Predicting Understandability of a Software Project: A Comparison of Two Surveys

Ali Afzal Malik
Center for Systems and Software Engineering
University of Southern California
Los Angeles, CA 90089-0781, U. S. A.
1-213-740-6470
alimalik@usc.edu

Barry Boehm
Center for Systems and Software Engineering
University of Southern California
Los Angeles, CA 90089-0781, U. S. A.
1-213-740-5703
boehm@usc.edu

ABSTRACT
This paper summarizes the results of an empirical study conducted to explore the relative importance of eight pertinent COCOMO II model drivers in predicting the understandability of a software project. Here understandability of the project is measured solely from the perspective of the development team. This empirical study employed a survey that was targeted towards experienced practitioners. Results from this survey are compared with the corresponding results from a prior similar survey conducted on graduate students. While this comparison reveals some interesting differences between the importance given to each of the eight model drivers by these two categories of individuals it also highlights some subtle similarities.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics.*

D.2.9 [**Software Engineering**]: Management – *cost estimation, time estimation.*

General Terms
Measurement, Economics, Experimentation.

Keywords
Empirical study, COCOMO II, model drivers, survey, understandability.

1. INTRODUCTION
The presence of vague and incomplete requirements still continues to be one of the major reasons for the failure of software projects [2]. According to a study by James Martin [4], 56% of all the software bugs can be traced back to the errors made in the requirements phase. This study indicates that approximately half of these defects are caused by missing requirements and the other half is due to unclear, ambiguous, and incorrect requirements. The importance of clarity in requirements is further corroborated by the Standish Group's CHAOS report [6] which ranks "clear statement of requirements" among the top three reasons for the success of a project.

Given that clear, crisp, and unambiguous requirements are the key to success of a software project one is faced with several challenging questions. Amongst them, one of the most important questions is: "How one can determine whether the requirements of a software system are clear or not?" Another way of looking at the same problem is to pose a slightly different question: "How well have the stakeholders of a project understood its requirements?"

One of the first steps in addressing this critical question is to quantify the abstract notion of the understandability of a software project so that it can be measured objectively. This would enable one to take appropriate corrective action at the right time and allow providing answers to questions such as "How much requirements engineering is enough?"

Table 1. Relevant COCOMO II model drivers

S#	Model Driver	Description
1	PREC	Product precedentedness
2	RESL	Architecture/Risk resolution
3	CPLX	Product complexity
4	ACAP	Analyst capability
5	PCAP	Programmer capability
6	APEX	Applications experience
7	PLEX	Platform experience
8	LTEX	Language and tool experience

This paper is a continuation of our earlier work [3] which attempted to quantify this abstract notion of the understandability of a software project by using a subset of the inputs of a software cost estimation tool – COCOMO II [1]. These inputs, reproduced in Table 1, represent drivers of the COCOMO II model and are available as early as the project's inception phase when project planning activities are going on. These model drivers were combined in a weighted sum formula devised to predict the understandability of a software project. This formula was tested for its predictive power on 24 academic projects [5]. It was found that it correctly predicted the understandability of these projects in more than 80% of the cases.

The understandability of a software project – abbreviated as UNDR – is defined here in exactly the same way as in our previous work [3]: "the degree of clarity of the purpose and requirements of a software system to the developers of that system at the end of the Inception phase". Note that this definition considers the perspective of only one major stakeholder i.e. the development team. To further ensure consistency, our methodology here also matches our earlier work. We use the same technique in coming up with weights that indicate the relative importance of these model drivers in predicting UNDR. This time, however, instead of focusing on the students we have surveyed a different category of individuals – experienced practitioners.

The remaining paper is organized as follows. Section 2 describes the methodology of the survey we conducted on experienced practitioners to determine their perception of the relative importance of each of the chosen eight COCOMO II model drivers. The results of this survey are summarized in Section 3. Section 4 compares and contrasts the results of this survey with our previous work. Section 5 briefly discusses some important qualitative aspects of these results and highlights some threats to their validity. Our major findings are summarized in Section 6 which also talks about the future work in this area.

2. SURVEY METHODOLOGY

This survey comprised a one-page questionnaire [7] that was self-contained in the sense that it provided all the background information needed to fill the survey. In particular, it provided the definition of understandability being used in this context as well as the description of each of the eight COCOMO II model drivers. This was done to minimize guesswork and prevent confusion due to multiple conflicting interpretations.

The questionnaire asked the participants to rank each of the eight relevant COCOMO II model drivers on a scale of 1 (least important) to 5 (most important). The option of labeling a model driver as "not applicable" was also provided. In addition to this, the participants were allowed to identify other drivers they considered important in determining the understandability of a software project. Furthermore, space was provided for open-ended comments.

3. RESULTS

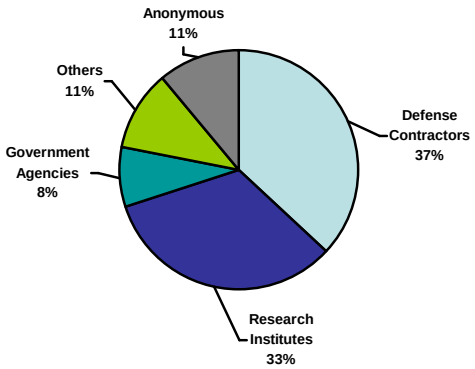

Figure 1. Participants' categories

A total of 36 people from 3 different continents participated in this survey. Figure 1 depicts the participants' categories. As is clear from this figure, the majority of the participants represented defense contractors and research institutes.

Table 2 summarizes the results of this survey in the form of weights for each of the eight model drivers. The model drivers in this table have been sorted in descending order of their weights. Each of these weights is an average of the responses of the participants. These weights represent the importance assigned by the participants *as a group* to each model driver in predicting UNDR.

Table 2. Model driver weights

S#	Model Driver	Weight
1	PREC	4.17
2	CPLX	3.86
3	APEX	3.53
4	ACAP	3.33
5	RESL	3.14
6	PLEX	3.08
7	LTEX	2.72
8	PCAP	2.64

According to these weights, the scale factor PREC (product precedentedness) seems to be the most important driver in determining UNDR. It is followed by CPLX (product complexity). PCAP (programmer capability) appears to be the least relevant for this group. These weights suggest that the driver with the highest weight (PREC) is almost 60% more effective in predicting UNDR vis-à-vis the driver with the lowest weight (PCAP).

PREC, CPLX, APEX, ACAP, and RESL were reported "not applicable" by one participant each while PLEX, LTEX, and PCAP were reported "not applicable by two participants each. These results clearly indicate that the vast majority of the participants were in agreement with the importance of our set of eight drivers in determining UNDR.

Furthermore, only about 6% of the participants suggested additional drivers that could be used in determining UNDR. None of these suggested additional drivers, however, belonged to the COCOMO II set of model drivers. The vast majority seemed to agree with the completeness of our set of eight drivers.

4. COMPARISON OF SURVEYS

A similar survey [3] was conducted earlier on 22 graduate students in a software engineering class at USC. In this survey too the students were asked to rank the eight COCOMO II model drivers on a scale of 1 (most important) to 5 (least important). The only notable difference between these two surveys is the fact that the students were not provided the option to mark a model driver as "not applicable" in predicting UNDR. The results of this students' survey, however, suggest that this does not seem to make a significant difference since not even a single student assigned a weight lower than 2 in ranking the model drivers. In

other words, each and every student thought that these eight model drivers had some significance in predicting UNDR.

Figure 2 depicts the results of these two surveys. In order to facilitate comparison the weights from these two surveys are presented in descending order of importance.

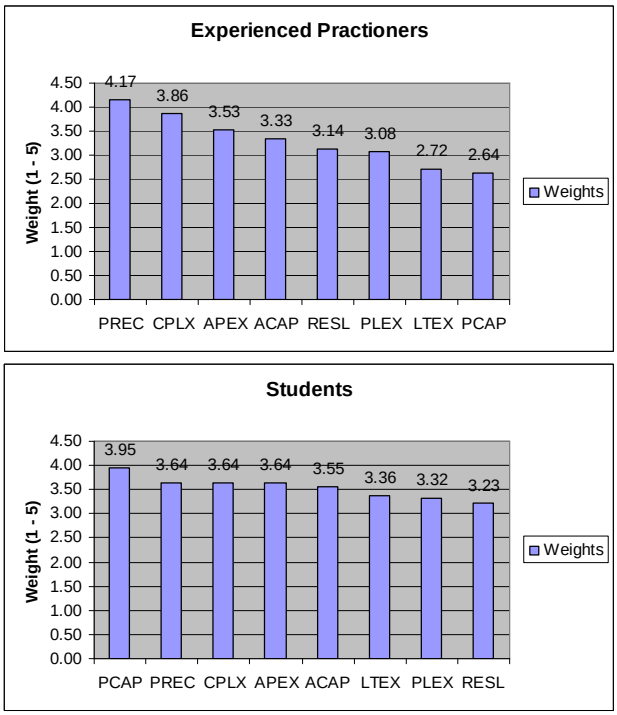

Figure 2. Comparison of weights

One difference that clearly stands out is the importance given to PCAP (programmer capability). The group of students seems to value it the most in predicting UNDR while the group of experienced practitioners values it the least. The scale factor RESL (architecture/risk resolution) seems to be the least important driver for students whereas the working professionals rank it somewhere in the middle. Another important difference between these two categories of individuals is in the range of weights. For the group of students, the driver with the highest weight (PCAP) is only about 20% more important in predicting UNDR than the driver with the lowest weight (RESL). For the group of experienced practitioners this range is almost three times bigger.

The coefficient of determination (R^2) between these two sets of weights is 0.013 which clearly indicates that there is no simple relationship between the two. A careful look at Figure 2, however, reveals a couple of subtle similarities between the weights assigned by these two categories of individuals. Firstly, even though the ranking order of weights for PLEX (platform experience) and LTEX (language and tool experience) is switched, both categories seem to consider these two drivers to be relatively less important in determining UNDR. Secondly, the relative importance of four drivers – PREC (product precedentedness), CPLX (product complexity), APEX (applications experience), and ACAP (analyst capability) – appears to be almost the same for both categories.

5. DISCUSSION

The differences in relative importance of COCOMO II model drivers across these two categories of individuals can be accounted for by taking into consideration the differences in the level of work experience of individuals in these categories. Most students who took this survey are novices with at most a couple of years of experience. Most experienced practitioners, on the other hand, had decades of industry experience under their belt.

Another explanation for these differences could come from the fact that these two categories of individuals are involved in projects of different sizes, resources, and duration. Software engineering academic projects at USC are typically less than ten thousand logical SLOC, are done in teams of four to eight students, and are a couple of semesters long. Industry projects, on the other hand, are often over a million logical SLOC in size, involve hundreds of personnel, and take years to complete.

These differences in work experience and typical characteristics of projects undertaken by members of these two categories seem to provide a plausible explanation for the results we have found. For typical academic projects programmer capability (PCAP) is much more important than analyst capability (ACAP) while the reverse is true for industry projects. By the same token, architecture/risk resolution (RESL) is significantly more important for typical industry projects as compared to academic projects.

A couple of caveats are in order. Firstly, the weights from the experienced practitioners should be taken with a grain of salt. Since the majority of the experienced practitioners who took this survey were either software engineering researchers or employees of defense contractors these weights may be biased towards their particular preferences. The weights given in Table 2, therefore, should not be taken to represent the collective opinion of all categories of experienced practitioners weighed equally.

Another threat to the validity of our results comes from the simple observation that these two categories of individuals possess different skill sets and experience in using software cost estimation tools. The experienced practitioners are much more advanced in their usage of such tools and therefore possess a much thorough understanding of the model drivers. Students, on the other hand, have a limited understanding of these drivers based solely on their past academic work.

6. CONCLUSIONS AND FUTURE WORK

The results of this empirical study indicate that our set of eight COCOMO II model drivers seems to be a useful resource in predicting the understandability of a software development project. The relative importance of these drivers, however, varies from one class of individuals to another as shown by the comparison of the results of our two surveys. A careful analysis shows that despite the prevalent differences there are some similarities between the weights assigned by different categories of individuals. As shown above, differences in work experience and project characteristics appear to provide plausible explanations for the differences in the importance assigned to these drivers by these two categories.

We were able to estimate the predictive power of the model driver weights given by students using a collection of academic projects. The same exercise needs to be done for the weights provided by working professionals. For this we plan to gather corresponding data from industry-scale projects.

Also, we plan to explore different variants of our definition of understandability. Currently, our definition encompasses the perspective of the development team only. The perspective of other success critical stakeholders such as users, customers, and maintainers, etc. also needs to be included in order to complete the picture. Another aspect worth exploring is the change in the relative importance of model driver weights in response to the development phase under consideration. The existing definition of understandability looks at the time period corresponding to the end of the Inception phase. Determining the weights for other phases of development e.g. Elaboration, Construction, and Transition, etc. would provide further insights into the quantitative aspect of understandability.

7. ACKNOWLEDGMENTS

We would like to thank all the students and the experienced practitioners who participated in our surveys.

8. REFERENCES

[1] Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., and Steece, B. 2000 Software Cost Estimation with COCOMO II. Prentice Hall.

[2] Boers, S. (Ravenflow CEO) 2008 Ravenflow Launches World's First Requirements Definition Add-in for Microsoft Word. News. http://www.ravenflow.com/news/pr_070708.php

[3] Malik, A., Boehm, B., and Brown, A. W. 2008. Predicting Understandability of a Software Project Using COCOMO II Model Drivers. 23rd International Forum on COCOMO and Systems/Software Cost Modeling and ICM Workshop 3 (Los Angeles, USA, October 27 – 30, 2008).

[4] Martin, J. 1984 An Information Systems Manifesto. Prentice Hall.

[5] Projects archive, http://ebase.usc.edu/Projects/index.jsp

[6] Standish Group. 1995 CHAOS. Standish Research Paper. http://www.standishgroup.com

[7] Survey questionnaire, http://csse.usc.edu/csse/event/2008/cocomoicm08/presentationByDay/Tues_Oct28/1c_COCOMOForumUNDRSurvery.doc