ :

$$q'_i = q \cup S_1 \cup S_2 \dots \cup S_i$$

Formally, the goal of  $\mathcal{W}$  is find the smallest  $i$  value such  $q'_i$  selects cases with the least median estimates.

According to Figure 2, after *retrieving* and *reusing* comes *revising* (this is the "verify" step). When revising  $q'$ ,  $\mathcal{W}$  prunes away irrelevant ranges as follows:

1. Set  $i = 0$  and  $q'_i = q$
2. Let  $Found_i$  be the test cases consistent with  $q'_i$  (i.e. that do not contradict any of the attribute ranges in  $q'_i$ ).
3. Let  $Effort_i$  be the median efforts seen in  $Found_i$ .

4. If  $Found$  is too small then terminate (due to over-fitting). After Shepperd [41], we terminated for  $|Found| < 3$ .
5. If  $i > 1$  and  $Effort_i < Effort_{i-1}$ , then terminate (due to no improvement).
6. Print  $q'_i$  and  $Effort_i$ .
7. Set  $i = i + 1$  and  $q'_i = q_{i-1} \cup S_i$
8. Go to step 2.

On termination,  $\mathcal{W}$  recommends changing a project according to the set  $q' - q$ . For example, in Figure 3, if  $q' - q$  is *rely* = 3 then this treatment recommends that the best way to reduce the effort this project is to rejecting *rely* = 4 or 5.

One useful feature of the above loop is that it is not a black box that offers a single "all-or-nothing" solution. Rather it generates enough information for a user to make their own cost-benefit trade offs. In practice, users may not accept all the treatments found by this loop. Rather, for pragmatic reasons, they may only adopt the first few  $S_i$  changes seen in the first few rounds of this loop. Users might adopt this strategy if (e.g.) they have limited management control of a project (in which case, they may decide to apply just the most influential  $S_i$  decisions).

Implementing  $\mathcal{W}$  is simpler than the STAR/NOVA approach:

- Both NOVA and STAR contain a set process models for predicting effort, defects, and project threats as well as Monte Carlo routines to randomly select values from known ranges. STAR and NOVA implement simulated annealing while NOVA also implements other search algorithms such as ASTAR, LDS, MAXWALKSAT, beam search, etc. STAR/NOVA are 3000 and 5000 lines of C++ and LISP, respectively.
- $\mathcal{W}$ , on the other hand, is 300 lines of AWK script

Our pre-experimental suspicion was that  $\mathcal{W}$  was too simple and would need extensive enhancement. However, the results shown below suggest that, at least for this task, simplicity can suffice (but see the future work section for planned extensions).

Note that  $\mathcal{W}$  verification results are more rigorous than those of STAR/NOVA.  $\mathcal{W}$  reports results on data that is extenal to its deliberation process (i.e. on the test set). STAR/NOVA, on the other hand, only reported the changes to model output once certain new constraints were added to the model input space.

## 5. DATA

Recall that a CBR system takes input a query  $q$  and cases  $C$ .  $\mathcal{W}$  has been tested using multiple queries on the data sets of Figure 4. These queries and data sets are described below.

### 5.1 Data Sets

As shown in Figure 4, our data includes:

- The standard public domain COCOMO data set (Cocomo81);
- Data from NASA;

Dataset	Attributes	Number of cases	Content	Historical Effort Data					
				Units	Min	Median	Mean	Max	Skewness
coc81	17	63	NASA projects	months	6	98	683	11400	4.4
nasa93	17	93	NASA projects	months	8	252	624	8211	4.2
desharnais	12	81	Canadian software projects	hours	546	3647	5046	23940	2.0
maxwell	26	62	Finish banking software	months	6	5189	8223	63694	3.3
isbsg	14	29	Banking projects of ISBSG	minutes	662	2355	5357	36046	2.6
Total: 328									

Figure 4: The 328 projects used in this study come from 5 data sets.

```
@project
@attribute ?TeamExp -1 0 1
@attribute ?ManagerExp 3 4 5 7
@attribute PointsAjust 302 397 697
```

Figure 5: The *desharnais-team-m* query selects for medium sized projects being developed by teams with low experience.

- Data from the International Software Benchmarking Standards Group (ISBSG);
- The Desharnais and Maxwell data sets;

Except for ISBSG, all the data used in this study is available at <http://promisedata.org/data> or from the authors.

Note the skew of this data (min to median much smaller than median to max). Such asymmetric distributions complicate model-based methods that use Gaussian approximations to variables.

There is also much divergence in the attributes used in our data:

- While our data include effort values (measured in terms of months or hours), no other feature is shared by all data sets.
- The Cocomo and NASA data sets all use the features defined by Boehm [5]; e.g. analyst capability, required software reliability, memory constraints, and use of software tools.
- The other data sets use a wide variety of features including, number of entities in the data model, number of basic logical transactions, query count and number of distinct business units serviced.

This attribute space divergence is a significant problem for model-based methods like STAR/NOVA since those systems can only accept data that conforms to the space of attributes supported by their model. For example, this study uses the five data sets listed in Figure 3. STAR/NOVA can only process two of them (coc81 and nasa93).

CBR tools like  $\mathcal{W}$ , on the other hand, avoid these two problems:

- $\mathcal{W}$  makes no assumption about the underlying distributions of the variables.
- $\mathcal{W}$  can be easily applied to any attribute space (caveat: as long but there are some dependent variables).

## 5.2 Queries

Figure 3, Figure 5 and Figure 6 show samples of the  $\mathcal{W}$  query language:

- The idiom “@attribute name range” defines the range of interest for some attribute “name”.
- If “range” contains multiple values, then this represents a disjunction of possibilities.
- If “range” contains one value, then this represents a fixed decision that cannot be changed.
- The idiom “?x” denotes a controllable attribute (and  $\mathcal{W}$  only generates contrast sets from these controllables). Note that the “range”s defined for “?x” must contain more than one value, otherwise there is no point to making this controllable.

This study used 13 different queries:

- For the ISBSG data set, we used our recent experience to describe the constraints suitable for a standalone or client server system (denoted *stdalone* and *clientsrv*).
- For the Desharnis data set, we posed queries representing:
  - (*s, m, l*) denotes (small, medium, large) projects;
  - (*mnggr, team*) denotes (manager, team) experience being low.

For example, Figure 5 shows the *desharnais-team-m* query.

- For the Cocomo and NASA data sets, we used our contacts at the Jet Propulsion Laboratory to write queries describing typical *flight* and *ground* systems, and for two versions of the guidance systems for the Orbital Space plane (called *osp* and *osp2*).
- Lacking direct experience with the Finnish financial system, we could not pose specific queries to the Maxwell dataset. Instead, we made half the attributes controllable and used that for the Maxwell query (see Figure 6).

## 6. EXAMPLE

Figure 7 shows an example of  $\mathcal{W}$  running on the (dataset, query) of (coc81, osp):

- The **training results** are shown on lines 5 to 8. These are the the  $S_i$  scores on all the ranges with a positive LOR. Note that one range has an outstandingly high LOR; *stor* = 3 has an LOR of 0.41 while all the rest are under 0.1. That is, we should expect most of the benefit from applying just one range (*stor* = 3).

```

@project
@attribute ?T01 1 2 3 4 5
@attribute ?T02 1 2 3 4 5
@attribute ?T03 2 3 4 5
@attribute ?T04 2 3 4 5
@attribute ?T05 1 2 3 4 5
@attribute ?T06 1 2 3 4
@attribute ?T07 1 2 3 4 5
@attribute ?T08 2 3 4 5
@attribute ?T09 2 3 4 5
@attribute ?T10 2 3 4 5
@attribute ?T11 2 3 4 5
@attribute ?T12 2 3 4 5
@attribute ?T13 1 2 3 4 5
@attribute ?T14 1 2 3 4 5
@attribute ?T15 1 2 3 4 5

```

Figure 6: The *maxwell* query.

- The **baseline effort estimates** from  $q_0$  are shown on lines 10,11. These are all the efforts of all the cases selected by the osp query. These efforts are summarized on lines 13,14 (the median effort is 313.5).
- The **testing results** are shown after line 18. In this example,  $q_1$  i was  $q_0$  (osp) plus the top-ranked attribute range ( $stor = 3$ ). The effects of  $q_1$  are shown on lines 19. Now, only five cases are selected with a median effort of 55. Testing stops at line 19 since, as expeted, applying the next ranges did not improve the median or spread results.

Using this example, we define two performance measures that will be used extensively:

- The *improvement* found by  $\mathcal{W}$  is  $100 * \frac{initial-final}{initial}$ . For example, the *median improvement* seen in this example was  $\frac{313-55}{313} = 82\%$ .
- The *spread* is the 75th – 25th percentile interval and is a non-parametric measure of the variance in the estimates. The spreads seen in this example were, for  $q_0$ ,  $653 - 52 = 601$  and, for  $q_1$ ,  $240 - 50 = 190$ . Hence, the *spread improvement* was  $\frac{601-190}{601} = 68\%$ .

Reducing *both* the median and spread are useful since the former relates to development cost and the latter relates to how much we can trust and estimate (the smaller the spread, the more we can trust it).

## 7. RESULTS

Figure 8 shows the improvements seen in our 13 queries, running on the data sets of Figure 4. As shwon by the last line of Figure 8, the usual improvements where (36,68) for (median,spread). Note that, unlike STAR/NOVA, these are results on real-world data sets not used during training.

Figure 9 displays the Figure 8 results graphically. The dashed line indicates the median of the improvements for each axis. As can be seen, in the majority of cases,  $\mathcal{W}$  found treatments that improved both the median and spread of the effort estimates. Sometimes, the improvements are quite dramatic.

- In the case of (coc81.ground), a 95% effort improvement was seen when  $q'$  found that ground systems divide into two

```

1 historical data : coc81
2 new project(s) : osp
3 #n score range
4 -----
5 1 0.41 stor = 3
6 2 0.09 acap = 5
7 3 0.00 cplx = 4
8
9 q_0:
10 9 15 40 41 50 55 61 79 83 240 387
11 423 523 539 605 702 1063 1600 2040 6400
12
13 25% 50% 75%
14 Baseline: 52.5 313.5 653.5
15
16 Results of applying the top n-th ranges
17 found during training
18
19 q_1: stor=3 : {15 50 55 240 702} (median,spread)= 55,190

```

Figure 7: Running (coc81,osp).

dataset	query $q$	Improvement	
		median	spread
coc81	flight	0%	18%
coc81	ground	95%	83%
coc81	osp	82%	68%
coc81	osp2	31%	71%
desharnais	mngr-s	23%	85%
desharnais	mngr-m	23%	85%
desharnais	mngr-l	36%	-45%
desharnais	team-s	-11%	-206%
desharnais	team-m	0%	60%
desharnais	team-l	-15%	-93%
ISBSG	stdalone	69%	100%
ISBSG	clientsrv	42%	88%
maxwell		44%	76%
nasa93	flight	61%	73%
nasa93	ground	77%	15%
nasa93	osp	49%	48%
nasa93	osp2	27%	42%
	median	36%	68%

Figure 8: Improvements ( $100 * \frac{initial-final}{intial}$ ) for 13 queries. Gray cells show negative improvements.

groups (the very expensive, and the very simple). In this case  $\mathcal{W}$  found the factor that drove a ground system into the very simple case.

- In the case of (ISBSG,stalone), a 100% improvement in spread was seen when  $q'$  selected a group of projects that were co-developed and, hence, all had the same development time.

Other improvements were more modest: in the case of (coc81,flight), the median did not improve (but did not get worse) while the spread was only reduced by 18%.

One data set had consistently worse results than any other. The gray cells of Figure 8 indicate when  $\mathcal{W}$  failed (i.e. whereand the treatments increased median development effort or spread, or both). Note that the gray cells are found only in the Desharnais results.

On investigation, the root cause of the probelm was the granularity of the data. Whereas (e.g.) coc81 assigns only one of six values

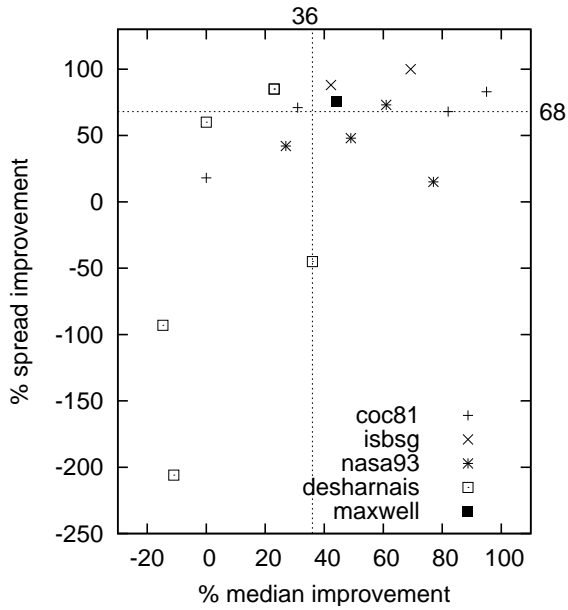


Figure 9: Median and spread improvements for effort estimation. Dashed lines mark the median values of both axes.

to each attribute, Desharnais’s attributes had a very wide range. Currently, we are exploring discretization policies to [11,44] reduce attributes with large cardinality to a smaller set. Tentatively, we can say that discretization solves the problem with Desharnais but we are still studying this aspect of our system.

## 8. DISCUSSION

### 8.1 Threats to Validity

*External validity* is the ability to generalize results outside the specifications of that study [32]. To ensure the generalizability of our results, we studied a large number of projects. Our datasets contain a wide diversity of projects in terms of their sources, their domains and the time period they were developed in. Our reading of the literature is that this study uses more project data, from more sources, than numerous other papers. All the papers we have read, as well as, Table 4 of [21] list the total number of projects in all data sets used by other studies. The median value of that sample is 186, which is less much less than the the 328 projects used in our study.

*Internal validity* questions to what extent the cause-effect relationship between dependent and independent variables hold [2]. The general internal validity issue is that data mining experiments (like those discussed above) do not collect new data, but only generates theories from historical data. Ideally, we should take a learned theory and apply it to some new situation, then observe if the predicted effect occurs in practice. Note that if no explicit theory is generated, then it cannot be applied outside of the learning system. That is, all CBR systems suffer from issues of internal validity since, unlike decision tree learners [36] or rule-based learners [10] they do not generate an explicit theory. However, it is possible to mitigate this problem by simulating how an CBR system might be applied to a new situation. Note that  $\mathcal{W}$ ’s testing phase generates estimates using test data that is not used in training.

study	NOVA	$\mathcal{W}$
nasa93 flight	72%	61%
nasa93 ground	73%	77%
nasa93 osp	42%	49%
nasa93 ops2	5%	27%

Figure 10: Comparing improvements ( $\frac{initial-final}{initial}$ ) found by NOVA (from Figure 1) and  $\mathcal{W}$  (Figure 8).

### 8.2 Future Work

While a promising prototype, there are many design options left to explore within  $\mathcal{W}$ . For example:

- $\mathcal{W}$  a generalization of the Euclidean distance measure used in  $\mathcal{W}$ ’s training phase is

$$Distance = \sqrt{\sum_{i=1}^n w_i (x_i - y_i)^2} \quad (2)$$

In the above,  $w_i$  is a weighting applied to each attribute. Currently,  $\mathcal{W}$  uses  $w_i = 1$  but there are any number of proven and fast feature weighting schemes that can focus the inferring away from noisy attributes [8, 15].

- There a few too many *magic numbers* within the current version of  $\mathcal{W}$ . For example, during training, the best and rest sets are of size  $K_1 = 5$  and  $K_2 = 15$ . Now that we have baseline results, it would be prudent to conduct experiments to see if there are better values for  $K_1$  and  $K_2$ .
- $\mathcal{W}$  does a greedy linear-time search to explore the contrast set. Perhaps some of STAR/NOVA’s search algorithms could be applied to  $\mathcal{W}$  to improve that aspect of the inference.

Also, given the power of contrast sets to extract relevant attributes, future work exists in seeing if  $\mathcal{W}$  is a useful feature subset selector for other domains.

## 9. SUMMARY

Our research has now produced two different methods for finding changes that most effect a project. Both methods search for changes that most effect (and improve) the output effort estimates:

- $\mathcal{W}$  is a *case-based* method that searches relevant historical examples.
- STAR/NOVA [14] are *model-based* methods that applies AI search to the input space of software process models.

Figure 10 shows that estimation improvements found by  $\mathcal{W}$  (in this report) to the improvements reported previously (in Figure 1). This table is much shorter than Figure 8 since, due to the modeling restrictions imposed by the software process models, STAR/NOVA cannot be applied to all the data sets that can be processed by  $\mathcal{W}$ . It is encouraging to note that the results are similar. Better yet, the  $\mathcal{W}$  improvements are not always less than those found by STAR/NOVA.

The numbers in Figure 10 cannot be directly compared due to the different experimental methods that generated them (we are currently building a rig that will allow a rigorous statistical comparison



of  $\mathcal{W}$  vs STAR/NOVA). Regardless of the results of that statistical analysis, there are clear indicators of when we would prefer  $\mathcal{W}$  or STAR/NOVA:

- $\mathcal{W}$  is a case-based method. If historical cases are not available, then STAR/NOVA is the preferred method.
- STAR/NOVA is based on the USC COCOMO suite of models. If the local business users do not endorse that mode, then  $\mathcal{W}$  is the preferred method.

## 10. REFERENCES

- [1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7:39–59, 1994.
- [2] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.
- [3] 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>.
- [4] 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>.
- [5] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [6] 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.
- [7] C. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Trans. on Computers*, pages 1179–1185, 1974.
- [8] 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>.
- [9] Z. Chen, T. Menzies, D. Port, and B. Boehm. Finding the right data for software cost modeling. *IEEE Software*, Nov 2005.
- [10] P. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.
- [11] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995. Available from <http://www.cs.pdx.edu/~timm/dm/dougherty95supervised.pdf>.
- [12] O. El-Rawas. Software process control without calibration. Master's thesis, 2008. Available from <http://unbox.org/wisp/var/ous/thesis/thesis.pdf>.
- [13] 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>.
- [14] P. Green, T. Menzies, S. Williams, and O. El-waras. Understanding the value of software engineering technologies. In *IEEE ASE'09*, 2009. Available from <http://menzies.us/pdf/09value.pdf>.
- [15] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge And Data Engineering*, 15(6):1437–1447, 2003. Available from <http://www.cs.waikato.ac.nz/~mhall/HallHolmesTKDE.pdf>.
- [16] 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.
- [17] 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>.
- [18] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. Experiences using casebased reasoning to predict software project effort, 2000.
- [19] C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.
- [20] C. Kirsopp and M. Shepperd. Case and feature subset selection in case-based software project effort prediction. In *Proc. of 22nd SGAI International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, Cambridge, UK*, 2002.
- [21] B. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Trans. Softw. Eng.*, 33(5):316–329, 2007. Member-Kitchenham, Barbara A.
- [22] Y. Li, M. Xie, and T. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82:241–252, 2009.
- [23] E. Mendes, I. D. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.
- [24] 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>.
- [25] T. Menzies, D. Owen, and J. Richardson. The strangest thing about software. *IEEE Computer*, 2007. <http://menzies.us/pdf/07strange.pdf>.
- [26] T. Menzies, O. El-Rawas, J. Hihn, and B. Boehm. Can we build software faster and better and cheaper? In *PROMISE'09*, 2009. Available from <http://menzies.us/pdf/09bfc.pdf>.
- [27] T. Menzies, O. Elrawas, D. Baker, J. Hihn, and K. Lum. On the value of stochastic abduction (if you fix everything, you lose fixes for everything else). In *International Workshop on Living with Uncertainty (an ASE'07 co-located event)*, 2007. Available from <http://menzies.us/pdf/07fix.pdf>.
- [28] T. Menzies, O. Elrawas, J. Hihn, M. Feathear, B. Boehm, and R. Madachy. The business case for automated software engineering. 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>.
- [29] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes. Specialization and extrapolation of induced domain models: Case studies in software effort estimation. In *IEEE ASE, 2005*, 2005. Available from

<http://menzies.us/pdf/05learncost.pdf>.

- [30] T. Menzies and H. Singh. Many maybes mean (mostly) the same thing. In M. Madravio, editor, *Soft Computing in Software Engineering*. Springer-Verlag, 2003. Available from <http://menzies.us/pdf/03maybe.pdf>.
- [31] T. Menzies, S. Williams, O. El-rawas, B. Boehm, and J. Hihn. How to avoid drastic software process change (using stochastic stability). In *ICSE'09*, 2009. Available from <http://menzies.us/pdf/08drastic.pdf>.
- [32] D. Milic and C. Wohlin. Distribution Patterns of Effort Estimations. In *Euromicro*, 2004.
- [33] A. Miller. *Subset Selection in Regression (second edition)*. Chapman & Hall, 2002.
- [34] M. Možina, J. Demšar, M. Kattan, and B. Zupan. Nomograms for visualization of naive bayesian classifier. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 337–348, New York, NY, USA, 2004. Springer-Verlag New York, Inc.
- [35] 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.
- [36] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1992. ISBN: 1558602380.
- [37] 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.
- [38] 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.
- [39] A. Saltelli, K. Chan, and E. Scott. *Sensitivity Analysis*. Wiley, 2000.
- [40] R. C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, New York, NY, USA, 1983.
- [41] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12), November 1997. Available from [http://www.utdallas.edu/~rbaner/SE\\_XII.pdf](http://www.utdallas.edu/~rbaner/SE_XII.pdf).
- [42] M. J. Shepperd. Case-based reasoning and software engineering. Technical Report TR02-08, Bournemouth University, UK, 2002.
- [43] Spareref.com. Nasa to shut down checkout & launch control system, August 26, 2002. <http://www.spaceref.com/news/viewnews.html?id=475>.
- [44] Y. Yang and G. I. Webb. A comparative study of discretization methods for naive-bayes classifiers. In *Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, 2002.