An empirical study into the use of measurement to support OO design evaluation

Colin Kirsopp, Martin Shepperd, Steve Webster

Empirical Software Engineering Research Group
School of Design, Engineering & Computing,
Bournemouth University,
Talbot Campus,
Poole, UK

Email: ckirsopp, mshepper, swebster @bournemouth.ac.uk

Abstract

This paper describes a case study into using measurement to aid design evaluation. The study was carried-out as an assignment by final year B.Sc. students from an I.T. course. Approximately half of these were part-time students working full-time in industry and the others were full-time students who had already undertaken a one-year industrial placement. After some preparatory tuition, the subjects were given a small system (15 classes) written in Java and a tool to enable them to apply a wide variety of metrics to this system. They were then asked to use measurement results and design heuristics to help them critique the design of the system.

The main results from this work are that:

- *measurement was found to be a useful aid to design evaluation;*
- *following the study, most subjects said that they would now be more inclined to use measurement than they were previously;*
- *support is needed for measurement selection as well as measurement collection;*
- *there were significant differences in the way the task was carried out between the full-time students and the industrial subjects*

Keywords: object-orientation, case study, architecture evaluation, design heuristics

1 Introduction

The OO metric literature abounds with descriptions of metrics for measuring a multitude of aspects of OO designs. Many of these metrics are described as being useful for identifying design problems. However, there is a lack of published work demonstrating the utility of these measures in actually aiding the design process. The work described in this paper was prompted by this lack of empirical work.

A simple way to locate this work in the measurement field is to explain where it fits into the framework proposed by Whitmire [16]. This framework identifies 3 views onto 4 objects of measurement. The *strategic view* considers the long-term view of the organisation, the *tactical view* considers the current project and the *technical view* considers technical design issues. The objects to measure are: *processes*, the things we do; *projects*, instances of processes; *products*, the deliverables created and *resources*, such as people or time. This work is concerned with the technical view of design products. The framework has a further dimension that considers the role of technical measurement within design. Whitmire identifies the following roles for measurement; estimation, prediction, assessment, comparison and investigation. This study falls under the heading of assessment. That is, the evaluation of a single system by checking product measures against predefined thresholds (this contrasts with comparison, which compares the measures from one system against the same measures from a second system). Rather than assigning our own arbitrary thresholds, we utilised suggestions made in the OO heuristics literature.

The work presented in this paper is a case study involving the assessment and redesign of a small application. The subjects were to assess the architecture of a given application, with the aid of metrics and heuristics. They were then to redesign and implement their proposed changes, and evaluate their new design against the original. The re-design was intended solely to improve the architecture of the application without changing its external behaviour. To maintain the necessary brevity, this paper only concentrates on the first part of this task, the original design assessment.

The rest of the paper is structured as follows. In the next section, a brief introduction is given to OO design heuristics and the relationship between heuristics and measurement. The following section describes the design and performance of the case study. A number of interesting results, which emerged from the analysis of the study, are then discussed. The final section attempts to draw some conclusions from this work and points toward future work.

2 Design heuristics

Design heuristics are rules of thumb that can guide designers as they choose between various alternatives. They are separately gleaned pieces of distilled design experience that can be shared. They capture the experience of skilled designers. These partial definitions begin to convey what design heuristics are about but the topic is broad and lacks a precise definition.

Object oriented design heuristics have arisen in a number of ways, although they have not always been labelled as heuristics. Sets of general recommendations have accompanied design methods [4, 9]. Guidelines on specific topics have come out of narrowly focussed pieces of research [11, 12]. More recently, experience-based sets of heuristics have been proposed independently of any particular design method [10, 15]. There has also been the emergence of empirically based heuristics, in which measurement values falling outside of a statistically derived ‘normal’ range are deemed to be violations [2].

Since heuristics are simply rules of thumb and typically without supporting empirical evidence, how do we know that they constitute good advice? A high degree of agreement between the heuristics from different authors might increase confidence in the validity of their heuristics (provided the authors arrived at their heuristics independently). Within the literature, there appears to be little contradiction between authors. Differences do exist in the choice of particular threshold values, but on the general trends, there appears to be consensus on most issues.

When heuristics are applied, there are occasional apparent contradictions that need explanation. This point is largely one of context. In their abbreviated ‘snappy’ form, heuristics are presented without any explicit context and this can lead to apparent contradictions. Where an apparent contradiction appears, consideration of the rationale for each heuristic, and it’s associated context, will usually show that the two heuristics should not be applied in the same situation. On some occasions, two opposing heuristics can be applied within the same context and this implies a trade-off, which can be resolved only through an explicit design decision.

It bears reiteration that heuristics are only guidelines, not hard-and-fast rules. Decisions can be taken to allow particular heuristic violations if, in the particular situation under consideration, they are not considered problematic.

2.1 Relation to metrics

The three example heuristics shown below have different styles of assessment. The first heuristic involves a value judgement on the part of the designer. We will call this a semantic heuristic as its application depends upon the meaning of the class. Conformance to the second heuristic could be checked purely based on the structure of the design/code. We will call this a syntactic heuristic. Some heuristics fall between these two classifications. Although they are to some extent measurable, some semantic interpretation is also required. The third heuristic falls into this hybrid category. We can measure its size and coupling, but still have to assess semantically whether it has too much control and responsibility.

An object removed from the context of the immediate application, should, in isolation, still represent a coherent and complete object-oriented concept. [3]

Avoid having too many services per class. Each class typically has no more than six or seven public services. [8]

Do not create god classes/objects in your system. [15]

Metrics are related most naturally to syntactic heuristics. Measurement values may be used to automatically check the conformance of a design to a syntactic heuristic. Heuristics may be created specifically for particular metrics [2]. It is also possible to define metrics specifically to automate the detection of violations of particular heuristics.

Following Whitmire's classification, this paper is concerned with the technical assessment of design products using metrics. This usage 'compares measures from a product or process to predetermined values…' These predetermined values may be supplied from heuristics (among other sources).

3 The case study

3.1 The subjects

The subjects selected for this case study were those taking a unit at Bournemouth University entitled 'Object-oriented software engineering' (OOSE). The unit was selected because it was an advanced unit (presuming a level OO knowledge), also, the necessary preparation and task itself fitted well into the remit of this course. Originally there were 16 subjects, 9 were final year B.Sc. students and 7 were working full-time in IT (taking the unit for a part-time course). One of the students has been excluded from the analysis, since they failed to complete the task in the manner requested, leaving 8 students and 7 industrialists.

The first part of the OOSE course introduced the subjects to Java, the latter part of the course covered design evaluation, design patterns, design presentation using the UML and the process of OO development. The work on design evaluation covered both heuristic-based evaluation and OO design metrics. The subjects received 3 one-hour lectures on these topics and they spent an equivalent amount of time on paper reviewing and focused discussion. The metrics teaching material focused on the work of Chidamber & Kemerer [6, 7] and Brito e Abreu [1, 2], as well as covering the Lorenz & Kidd metrics [14]. The heuristics material was based on our own review of OO design heuristics [13] and the book by Riel [15].

The chart below gives some idea of the subject's levels of OO experience prior to the OOSE course. Shown are the maximum levels of experience of the subjects in any OO language and in any OO design method. Almost half of the subjects had used an OO language professionally and all but two had prior experience of at least the level of a formal taught course. There is generally less experience of OO design methods. Only two of the subjects had used any of these professionally and three had not received any prior formal training in OO design.

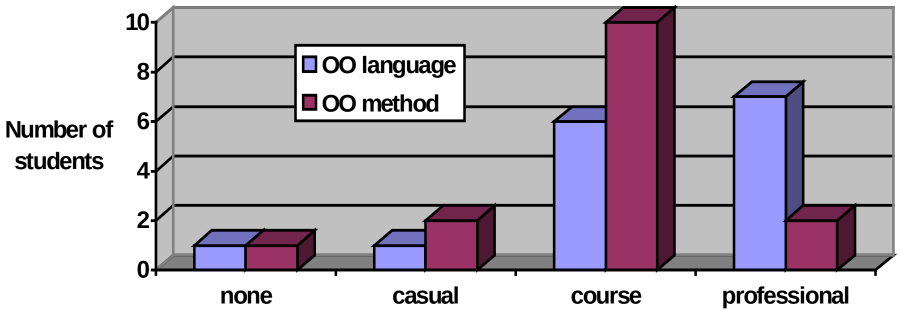

Figure 1: Subject's experience in object-orientation

3.2 The task

The application used for the study was the Second Language Vocabulary Tutor (Vocab Tutor). This is a small application to allow learners of a second language to increase their vocabulary through a process of actively learning new words and testing themselves on their progress. It is coded in Java 1.1 and has a GUI interface built using the standard AWT components. The Vocab Tutor consists of 15 classes and 1500 lines of code[1]. Five of the classes make up the GUI and the remaining ten implement the problem domain. The application contains a simple inheritance hierarchy with a depth of one, a single parent class with a number of child classes.

The subjects were provided with a measurement tool capable of the automatic extraction of design metrics from Java source code. This tool was developed by one of the authors, as part of his ongoing Ph.D. research. The tool allows the definition of new metrics by the users and it allows the grouping of individual metrics into user defined metric sets, for easy repeat collection. The tool also allows ad-hoc investigation of the data model used to store design information. This can be useful in further investigating notable measurement results.

A large number of metrics were pre-installed into the tool, including metrics suites from Chidamber & Kemerer [5], Lorenz & Kidd [14] and Abreu [2]. This meant that the subjects did not have to code any metrics if they restricted themselves to those in the pre-installed suites.

[1] Any readers wishing to obtain a copy of the case study materials (or analysis data) should contact the primary author.

3.3 Data collection

Two different elements made up the data collected for this case study. Firstly, each subject was required to produce two written reports, one for each section of the task. Secondly, after the reports were handed in, the subjects were asked to complete a debriefing questionnaire. The data collected from the questionnaires and the reports were organised into a case study database and stored in relational format (MS Access).

The reports produced by the subjects had to be codified into a form suitable for further analysis. In considering the assessment of a single design, the main output of the assessment is the set of problems identified. Additional information might include any justification for believing particular elements of the design to be problematic. From the phrasing of the task, these problems are likely to appear as heuristic violations. These violations might be justified using measurement results.

As mentioned earlier in the description of design heuristics, a heuristic violation in itself may not represent a problem. If the violation occurs outside of the intended context of the heuristic, the apparent violation will have resulted from a misapplication of the heuristic.

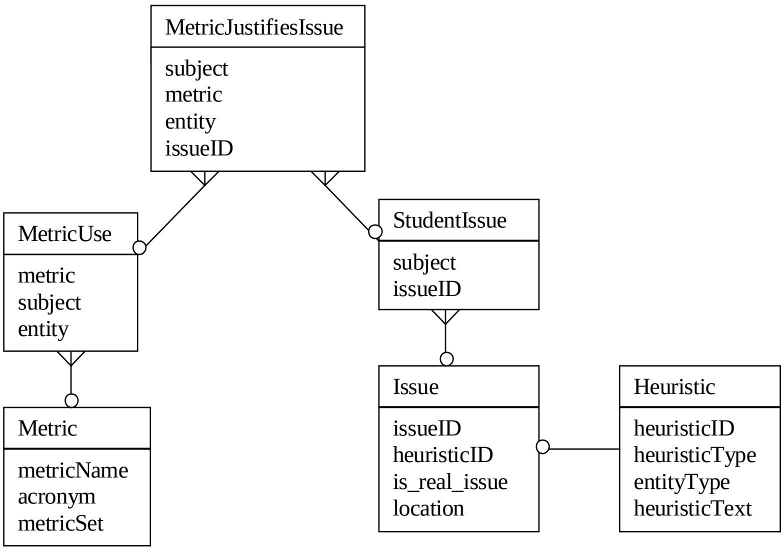

Figure 2: An aspect of the case study database model

Even within the correct context, heuristics are only guidelines and not 'hard-and-fast' rules that always apply. For these reasons, the authors wish to separate the notions of problem and heuristic violation. We will use the term 'issue' to describe any reported heuristic violation, and 'real issue' to describe a heuristic violation that we (subjectively) believe indicates a design problem. The authors have identified their own list of design weaknesses in the application. These weaknesses have been characterised by severity into trivial, significant and important. This list has been further refined in the light of the subjects' analyses. The data model in figure2 should help to clarify the relationships between these variables.

Any mention of a heuristic violation in the subjects' reports was entered into the case study database. References to measurement results and the use of measurement results to justify these issues were similarly recorded. Additional information was also recorded, such as the nature of the heuristic that was violated, the location of the violation, etc.

The debriefing questionnaire was designed to collect both information not covered by the reports and information to add corroboration to the report data. It included sections on attitudinal information, background experience, and time expended.

4 Results

The results presented in this section are grouped into five sections. First is an analysis and discussion of how the subjects selected the metrics they used to perform the assessment. The second section covers an analysis of the types of issue found. Patterns of use for particular metrics are analysed in the next section. In the fourth section, an analysis is performed to check for any differences in the way the industrialists and the students went about the task. Finally, some data is presented concerning the subjects attitudes toward software measurement.

4.1 Metric selection

The first question, which faced the subjects when performing the task, was the selection of the metrics they were to use. How would they go about this selection? Below are two pie charts that show the results from two of the questions from the debriefing questionnaire.

1. *'Did you consider taking a pick'n'mix approach to selecting metrics? (taking some metrics from one authors set and some from another)'*
2. *'Did you consider using a different set of metrics for analysing the design*

refinement than you did for the initial assessment?

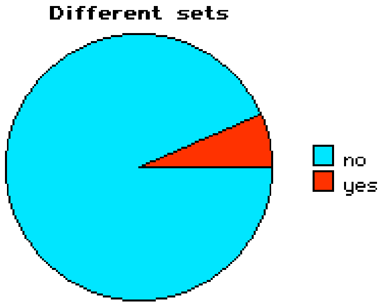

Figure 3: Results from the metric selection section of the questionnaire

The results from the first question show that over half of the subjects didn't even consider selecting across suites of metrics proposed by different authors. They simply chose a set, as given, and tried to apply it.

Although this paper concentrates on the evaluation section of the task, the subjects also had to redesign the system based on their initial assessment, and compare their new version of the system to the original. From a measurement perspective, the nature of this second task is a qualitatively different task from the first [16]. Firstly, the measurement results are now being used to compare relative 'quality', rather than being used for finding specific problems. Secondly, the measurement results are being compared with similar results from another system rather than a threshold set by a heuristic. Question 2 in this section was intended to discover whether the subjects recognised this distinction and considered using different metrics to achieve these differing ends. The answer, quite overwhelmingly, is that they did not.

If most subjects did not consider selecting metrics from different authors, or using different sets of metrics for different tasks, how did they rationalise their selection? The following are typical answers given by the subjects when asked for a rationale for their metric selection.

'Ran them all and used the ones that best fit the process I was going to use.'

This is more a lack of selection. Such a shotgun approach seems likely to be very inefficient, particularly when scaled up to large applications.

'By …deciding which set was the most suitable to the application.'

This kind of vague reply was typical. It suggests a general lack of thought given to metric selection.

'The only suite I knew well and understood was CK.'

'The ones I could find information about.'

This type of reply shows the importance of comprehensible and accessible information about metrics. If you can't find information about a metric, or you can't understand that information, you won't use the metric. Accessibility of information seems to have been the main reason for the lack of use of the Lorenz and Kidd metrics [14]. Papers describing the CK and MOOD metrics were given to the subjects. The Lorenz and Kidd metrics are described in a book and as such were not so easily (cheaply) available.

4.2 Analysis of issues

As previously mentioned, issues are heuristic violations. They can be divided into four quadrants as shown in table1. The found column contains the number of distinct issues identified by the subjects. Those issues in the 'not found' column are those issues identified by the authors, but not found by any of the subjects. The rows divide the issues based on whether the authors considered those particular heuristic violations highlighted real design problems. Only three of the quadrants are used as the number of false issues not found is not assessable (possibly infinite!).

	Found	Not found
Real issue	86	98
False issue	29	

Table 1: Issues by issue quadrant

This classification into 3 categories can be used for further analysis. Table2 shows the distribution of issues by the assessment type of heuristic violated and the issue quadrant. These figures include all subject issues not just distinct issues, as is the case above.

	Real-issues found	Non-issues found	Real-issues not found	total
semantic	15	12	5	32
hybrid	89	11	13	113
syntactic	84	16	74	174
total	188	39	92	319

Table 2 : Contingency table of heuristic-type by issue quadrant

A high degree (44%) of semantic issues raised by subjects were erroneous. This might indicate that the subjective interpretation necessary in applying semantic heuristic makes them more difficult to identify correctly. It may also be the case that where syntactic metrics have been used as proxies for measures of semantic properties, they have been largely misleading. There is also a notably high fraction of the syntactic issues that were not found (43%). At first sight this seems counter intuitive, as syntactic issues should be amenable to automated metrics and so should be easily found. Further investigation has shown that these unfound issues are mostly trivial issues, in the fine detail of the design. The subjects generally did not collect suitable metrics to highlight these problems. This large section of trivial syntactic issues can be clearly seen in the table3, which shows the break down of real-issues by their level of importance. This table also shows that semantic heuristics lead to few, but generally important issues.

	trivial	Signif-icant	Import-ant	total
semantic	2	5	13	20
hybrid	5	56	41	102
syntactic	65	63	30	158
total	72	124	84	280

Table 3: Heuristic-type by issue significance

4.3 Analysis of metrics

In this section, the manner in which the actual metrics were employed will be examined. The uses of a metric can be considered at a number of levels. A measurement result from the metric could be referenced in the subjects report. This measurement result may, or may not, have been used to justify a particular issue. The issue may, further more, be considered (by the authors) either to be a 'real' problem with the design or a non-issue. Figure4 displays the uses of metrics at these three levels.

Reading from left to right, the first four metrics are from the Lorenz & Kidd set [14]. Each of these metrics was only referenced once and not used to justify an issue. The next six metrics are from the MOOD metrics [1, 2]. The final six metric are those proposed by Chidamber and Kemerer (CK) [5-7]. A point to note is the obvious difference in usage between the MOOD metrics and the CK metrics. This is due to the different nature of the metrics and the way the uses are counted. A use is defined as the application of a particular metric to a particular entity by a particular subject. The entity measured by the MOOD metrics is an entire system. There is therefore only scope for one use of each of these metrics by each subject. The CK metrics on the other hand measure classes. This means that, potentially, each subject may use each CK metric with each class in the system.

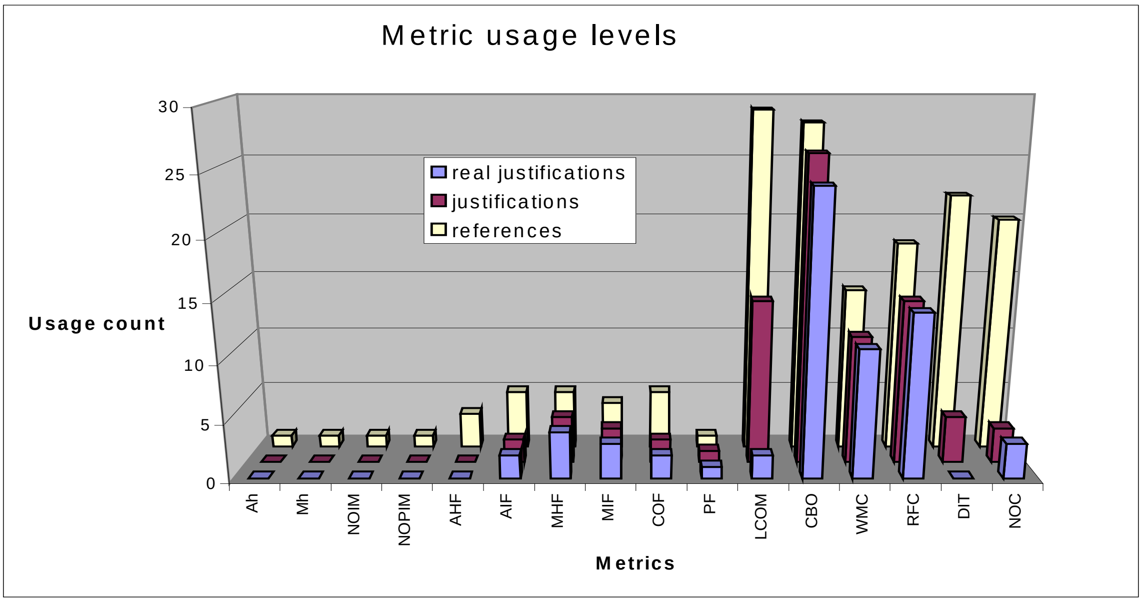

Figure 4: Metric usage, by metric and usage type

Reading the diagram from back to front: firstly we see the total number of references to each metric; then the subset of these references which were used to justify issues; then the subset of these justifications which were taken to be 'real' issues. A large drop from the references (back) to the justifications (middle) indicates that few of the references were used in justifying issues. This may mean that there were few problems in the system that the metric was helpful in indicating, or that the results of the metric were difficult to interpret into concrete problems. A large drop from the justifications (middle) to the real justifications (front) indicates that few of the issues justified were considered to be real issues, i.e., the metric results led to erroneous conclusions (or had been misinterpreted).

Considering some specific examples, CBO and LCOM are both heavily referenced. This indicated that the subjects thought that there were coupling and cohesion issues to be considered within the system. A clear difference between these metrics can be seen in the rate at which their use declines toward the front of the chart. CBO decreases very little from front to back, indicating that it clearly indicates particular problem areas to the subject and the problems are identified are consider to be 'real' problems. LCOM on the other hand halves in value between references and justifications. This may mean that there were less cohesion problems in the system, or that subjects had difficulty finding cohesion problems using this metric. The LCOM value falls almost to nothing when we consider real justifications. This means that on most occasions where the subjects interpreted LCOM's result as showing a problem, they were mistaken.

The DIT and NOC metrics have very similar patterns of use, reasonably high references and then very few justifications. This appears to be largely due to the size and 'shape' of the system. There will be little scope for finding problems using inheritance-based counts in a system containing only 1 level of inheritance (excluding system classes).

The questionnaire included the opportunity for the subjects to express their views about the usefulness and the understandability of the metrics that they selected. Responses were given on a numeric scale between 1 and 4 (4 being the most useful/understandable). The table below shows the median values (metrics on which fewer than 5 opinions were expressed have been excluded).

Metric	Useful	Understandable
CBO	4	4
WMC	4	3.5
RFC	3.5	2
COF	3	4
POF	3	3
NOC	2.5	4
DIT	2	4
AHF	2	4
MHF	2	4
AIF	2	4
MIF	2	4
LCOM	2	3

Table 4: Subjects opinions on the metrics they selected

Note the parallels between this table of subjective views and the analysis of actual usage earlier. LCOM's possible problems again appear with the equal lowest usefulness and the equal second lowest understandability. The subjects did not find DIT or NOC particularly useful for evaluating this system although they were easily understood. Low values for usefulness of the MOOD metrics reflect the difficulty of trying to infer specific problems from system wide metrics.

4.4 Difference between students and industrialists

It has been a point of discussion for some time whether the results of empirical work based on student subjects gave an accurate picture of how industrialists would behave. Since this case study contains an almost equal split of students and industrialists it presents an opportunity to add some data to this discussion.

The data for the number and significance of issues identified appears to show a clear distinction between students and industrialists.

The number of issues found by the individual students and the industrialists (data not shown) was significantly different (2-Sample t-Test $\alpha = 0.05$). There is also a significant ($\alpha = 0.001$) difference in the distribution of the importance of the issues found between the two groups.

Although there may be a temptation to say that these differences are due to the student-industrialist difference between the groups, there are a number of other differences between the groups that might contribute to these results. For instance, the 'industrialists' are generally older (more mature) than the full-time students, their mode of work as part-time students may have affected the way they performed the task, etc. However the results are interpreted, there was clearly a difference in the way these two groups performed (or at least reported) this task. Furthermore, if only full-time students had been used the results from this case study would clearly have been different.

	Count	Mean	Median	Min	Max
Students	8	21.4	21	6	37
Industrialists	7	8.0	6	4	18

Table 5: Summary of issues found by subject type

	Count	trivial	significant	important	total
Students	8	10	88	43	141
Per subject		1.25	11.0	5.38	21.4
Industrialists	7	1	14	32	47
Per subject		0.14	2.0	4.57	8.0

Table 6: Significance of issues found by subject type

4.5 Subjects views on measurement

The three pie charts shown below represent the answers given in the questionnaire concerning the subjects' views on measurement.

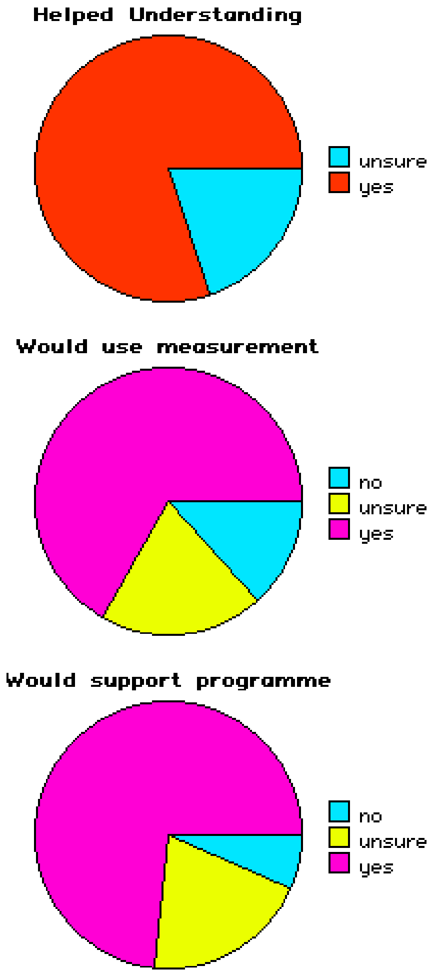

Figure 5: Subjects' views on measurement

Question 1: '*Do you feel that relating ideas about quality to measurement data has helped you understand design quality more fully?*'

Most subjects clearly thought that it had. This may indicate that measurement has a role in the teaching of OO design.

Question 1: '*Given suitable tool support, would you choose to apply measurement to help you design future projects?*'

Again, the replies were generally positive.

Question 3: '*Would your experience with this assignment make you more supportive of a suggestion to implement a measurement programme in a company for which you worked?*'

This question considered the *change* in attitude toward the implementation of a measurement programme following an experience of using measurement. Very few of the subjects seem to have considered this case study a negative experience.

5 Discussion

5.1 Threats to validity

It must be acknowledged that this study took place in a university environment as an assignment, and not in the work place on a real project. In general terms, it would be difficult to try to predict whether, and in what way, this may have affected the results. One specific problem is the possibility that the reports and questionnaires reflected what the subjects believed their tutor (one of the authors) wanted to hear. To alleviate this problem, it was made clear to the subjects that their questionnaire answers would in no way affect their marks. They were also informed that the analysis of the questionnaires and reports would only begin after the assignment marking had been completed.

Another potential threat is the size of the system used in the case study, 15 classes is not an industrial scale piece of software. However, the task was essentially a design review, and design reviews of this detail would only be done in practice on small subsystems. A complication is that when reviewing a subsystem, the context of the entire system must be considered and this was not present in the case study. As has already been mentioned, the size of the application rendered some of the metrics less useful than might have been the case in a larger application. The example mentioned previously was that DIT was of little use in a system containing only one level of inheritance.

In dealing with a particular application in such depth, some of the results are obviously dependent on the set of potential problems with this application. Some metrics or heuristics may have been used heavily because there were problems with this application that they were useful in highlighting. Similarly, some generally useful metrics may have been underused because the application was not weak in their areas of use.

Generally, the levels of experience of the subjects in OO design and in the use the Java language were not as high as we would have liked. Better levels of experience would have given us more confidence in the applicability of the results to industry.

5.2 Summary of results

The views expressed by the subjects were generally positive, with the large majority of both students and professionals saying that, given suitable support, they would use measurement for future projects.

Using syntactic heuristics leads to larger numbers of more accurate issues, but care must be taken to separate the trivial from the important. Violations of semantic heuristics are almost always important issues, however, their subjective nature means that great care must be taken to ensure they are being correctly interpreted and applied.

Although measurement was useful in finding problems, it was also seen that metrics could easily be misapplied or their results misinterpreted. Some metrics were markedly less reliable than others at discovering the problems they were purported to highlight. Another notable point is that the subject's choice of metrics was often not suitable for the task at hand. It would seem that support for metric selection, as well as metric collection, is essential for successful software measurement.

There were significant differences in way the full-time students and the industrialists carried out the study's task.

5.3 Future work

The work presented here is only part of the analysis from the evaluation section of the case study. Analysis of the data also reveals patterns in issue finding depending on the type of entity being assessed (class, method, hierarchy, etc.). There is also evidence that different styles/approaches to assessment were used by different subjects. This difference might be exploited to improve fault detection in design reviews, by utilising the reduced issue-overlap between styles in choosing review teams.

This work suggests that there may be differences in the way students and industrialists perform software tasks. Since much of the empirical work published within software engineering utilises student subjects, it would seem prudent to undertake work to directly test the validity of generalising such studies to industry.

This paper has concentrated on the application of measurement to design assessment, but this is just one use of measurement within the software process. Empirical studies might also be used to investigate the use of measurement in areas such as: initial design formation; design comparison during iterative design refinement (covered in the second part of this case study); and maintenance tasks.

6 Conclusion

Acres of journal space have been taken up in descriptions of particular measures. We have received a plethora of validation schemes: measurement theoretic, axiomatic, property-based. Although these foundations need to be laid, they are of limited interest to industry. It's time to start thinking about measurement utility. How can software measurement benefit the software industry? How can we maximise the usefulness of measurement in aiding particular software tasks? It's time to think about technology transfer. Software measurement must ultimately become a weapon in the software engineers armoury, or fade into obscurity like so many other tarnished 'silver bullets'.

7 References

[1] Abreu, F.B. and R. Carapuca, 'Candidate Metrics for Object-Oriented Software within a Taxonomy Framework', *J. of Systems & Software*, 26, pp87-96, 1994.

[2] Abreu, F.B., M. Goulao, and R. Esteves. 'Toward design quality evaluation of object-oriented software systems', in *Proc. 5th International Conference on Software Quality*. Austin, Texas, USA: 1995.

[3] Berard, E.V., *Essays on object-oriented software engineering*. Prentice Hall: 1993.

[4] Booch, G., *Object-Oriented Design: with applications*. Benjamin Cummings: 1991.

[5] Chidamber, S.R., D.P. Darcy, and C.F. Kemerer, Managerial use of object oriented software metrics. Working paper No. 750, Kratz Graduate School of Business, 1997.

[6] Chidamber, S.R. and C.F. Kemerer, 'Towards a metrics suite for object oriented design', *SIGPLAN Notices*, 26(11), pp197-211, 1991.

[7] Chidamber, S.R. and C.F. Kemerer, 'A metrics suite for object oriented design', *IEEE Transactions on Software Engineering*, 20(6), pp476-493, 1994.

[8] Coad, P., 'OOD Criteria, Part3', *Journal of Object Oriented Programming*, (September), pp67-70, 1991.

[9] Coad, P. and E. Yourdon, *Object-Oriented Design*. Yourdon Press/Prentice Hall: 1991.

[10] Coplien, J.O., *Advanced C++ : programming styles and idioms*. Adison Wesley: 1992.

[11] Daly, J., *et al.* 'Evaluating the effect of inheritance on the maintainability of object-oriented software', in *Proc. Empirical Studies of Programmers*. Washington, DC: 1996.

[12] Hursch, W.L., *Should Superclasses be Abstract?*, in *LNCS 821 ECOOP 94 European Conference on Object_Oriented Programming*, M. Tokoro and R. Pareschi, Editor, Springer Verlag: 1994.

[13] Kirsopp, C. and S. Webster, 1996. 'Object-Oriented Design Heuristics BOF Session'[online]. Available from: www.trireme.com/ot97/posters/kirsopp01 [Accessed 01/05/99]

[14] Lorenz, M. and J. Kidd, *Object-Oriented Software Metrics*. Object-Oriented Series, Prentice Hall: Englewood Cliffs, N.J., 1994.

[15] Riel, A.J., *Object oriented design heuristics*. Addison Wesley: Reading, Massachusets, USA, 1996.

[16] Whitmire, S.A., *Object-oriented design measurement*. John Wiley & Sons, Inc. 1997.