Empir Software Eng (2012) 17:1–17
DOI 10.1007/s10664-011-9193-5

EDITORIAL

Special issue on repeatable results in software engineering prediction

Tim Menzies · Martin Shepperd

Published online: 3 January 2012

1 Motivation

The goal of science is *conclusion stability*, i.e. to discover some effect X that holds in multiple situations. Sadly, there are all too few examples of stable conclusions in software engineering (SE). In fact, the typical result is *conclusion instability* where what is true for project one, does not hold for project two.

We can find numerous studies of the following form: there is as much evidence *for* as *against* the argument that some aspect X adds value to a software project. Below are four examples of this type of problem which we believe to be endemic within SE.

- Jørgensen (2004) reviewed 15 studies comparing model-based to expert-based estimation. Five of those studies found in favor of expert-based methods, five found no difference, and five found in favor of model-based estimation.
- Mair and Shepperd (2005) compared regression to analogy methods for effort estimation and similarly found conflicting evidence. From a total of 20 empirical studies, seven favored regression, four were indifferent and nine favored analogy.
- Kitchenham et al. (2007) reviewed empirical studies that checked, if data imported from other organizations were as useful as local data (for the purposes of building effort models). From a total of seven studies, three found that models from other organizations were not significantly worse than those based on local data, while four found that they were significantly different (and worse).

T. Menzies (✉)
West Virginia University, Morgantown, WV, USA
e-mail: tim@menzies.us

M. Shepperd
Brunel University, Uxbridge, UB8 3PH, UK
e-mail: martin.shepperd@brunel.ac.uk

– Zimmermann et al. (2009) learned defect predictors from 622 pairs of projects $\langle project_1, project_2 \rangle$. In only 4% of pairs did defect predictors learned in $project_1$ worked in $project_2$.

One explanation for conclusion instability, is that given the divergent nature of software projects and software programmers, it is to be expected that different researchers find different effects. Whilst the search for the universal model or predictor is unlikely to be fruitful (Wolpert and Macready 1997), we hope that there is still much scope for generalizing in useful subspaces of the universe of all possible software engineering (SE) phenomena; otherwise, we have little hope of developing general principles. And if that were to be true, then the purpose of empirical SE is undermined.

In this special issue, we explore conclusion instability in prediction systems about software engineering. This editorial is an attempt to synthesize a diverse field. The rest of this special issue contains articles that explore more specific issues in depth.

This editorial is organized as follows:

– Section 2 poses the question of what exactly do we mean by conclusion instability and provides some formal apparatus to provide answers.
– Section 3 explores why such a situation might arise.
– This is followed in Section 4 by a discussion of research directions regarding instability reduction.
– Finally 5 reviews the papers in this issue.

This collection of papers is very much a community effort. We warmly thank both our contributors and reviewers without whom this issue would not have been possible.

2 A Formal Framework

In this section we explore in more detail exactly what we mean by conclusion instability with the intention of providing a framework on which we can locate our subsequent ideas and the more detailed contributions of the other authors of this special issue. A fuller account with examples may be found in Shepperd and MacDonell (2012).

Fundamentally, what are we doing when we compare two or more competing prediction systems? We try to make some inference about the difference in their performance over a population of software engineering entities from a sample contained in a data set D. D will contain status variables $X_1, X_2, \ldots X_k$ and Y as the response variable (e.g. person-hours if we are trying to predict effort). We apply prediction systems $P_{1,2,\ldots}$ and predict $\hat{y}_i$, where i is the ith case (e.g. a project if we're predicting effort) where $1 \leq i \leq n$ in D. This is done for each prediction system. The i's are chosen according to a validation scheme V, for instance a k-folds or leave-one-out cross validation procedure.

For each case y_i that we make a prediction for, we compute the prediction error or residual[1] $y_i - \hat{y}_i$. S is then some statistic (usually related to accuracy) defined over the errors, e.g. the mean residual or the sum of the squares. Many different statistics have been proposed over the years e.g. MMRE and R-squared. We do not propose to get involved in that particular debate since it is explored elsewhere in this special issue (Angelis and Mittas 2012) and elsewhere (Kitchenham et al. 2001). What all these statistics have in common is that each is some kind of summary statistic defined over the residuals. The purpose of an empirical validation is to estimate the differences in the chosen accuracy statistic S *between* prediction systems $P_{1,2,\ldots}$. Many researchers choose to use some inferential test to determine the likelihood of the difference $S(P_1) - S(P_2)$ being due to chance. For example, researchers might employ a paired t-Test to compare means of the residuals from two samples arising from P_1 and P_2.

The important point is that a validation study can be conceived as an estimator function of the population statistic $\hat{S}$. Each dataset that is used is a sample from some underlying population. Of course these are not random samples, although the practical challenges to making it otherwise are immense. The error of the sample is the difference $S - \hat{S}$, the bias is $S - E(\hat{S})$ where E is the expected value and the variance is $E((S - \hat{S})^2)$. Given that researchers are using different estimators of different statistics on different samples, it is not difficult to understand why conclusion instability might arise. Also it must be stressed that we are interested in the bias and variance of the estimators *not* the original predictions made by the prediction system.

Conceptually we might define a conclusion (from an empirical evaluation) as being some empirical preference relation $S(P_1) \prec S(P_2)$ meaning from an experimental observation we (strictly) prefer P_2 to P_1. A weak preference relation is denoted $\preccurlyeq$ which could be interpreted as P_2 is not worse than P_1. Note that a single study might lead to more than one conclusion if, for example, more than two prediction systems are evaluated or more than one data set used. We might start to build a chain of preferences such as $S(P_1) \prec S(P_2) \prec S(P_3)$.

We should also note that another useful empirical preference relation is indifference $\sim$ which is not the same as equality. We might interpret it as meaning the difference in $S(P_1)$ and $S(P_2)$ is not large enough to be statistically significant or the effect size is not deemed to have any practical impact. For instance if an empirical study were to compare using two case-based reasoners to predict software project effort, one configured $k = 50$ and the other $k = 51$ it is quite possible that the difference in performance is so small that the researchers could not reasonably distinguish between the desirability of using either prediction system, hence we would be indifferent.

Ultimately the goal of empirical research is to establish a preference ordering over the set of prediction systems. Unfortunately there are certain difficulties. First, the

[1]Note if we were evaluating classifiers we would base our statistics on the confusion matrix rather than residuals (Baldi et al. 2000).

cardinality of P is quite high so a complete ordering implies many, many relations ($P \times P$). Second, which is the main theme of this special issue, not all the empirical relations are what might informally be thought of as consistent. Therefore conclusion instability might be thought of as the situation where the set of preference ($\prec, \preccurlyeq, \sim$) relations over the set P of prediction systems $\{P_1, P_2, \ldots\}$ are not order-theoretic: that is one or more of the properties of transitivity and antisymmetry of the $\prec$ relation are violated. Since we are dealing with semi-orders, we do not require completeness; which is convenient since empirical evaluation is an ongoing process.

Given this view of empirical evaluation as an attempt to find an ordering over a set of preference relations, it opens up the possibility of graphical notations such as Hasse Diagrams (Davey and Priestley 2002). This is a potentially powerful means of summarizing multiple empirical results, since it abstracts away from details such as the choice of accuracy statistic and data set, and focuses upon the essence; that is, which P we should choose.

So why might we observe empirical relations that are not order-theoretic, in other words suffer from conclusion instability? We can conceive of the difference $|S(P_1) - S(P_2)|$ as being the detected or *observed* effect of changing between the competing prediction systems. In other words what is the effect upon our accuracy statistic S of changing our prediction system from P_1 to P_2? And depending upon the direction of the effect (i.e. an increase or otherwise in prediction accuracy) this will guide our preference for one P over another and this is expressed as an empirical relation. Of course what is observed is some sample of projects contained in D, whereas we wish to make conclusions about a defined (and larger) population, e.g. all software projects. In other words we try to estimate S based on the empirical validation procedure. Other studies also estimate S and these estimates may differ.

In order to avoid problems of differing units resulting from different data sets or accuracy statistics, it is commonplace to use a standardized effect size measure. As we are interested in differences between values for S, $|S(P_1) - S(P_2)|$, the usual approach is to normalize by some estimate of the population standard deviation, possibly pooled where there are clear differences between the samples. This gives rise to a measure such as Cohen's d or Glass's Δ (Glass et al. 1981). Cohen has suggested—as a crude mechanism—that effect sizes can then be allocated to the categories of small, medium or large and this offers another basis for empirical preference relations.

However, the *true* effect size (i.e. the difference in S due to changing the prediction system) of the population can be confounded by other sources of variance and consequently we *might* observe conclusion instability. We cannot directly measure the population statistic S and so have an estimator of S derived from the sample of the population represented by the dataset D. We might expect the estimates to vary by chance though. As the number and size of the samples increase, one might hope for estimate to converge on the true value.

This could arise because of different choices in data set or accuracy statistic or simply different levels of expertise from the research teams or for a multiplicity of other reasons. This is particularly likely if the underlying or true effect size is small. Figure 1 shows a hierarchical breakdown of why the true effect size might be confounded. The next section considers this very important question in more detail.

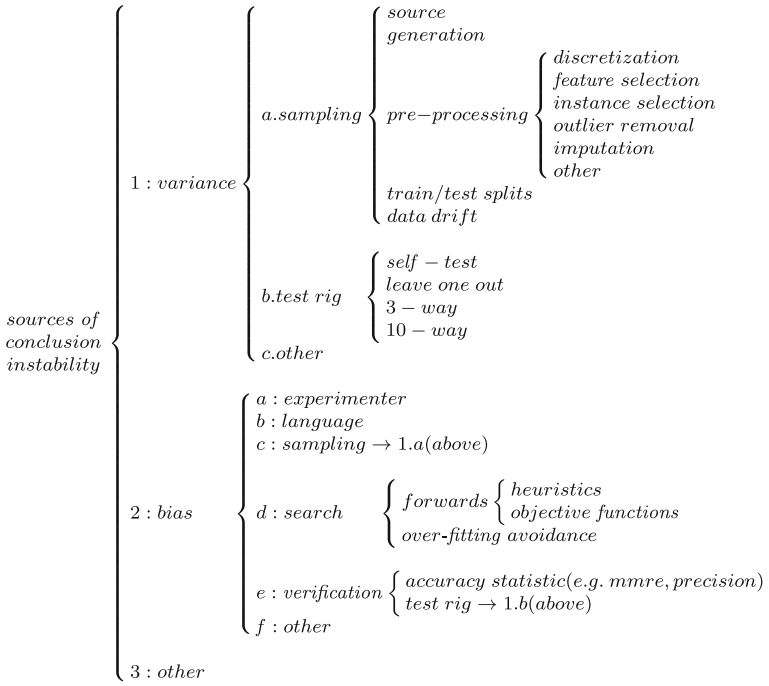


Fig. 1 *Jiggle1*: a partial list of sources of conclusion instability

3 Why Might Conclusion Stability Occur?

This section reviews known sources of conclusion instability. As shown in Fig. 1, there are two major contributors to this instability:

- *Bias* measures the distance from predicted to actual values. If an estimator is *very biased*, then its estimates will be consistently very different in one direction from the true or population (but unknown) value of S. As an example we might have a cross-validation procedure that is very optimistic by not separating the validation cases from the training cases and as a consequence consistently under-estimate prediction errors.
- *Variance* measures the distance between different predictions (it describes the spread or dispersion of the estimators. If we have *high variance*, it is most likely that we will have conclusion instability, particularly if the effect size is small. High variance can be reduced by repeating the validation many times so, for example, an $m \times n$ fold cross validation might be preferred to a simple n fold cross validation.

For continuous predictions (which are our principal concern) if we measure the accuracy of our estimator as mean square error then:

- The squared bias is a measure of the contribution to error of the central tendency or most frequent classification of the learner when trained on different training data.

– The variance is a measure of the contribution to error of deviations from the central tendency.

Figure 1 shows *Jiggle* (version 1.0), a partial list of causes of variance and bias. We suspect this list is incomplete. One of the goals of this special issue is to inspire other researchers to extend this version, perhaps to create *Jiggle2*, *Jiggle3*, etc. Another goal of this special issue is to inspire the research community to address the various parts of *Jiggle*, hopefully developing operators to tame all these *jiggle*s.

The rest of this section offers examples of the leaves of Fig. 1.

3.1 Variance from Sampling

The performance of a predictor can vary markedly from one data set to another. That is, how we *sample* the data prior to modeling can have an enormous effect on the results of the predictions.

3.1.1 Source Sampling

To the best of our knowledge, there is no effort prediction system that works best for all known data sets, i.e. no one prediction system dominates. A more usual report is that the 'best' model for a particular data set depends on that data set. For example Fig. 2 shows the differences in performance of 90 prediction models running on 20 data sets. Two aspects of this figure are worthy of comment:

1. The performance of these predictors can change widely, depending on the data sets. This may not be surprising since each data set represents a (highly non-random) sample from some more general population of software projects. This single observation explains much of the conclusion instability in the literature. Table 4 of Kitchenham et al. (2007) shows a sample of recent estimation studies, and how many data sets were used by each study. Of the ten studies in that sample, only one explored more than one data set. Hence, it is hardly surprising that different researchers favor different prediction models since:
   – they have assessed them on different data sets; and
   – the performance of predictors on different data sets is not always the same.

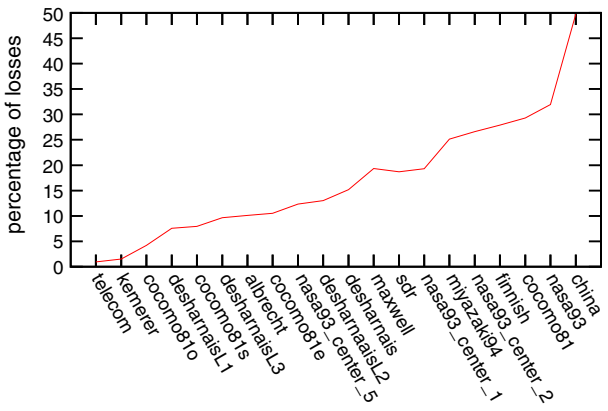


Fig. 2 Testing 90 different prediction systems on 20 data sets from http://promisedata.org/?cat=14. Compares each predictor performs against 89 others (using a Wilcoxon test, 95% confidence). The y-axis sums the number of losses seen for all methods for one data set. The x-axis is sorted by the number of losses per data set

2. Of course not all data sets lead to statistically significant differences in performance in predictors. For example, with the *telecom* data sets, the predictions from 89 of the predictors were so similar that none of them lost to any other. In terms of preference relations we would say we are indifferent.

Sampling bias is discussed further in three of the papers of this special issue (Azzeh 2012; Robles and Gonzalez-Barahona 2012; Turhan 2012).

3.1.2 Pre-Processing and Sampling

Regardless of where a data set comes from, it is often modified as part of the modeling process. Figure 3 illustrates the standard knowledge discovery in databases (KDD) cycle. Prior to applying some procedure to learn a prediction system, data may be extensively transformed. The following list of transforms shows just some of the possible transforms:

- *Discretization:* Numeric data may be discretized into a small number of bins. The effect is to concentrate the signal in a database into a small number of values and is known to significantly improve the performance of predictive systems (Dougherty et al. 1995; Fayyad and Irani 1993). Discretization can greatly affect the results of the learning: since there may be ways to implement discretization. Also, some discretizers change their behavior according to tuning parameters that are controlled by engineers (e.g. the number of bins).
- *Feature selection:* Sometimes, it is useful to prune columns of noisy data or multiple columns that are equally correlated to the target variable (Chen et al. 2005).
- *Instance selection:* Similarly, it is also useful prune rows or cases that are very noisy (contain signals not correlated to the target) or outliers (e.g. rows recording very rare examples) (Kocaguneli et al. 2010).
- Handling missing data items is another source of discrepancy between researchers. There are many different *imputation* methods for missing values (Little and Rubin 2002). Alternatively some researchers choose to remove missing values either through row or column deletion.

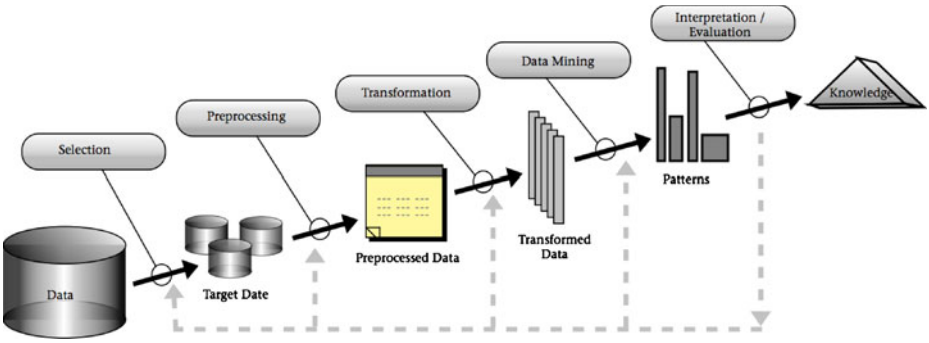


Fig. 3 The KDD cycle. From Fayyad et al. (1996)

– The data may be *transformed*, for example to deal with extreme outliers or non-Gaussian distributions using Tukey's ladder or a similar procedure. For example, Boehm argues that effort is exponential on the size of a project; hence, for linear regression modeling, he proposes using the natural log of all numerics (Boehm 1981).

The effects of pre-processing can be quite dramatic. For example, in the study that resulted in Fig. 2, 90 predictors were generated by combining 10 pre-processors (e.g. descritization into three bins or using natural logs) with 9 data miners (e.g. nearest neighbor). In those 90 predictors, the nearest neighbor algorithm jumped from rank 12 to rank 62, just by switching from three bins to logging.

Since pre-processing can be so important to the predictor, researchers often extensively experiment with different pre-processors. This means that different researchers can start with the *same* data set, yet end up learning predictors from *different data*. Furthermore, since subtle differences can lead to major effects upon the outcome, there can be problems of adequately documenting all the minutiae. The absence of community reporting protocols compounds this source of noise. We believe that pre-processing is potentially a major cause of researchers making different conclusions about what the 'best' predictors are.

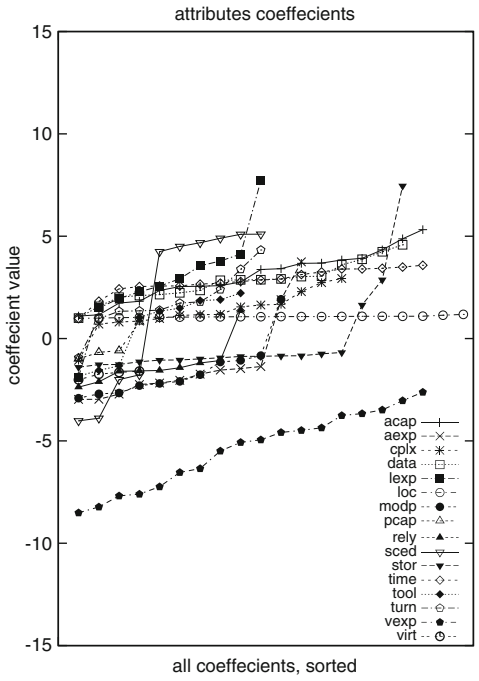
3.1.3 Train/Test Sampling

Another potential source of noise is the validation scheme. A repeated result in data mining (Witten and Frank 2005) is that if a predictor is trained and tested on the *same* data, then the resulting performance statistics *over-estimate* the future performance of the predictor. Hence, it is accepted practice to use some *hold-out* set for testing. For example, in a n-way cross-validation procedure, the data is divided into n equal size bins. Each bin then becomes one test set and the remaining $n - 1$ bins are used for training. Hall and Holmes (2003) go one step further and recommend repeating the n-way cross-validation m times, each time randomizing the order of the data. Such randomizations reduce *order effects* where the effects of learning are influenced by some trivial ordering of the data and perhaps also excessive variance.

An added requirement for cross-validation can be *stratification*. Stratification attempts to maintain the same class distributions in each bin as in the entire data set. Stratification is a heuristic procedure since it may encounter issues that have no singular solution (e.g. how to space rare outliers amongst the bins).

As with pre-processing, train/test set sampling implies that different researchers can start with the *same* data set, yet end up learning predictors from different instances and therefore arrive at different conclusions. These can arise from decisions that are rarely shared between researchers such as (a) the stratification heuristics or (b) the seed for the random number generator used in the $m \times n$-way cross-validation.

But is this a problem? Does it really matter if we build effort estimators from slightly different data? Our results suggest that, this is a significant problem since learning an effort estimation model is usually a highly unconstrained task. For example, calibrating models such as COCOMO involves the 14 attributes of Fig. 4 (right-hand-side) as well as the exponents for the three different development modes. When there is not enough data to constrain parametric model construction, minor



upper: in theory $\beta < 0$	acap: analysts capability
	pcap: programmers capability
	aexp: application experience
	modp: modern programming practices
	tool: use of software tools
	vexp: virtual machine experience
	lexp: language experience
middle	sced: schedule constraint
lower: in theory $\beta > 0$	data: data base size
	turn: turnaround time
	virt: machine volatility
	stor: main memory constraint
	time: time constraint for cpu
	rely: required software reliability
	cplx: process complexity

Fig. 4 COCOMO 1 effort multipliers, and the sorted coefficients found by linear regression from twenty 66% sub-samples (selected at random) from the NASA93 PROMISE data set; from Menzies et al. (2005). Prior to learning, training data was *linearized* in the manner recommended by Boehm (x was changed to $\log(x)$; for details, see Menzies et al. (2005)). After learning, the coefficients were unlinearized

changes to the training data (e.g. due to changes in the train/test set sampling) can lead to large conclusion instabilities in the internal parameters of such models.

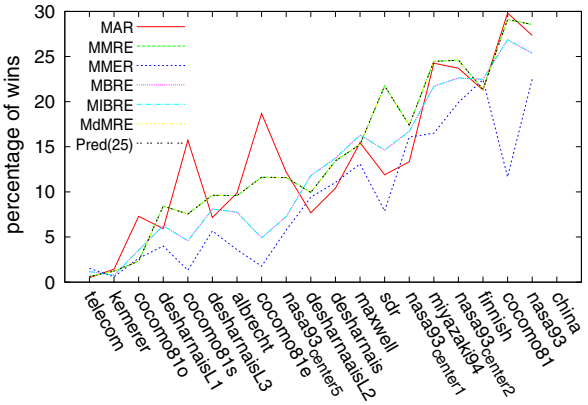
For example, Fig. 4 shows the results of tuning the effort model $effort = a \cdot LOC^b \cdot \prod_i \beta_i \cdot x_i$ on twenty 0.67% samples of the NASA93 data set from the PROMISE repository. In this study, β_i are the coefficients learned for each COCOMO effort multiplier. The thing to note in Fig. 4 is that changes to the training data lead to very large scale changes to the β_i values. In fact, in five cases, the β_i values changed sign from positive to negative.

In summary, seemingly minor decisions in the training and test set used for building a model can lead to very large conclusion instabilities in the learned model.

3.1.4 Experimenter Bias

There are many papers in the literature that propose some new effort estimator. Strange to say, all these different papers report that some new prediction method $P_i n$ is a better effort predictor than techniques published in previous work.

How can there be so many papers, all reporting a different *best* prediction system? Leaving aside all the issues discussed above, we note that researchers will generally spend more time exploring, debugging, extending their own favorite algorithm than some other algorithm taken from the literature. This creates an *experimenter bias*



Let n instances have actuals $y_1, y_2, .., y_n$. Prediction systems generate prediction $\hat{y}_i$ for instance i.
If $|y_i - \hat{y}_i|$ is AR_i (the *absolute residual* difference between prediction and actual) then MAR is the *mean absolute residual* $(\sum_i \hat{y}_i - y_i)/n$.
MRE_i and MER_i are the ratios AR_i/y_i and $AR_I/\hat{y}_i$. PRED(25) is the percentage of the n instances with $MRE_i \leq 25\%$.
MMRE and MMER are the averages $\sum_i MRE_i/n$ and $\sum_i MER_i/n$. MdMRE is the median MRE_i value.
Finally, the *balanced errors* are
$MBRE_i = |\hat{y}_i - y_i|/min(\hat{y}_i, y_i)$
and
$MIBRE_i = |\hat{y}_i - y_i|/max(\hat{y}_i, y_i)$

Fig. 5 Comparisons across 20 data sets and seven accuracy statistics

where the 'best' estimator found in an experiment just happens to be the latest one built by the researcher. It is important to stress that such bias is not necessarily some egotistical act on the part of researchers to discredit rival approaches. Rather, in their enthusiasm to demonstrate the value of some exciting new technique, they spend more effort on their preferred method than any other (e.g. a researcher may spend more time debugging or fine tuning their own algorithm than any other). Interestingly Michie et al. (1994) commented upon this phenomena in a major review of machine learning research nearly two decades ago.

3.1.5 Verification Bias

Another verification bias is the *accuracy statistic* used to score model performance (Fig. 5). Note that different performance statistics offer very different scores and therefore rankings to competing prediction systems on the *same* data set (Kitchenham et al. 2001; Myrtveit et al. 2005). Although some statistics may have preferable properties to others, an underlying difficulty is determining exactly what is intended by 'accuracy' and the specific goals of the would-be user of the prediction system. For example, risk aversion points towards a sum of the squares of the residuals type statistic whilst the need to manage a portfolio indicates that reducing bias is important. This issue is explored in two of our special issue papers (Angelis and Mittas 2012; Stensrud and Myrtveit 2012).

4 How to Reduce Conclusion Instability?

Having defined a problem, we now explore methods to reduce the problems. Note that the ideas of this section are very preliminary. The aim of this special issue was to highlight an issue that is under-explored in the current literature. Any discussion on how to reduce conclusion instability should be viewed as a work-in-progress document. Nevertheless, some promising research directions are offered below.

4.1 Avoid Invalid Comparisons

The first way to reducing conclusion instability is to acknowledge that it is a problem. For example, it is misleading to compare performance statistics from one paper to another when the two results are generated from different rigs (e.g. because they have been written by different authors). Researchers run the risk of having their papers rejected, unless they use the same rig to generate performance statistics for the treatments that they are exploring.

4.2 Full Reporting

Instead of merely reporting results based on measures of centre (e.g. mean or median), it would be helpful to include some discussion of the *variability* of those results. There are many ways to do this, e.g. using *visualizations* of the range of behaviors such as box plots or Fig. 4 (shown above). This is helpful as it gives some indication of how much an individual result might be expected to deviate from a reported measure of centre.

While visualizations can detect interesting patterns, they should be followed by *statistical tests* in order to check that any detected visual patterns are not over generalizations of the data due to, say, noise. Such tests should reflect on not just median behavior but also the *variance* around that median. There are many such tests and, to state a personal bias of the authors, we caution against tests that make parametric assumptions that cannot be justified (so Wilcoxon or Mann–Whitney rather than t-tests). Also, when researchers compare results between methods, they need to comment on more than just statistical difference. If two results are significantly different, but the difference in their central tendency scores is tiny, then the reader might be *indifferent* to the result.

Another way to study variability is through *sensitivity analysis* (Saltelli et al. 2000; Wagner 2007) where the conclusions of the papers are assessed by repeating the entire analysis to determine the sensitivity of the results to different experimental settings and parameters, for example:

- using, say, 10 times with 90% of the available training data.
- using different performance statistics;
- switching between leave-one-out and N-way validation;
- when using search-based techniques, adjusting the objective function.

We might conclude that *brittle conclusions* are those that are not found in the majority of those different treatments. In other words the conclusions are not stable but are highly sensitive to particular conditions.

4.3 Locality Studies

Note that if sensitivity analyses report that *all* conclusions are ‘brittle’, then that might lead to other kinds of studies. For example, if a conclusion does not hold across *all* the data sets, then analysts might consider *instance selection* studies to find subsets of the data where different conclusions hold. For example, analogy-based tools select *different* training data for each test instance (Shepperd and Schofield

1997). Elsewhere, Menzies et al. (2011) report useful results from clustering the data and then learning separate models on each cluster.

If conclusion change, depending on what data is used from different sources, then the issue might be to learn how little data is required before conclusions *stabilize* within one source. Two goals of these kinds of *stability studies* might be

- Studies on *minimality*: How few examples are needed before stabilization occurs?
- Studies on comparative *training rates*: Do different learners stabilize sooner?

4.4 Sampling Studies

Reporting experiments based on one data set may not detect the conclusion instability problem. Researchers need to base their conclusion on an analysis of multiple datasets. Clearly, sharing data strongly contributes to this process. When designing on-line repositories to store data, it is important to consider what *preservation policy* will enable others to access the data from many years to come. In this regard, we recommend against the use of proprietary software. Firstly, the formats of that software may exclude some users of the data. Secondly, if the software requires yearly license fees, then any financial shortfall in the repository will take the data offline. For example, the CeBase initiative was a multi-year project aimed at generating an experience base for software engineering. All that data is now inaccessible since, now that the project is over, there are no funds to maintain the servers or the software licenses.

In practice, given the current maturity of Web2.0 tools, there may be little added value in using proprietary software for building repositories. For example, despite the PROMISE repository (http://promisedata.org/data) using open source tools (WordPress), it is still accessible and used by hundreds of research teams. Of course it is important to ensure the quality of all archived data (Gray et al. 2011).

4.5 Blind Analysis

One way to avoid experimenter bias is by means of blind analysis (Kitchenham et al. 2002) where research teams are divided into two. Team One conducts the analysis, however, the treatment labels (e.g. competing prediction systems) are removed and meaningless (to Team Two) labels are substituted. This means the analysis by Team Two is independent since they do not know which treatment generates which result, so any bias towards, say, a locally-developed prediction system is removed.

4.6 Effect Size

Another factor to consider is not all significant differences between prediction systems detected by empirical studies have practical import.

- We may not prefer some prediction systems, despite superior prediction results, since they are are too complex to code, too slow to run, or generate wildly varying results, etc.

– We *do* prefer other predictions systems since they are simple to apply and the variation in their behavior is not excessive and that, for most datasets, they work reasonably well.

One open issue in our field is to operationalise *"preferred"*. Presently it is not usual to report effect size, however, it may provide useful insight into the relative merits of competing prediction techniques in practice.

4.7 Researcher Expertise

Effort estimation is a skill and different researchers are skilled at different things. Hence, when conducting a complex study with many tools, it is possible that some of those tools are being used in a sub-optimal manner. This is a problem since any conclusion that (say) "neural nets are bad for effort estimation" might really be a comment a researcher's lack-of-skill at configuring neural nets.

Hence, we advise that if researchers does not have skill at tool "X", they should (a) not use it; or (b) take the time required to learn that tool (which may take weeks to months to years); or (c) consult with experts in "X" before results about "X". Regarding the last point, perhaps it is wiser to conduct tool evaluation using multi-institutional consortiums where experts in "X" at one site can work with experts in "Y" at another site.

4.8 Improved Reporting Protocols

It is not enough to just report that a study used (e.g. linear regression) *without* detailing the data pre-processing applied prior to applying that prediction system. This is important since seemingly minor details can have major impacts on the results. For example, Keung et al. (2011), ranked 90 effort estimators built from ten pre-processors and nine learners. Seemingly small changes to the pre-processor had a significant impact on the ranking of a learner. For example, a $k = 1$ nearest neighbor algorithm jumped from rank 11 (one of the best) to rank 69 (one of the worst) when the pre-processor was changed from logging the numerics to dividing the numerics in bins of size $(\max - \min)/3$.

Further to the last section, we also advise that reporting protocols include some statement of how experienced are researchers in the tools they are using in their experiments. If a researcher is using tool "X", and they are not experienced in "X", then it would be honest to note that in the description of the experiment.

4.9 Learning Learners

Finally, we note one exciting, albeit complex, research possibility. Various researchers tune their data mining technology using feedback from the domain. The goal of this approach is to generate the right learner for the particular domain. Example work in this area include:

– Researchers who use one learner to tune the parameters of another. For example, prior to generating estimates, Corazza et al. use tabu search to tune the parameters of a support-vector machine (Corazza et al. 2010).

– Researchers who apply data miners to the output of the results generated by other data miners. The output of these *second-level learning* is to determine what features of data sets make them most suitable to different learners. For examples of this approach, see The STATLOG project (Michie et al. 1994) and Ali and Smith (2006).
– Learners that inputs background knowledge about distributions in a domain, then outputs a learner specialized to those distributions; e.g. see the AutoBayes research (Buntine et al. 1999; Fischer and Schumann 2003).

Learning learners is an active research area and much further work is required before we can understand the costs and benefits of this approach.

5 In This Issue

Our first paper takes an industrial perspective and argues that conclusion instability arises from none of the factors discussed above. Writing from a Microsoft perspective, Murphy argues, in *The Difficulties of Building Generic Reliability Models for Software* (Murphy 2012), that the computer industry is capable of producing generic predictive models, but *only if they are willing to apply the the same restrictions as other engineering disciplines*. Unfortunately, says Murphy, the limitations imposed on the development process to produce these models are too great for the majority of software development. In his view, conclusion instability is inherent due to the fast pace of change in the modern software industry.

Stensrud and Myrtveit discuss *verification bias*. Their paper *Validity and reliability of evaluation procedures in comparative studies of effort prediction models* (Stensrud and Myrtveit 2012) argues that a range of commonly used accuracy statistics used in effort estimation (MMRE, MMER, MBRE, and MIBRE) are invalid in the context of model selection, since each of them can systematically select inferior models. Furthermore, ranking agreement between the constituents of a composite accuracy statistic, does not prevent the selection of inferior models. In addition, when the constituents suggest contradictory rankings, they just contribute to increased conclusion instability.

Angelis and Mittas also discuss *verification bias*. Their paper *A Permutation test based on Regression Error Characteristic Curves for Software Cost Estimation Model* (Angelis and Mittas 2012) argues that conclusion stability arises from the way we compare our models. They propose a new inferential test, accompanying an informative graphical tool, which is more easily interpretable than the conventional parametric and non-parametric statistical procedures. Moreover, it is free from normality assumptions of the error distributions when the samples are small-sized and highly skewed. Finally, the proposed graphical test can be applied to the comparisons of any alternative prediction methods and models and also to any other validation procedure.

Sampling bias is discussed in Turhan's paper *On the Dataset Shift Problem in Software Engineering Prediction Models* (Turhan 2012). He postulates that conclusion instability arises when the data generating phoning changes properties. For data exhibiting such *dataset drift*, then yesterday's predictors may not work today. Software engineering community should be aware of and account for the dataset shift related issues when evaluating the validity of research outcomes.

Robles and Gonzalez-Barahona discuss *sample variance*. In their paper *On the reproducibility of empirical software engineering studies based on data retrieved from development repositories* (Robles and Gonzalez-Barahona 2012), they argue that among empirical software engineering studies, those based on data retrieved from development repositories (such as those of source code management, issue tracking or communication systems) are specially suitable for reproduction. However their reproducibility status can vary from easy to almost impossible to reproduce. This paper explores which elements can be considered to characterize the reproducibility of a study in this area, and how they can be analyzed to better understand the type of reproduction studies that they enable or obstruct. This characterization of studies and types of reproduction has allowed us to provide a simple method for understanding if a type of reproduction study can be performed: it is just a matter of comparing the attributes of the elements in the original study that should be reused in the reproduction one.

Finally, Azzeh's paper, *A Replicated Assessment and Comparison of Adaptation Techniques for Analogy-Based Effort Estimation* (Azzeh 2012) can be read as an example of the approach of Robles and Gonzalez-Barahona. In this paper, the author looks deeper into the options associated with adaptation in estimation by analogy, or EBA. The results and conclusions indicate that while no single EBA model was consistently better than any other, several specific models were generally more successful, and as such should form the basis for further investigations. That is, by better defining the experiment, the authors are able to increase the generality of the conclusions.

References

Ali S, Smith K (2006) On learning algorithm selection for classification. Appl Soft Comput 6(2): 119–138

Angelis L, Mittas N (2012) A permutation test based on regression error characteristic curves for software cost estimation models. Empir Softw Eng (Special issue on repeatable results in SE)

Azzeh MY (2012) A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. Empir Softw Eng (Special issue on repeatable results in SE)

Baldi P, Brunak S, Chauvin Y, Andersen C, Nielsen H (2000) Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16(5):412–424

Boehm B (1981) Software engineering economics. Prentice Hall, Englewood Cliffs

Buntine W, Fischer B, Pressburger T (1999) Towards automated synthesis of data mining programs. In: Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York

Chen Z, Menzies T, Port D (2005) Feature subset selection can improve software cost estimation. In: PROMISE'05. Available from http://menzies.us/pdf/05/fsscocomo.pdf

Corazza A, Di Martino S, Ferrucci F, Gravino C, Sarro F, Mendes E (2010) How effective is tabu search to configure support vector regression for effort estimation? In: Proceedings of the 6th international conference on predictive models in software engineering, PROMISE '10

Davey B, Priestley H (2002) Introduction to lattices and order, 2nd edn. Cambridge University Press, Cambridge, UK

Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In: 12th International Machine learning Conference. Morgan Kaufmann, San Mateo, pp 194–202. http://robotics.stanford.edu/users/sahami/papers-dir/disc.pdf

Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery in databases. AI Mag 17:37–54

Fayyad UM, Irani IH (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the thirteenth international joint conference on artificial intelligence, pp 1022–1027

Fischer B, Schumann J (2003) AutoBayes: a system for generating data analysis programs from statistical models. J Funct Program 13(3):483–508
Glass G, McGaw B, Smith M (1981) Meta-analysis in social research. Sage Publications, Beverly Hills, CA
Gray D, Bowes D, Davey N, Sun Y, Christianson B (2011) The misuse of the nasa metrics data program data sets for automated software defect prediction. In: EASE 2011. IET
Hall MA, Holmes G (2003) Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans Knowl Data Eng 15:1437–1447
Jørgensen M (2004) A review of studies on expert estimation of software development effort. J Syst Softw 70(1–2):37–60
Keung J, Kocaguneli E, Menzies T (2011) A ranking stability indicator for selecting the best effort estimator in software cost estimation. Autom Softw Eng. Available on-line at http://menzies.us/pdf/11draftranking.pdf (submitted)
Kitchenham B, MacDonell S, Pickard L, Shepperd M (2001) What accuracy statistics really measure. IEE Proc, Softw Eng 148(3):81–85
Kitchenham B, Mendes E, Travassos GH (2007) Cross versus within-company cost estimation studies: a systematic review. IEEE Trans Softw Eng 33(5):316–329. doi:10.1109/TSE.2007.1001. Member-Kitchenham, Barbara A
Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, El Emam K, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. IEEE Trans Softw Eng 28(8):721–734
Kocaguneli E, Gay G, Menzies T, Yang Y, Keung JW (2010) When to use data from other projects for effort estimation. In: IEEE ASE'10. Available from http://menzies.us/pdf/10other.pdf
Little R, Rubin D (2002) Statistical analysis with missing data, 2nd edn. Wiley, New York
Mair C, Shepperd M (2005) The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In: 2005 International symposium on empirical software engineering, p 10. doi:10.1109/ISESE.2005.1541858
Menzies T, Butcher A, Marcus A, Zimmermann T, Cok D (2011) Local vs global models for effort estimation and defect prediction. In: IEEE ASE'11. Available from http://menzies.us/pdf/11ase.pdf
Menzies T, Chen Z, Port D, Hihn J (2005) Simple software cost estimation: safe or unsafe? In: Proceedings, PROMISE workshop, ICSE 2005. Available from http://menzies.us/pdf/05safewhen.pdf
Michie D, Spiegelhalter D, Taylor C (eds) (1994) Machine learning, neural and statistical classification. Ellis Horwood Series in Artifical Intelligence. Ellis Horwood, Chichester, Sussex, UK
Murphy B (2012) The difficulties of building generic reliability models for software. Empir Softw Eng (Special issue on repeatable results in SE)
Myrtveit I, Stensrud E, Shepperd M (2005) Reliability and validity in comparative studies of software prediction models. IEEE Trans Softw Eng 31(5):380–391
Robles G, Gonzalez-Barahona JM (2012) On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. Empir Softw Eng (Special issue on repeatable results in SE)
Saltelli A, Tarantola S, Campolongo F (2000) Sensitivity analysis as an ingredient of modeling. Stat Sci 15(4):377–395
Shepperd M, MacDonell S (2012) Evaluating prediction systems in software project estimation. Inf Softw Technol, in press
Shepperd M, Schofield C (1997) Estimating software project effort using analogies. IEEE Trans Softw Eng 23(12):736–743
Stensrud E, Myrtveit I (2012) Validity and reliability of evaluation procedures in comparative studies of effort prediction models. Empir Softw Eng (Special issue on repeatable results in SE)
Turhan B (2012) On the dataset shift problem in software engineering prediction models. Empir Softw Eng (Special issue on repeatable results in SE)
Wagner S (2007) Global sensitivity analysis of predictor models in software engineering. In: Predictor models in software engineering, PROMISE'07. IEEE Computer Society, Los Alamitos
Witten IH, Frank E (2005) Data mining, 2nd edn. Morgan Kaufmann, Los Altos
Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project defect prediction. In: ESEC/FSE'09

Tim Menzies received his PhD from the University of New South Wales, Sydney, Australia (1995) and is the author of over 200 referred papers. In his research, he explores how AI and data mining can help humans in real-world situations. He is a leading figure in the field of applying data mining to SE data and is one of the founders of the PROMISE conference on repeatable experiments in software engineering. A former research chair for NASA (2002–2003), Dr. Menzies is now an associate professor at the West Virginia University's Lane Department of Computer Science and Electrical Engineering. He is an associate editor of the Automated Software Engineering Journal; the Empirical Software Engineering Journal; and the Journal of Visual Languages & Computing. He will serve as PC-co-chair for IEEE ASE 2012. For more information, see his home page http://menzies.us.

Martin Shepperd received a PhD in computer science from the Open University in 1991 for his work in measurement theory and its application to empirical software engineering. He is professor of software technology at Brunel University, London, UK. He has published more than 150 refereed papers and three books in the areas of software engineering and machine learning. He was editor-in-chief of the journal Information & Software Technology (1992–2007) and was Associate Editor of IEEE Transactions on Software Engineering (2000–2004). Presently he is an Associate Editor of the journal Empirical Software Engineering. He was Program Chair for Metrics 2001 and 2004 and ESEM 2011. He has supervised more then 20 PhD students to completion and been external examiner for more than 25 including for universities in Germany, Sweden, Malta, Norway, Pakistan and Australia. He also previously worked for a number of years as a software developer for a major UK bank.