Risk Analysis in Software Development

CRAIG COMSTOCK
University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD UK

ZHIZHONG JIANG
Manchester Business School
Booth Street West, Manchester, M15 6PB UK

PETER NAUDÉ
Manchester Business School
Booth Street West, Manchester, M15 6PB UK

Abstract: Software is the key factor influencing the success of computer-based systems. However, software is expensive to develop, and there are great potential risks to deliver software without any overrun of the cost. In this study we investigated the amount of risk in terms of the range of possible productivity that can be expected in a portfolio of software development projects. Working on a large database with 4106 software projects, this study revealed that inefficient development teams can spend as many as 13 times of the effort taken by proficient teams for the development. This study further examined the effects of project size and team size on the variance in software development effort. The results showed that to reduce the risks of cost overruns, a small project is better than a large project when the team size is chosen in advance, and small team is preferable to a large team for a project of a fixed size.

Key-Words: Risk, Cost, Project size, Team size, Software engineering

1. Introduction

In recent years dramatic improvements in hardware performance and vast increases in memory and storage capacity have led to more sophisticated computer-based systems. Software has become the key factor influencing the success of computer-based systems [1]. However, software is expensive to develop. While hardware costs have decreased considerably, comprising less than one fifth of total expenditure, the cost of software development remains consistently high [2]. Producing software in a cost-effective and optimally efficient is one of the ever-present problems facing software engineering managers.

It is useful to regard software development as an economic production process [3]. Faced with increasingly high development costs, software developing firms tend to look for ways to decrease their development costs. Clearly, accurate estimate of software development effort is essential since the client and project management must agree on the boundaries of cost, time and quality. A low estimate may either cause loss or compromise the quality of the software developed, resulting in partially functional or insufficiently tested software that requires later high maintenance costs. On the other hand, overestimates can result in noncompetitive contract bids as well as over allocation of development resources and personnel. Unfortunately, in the real world two-thirds of all major software projects substantially overran their estimates [4].

This paper worked on the latest release of the ISBSG data repository. We first developed a model to estimate software development effort which is the main cost of software development. We further
showed that incompetent development teams can spend as many as 13 times effort (in terms of man-hours) taken by proficient teams for the development. This study further found that a small project or small team size has the advantage of reducing the risk of cost overruns given a particular development team or a planned project.

This paper is organized in six sections. Section 2 introduces the ISBSG database and the key software metrics. Section 3 develops a model for the estimation of development effort. Section 4 shows the variations in software development by comparing the low-end effort estimate with the high-end effort estimate. Section 5 examines the effects of project size and team size on development risk. Section 6 is the conclusion of this study.

2. Data Description
The common difficulty in the study of software metrics is the lack of accessible and reliable large dataset. Many contemporary metrics repositories have limited use due to their obsolescence and ambiguity of documentation. On the other hand, the data collected by individual researcher usually have a small sample size, insufficient to develop robust results.

2.1 Sample
Our research used the data repository maintained by the ISBSG (International Software Benchmarking Standards Group) which has been widely researched. This study worked on Release 10, the latest publication of ISBSG data repository. The database contains information on 4106 projects among which two thirds were developed between the years 2000 and 2007. The data kept on each project includes 107 metrics or descriptive pieces of information, including project size, number of developers, organization type, programming language, man-hours worked on the project by phase, and major defects that made it to production.

2.2 Key Software Metrics
A large part of software cost comes from development effort. Software development effort is the time taken to complete a software project. It is usually defined as the total man-hours or man-months taken to complete the project. In the ISBSG data repository, the metric Normalized Effort records the total effort in hours spent on the project. For projects covering the full development life-cycle, the value for Normalized Work Effort is the same as the summary of total effort in hours. For projects covering less than a full development life-cycle, this value is an estimate of the full life-cycle effort for all reported teams. Our study is based on statistical regression analysis, which is the most widely used approach for the estimation of software development effort.

We will now discuss the variables in the data repository that will be used as the predictors for development effort:

2.2.1 Adjusted Function Points (AFP)
The “Function Point” is a measure of project size that has been widely used to overcome the difficulties of traditional measure of lines-of-code in project planning and control [5]. In the ISBSG data repository, the metric Adjusted Function Points gives the size of the project as measured in functional points. Since the projects in the ISBSG depository applied different counting approaches used for various projects (e.g., IFPUG, NESMA, Mark II), the functional size was adjusted by the adjustment factor which gives the Adjusted Function Points.

Project size is a major estimator in nearly all effort estimation models (e.g., COCOMO [6], ESTIMACS [7]). As project size is such an important estimator of development effort that most effort estimation models consist of two phases [4, 8]. In the first phase, an estimate of the software size is made; and in the second, the effort of the project is predicted based on the estimated software size.

2.2.2 Average Team Size
It is the average number of people that worked on the project through the entire development process. Researchers have examined the influential effect of team size on software development effort (e.g., [9, 10].
2.2.3 Language Type
This metric specifies the development language used for the project, including second-generation languages (2GL), third-generation languages (3GL), fourth-generation languages (4GL) and Application Generator (ApG). In practice, all fourth-generation languages are designed to reduce programming efforts, and they are more productive than third-generation languages [11].

2.2.4 Development Platform
This metric defines the primary target computer platform for the development, which is classed as main frame, mid range, multi platform, and PC. In most application software development, the target machine (e.g. mainframe, PC) often determines the platform characteristics in which programming needs to be accomplished [12]. Computer platform has been considered an important cost driver in estimating software effort (e.g., [13, 14]).

2.2.5 Development Type
This factor indicates whether the software development was a new development, enhancement or re-development of the existing software. While new development starts everything from scratch, software enhancement simply adds, changes, or deletes software functionality of legacy systems to adapt to new and evolving business requirements [15]. Development type is a potential factor affecting development effort.

2.2.6 CASE Tool Used
This field specifies whether the project used any CASE (Computer-Aided Software Engineering) tool. The promise of CASE tool is that it can increase development productivity and reduce development costs. However, Bruckhaus et al. [16] pointed out that the introduction of CASE tool does not necessarily improve productivity, and in certain situations it can actually decrease productivity as it increases effort on specific activities. Hence it is necessary to reconsider the undetermined effect of CASE tool.

2.2.7 How Methodology Acquired
This describes how the development methodology was acquired. This could be traditional, purchased, developed in-house, or a combination of purchased and developed. Very little research has been done to examine this factor. Some preliminary data analysis found development methodology insignificant to software development effort [14].

3. Statistical analysis
After the cleaning of the data, we then examined the data for the objective of estimating software development effort based on the key software metrics. The results indicated that we can use multiple regression analysis to estimate effort. For the variables introduced in the previous section, Normalized Work Effort, Adjusted Function Points, and Average Team Size were the only three variables that were measured in ratio scale, and all the others were measured in nominal scale. Since the data vary significantly, log-transformations were taken for the three variable measured in ratio scales. After the transformations, the scatterplot of Normalized Work Effort against Adjusted Function Points and the scatterplot of Normalized Work Effort against Average Team Size were plotted. The scatterplots show that a linear model can be applied to approximate their relationships (see Fig. 1).

Fig.1 Scatterplot of log(Effort) against log(AFP) and log(TeamSize).
The rule of thumb suggests a minimum sample size of $50+8k$ ($k$ is the number of predictors) [17]. Our valid sample size after data cleaning is 574 which is sufficient to perform regression analysis. We further examined if the problem of multicollinearity (strong correlations between predictor variables) exists in the data. The correlation tests indicated that there is no problem of multicollinearity. Hence we can apply regression analysis to assess the variables.

We first used automatic model selection based on Akaike’s information criterion (AIC). AIC is a measure of how good the fit is of an estimated statistical model [18]. In our study AIC measure only excluded CASE tool as the insignificant variable for effort. As regression based on AIC tends to overestimate the number of parameters when the sample size is large, it is suggested that the use of AIC should be combined with other statistical methods (e.g. ANOVA) to further assess the significance of the predictors. Nevertheless, regression based on AIC gives some guide on the significance levels of the remained variables. The variable How Methodology Acquired was thus further excluded. Table 1 below gives the summary of the regression results.

Table 1 Summary of the Regression Results

<table>
<thead>
<tr>
<th>Regression Terms</th>
<th>Coefficients</th>
<th>Standard Error</th>
<th>$p$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>4.380</td>
<td>0.281</td>
<td>$&lt; 10^{-15}$</td>
</tr>
<tr>
<td>log(AFP)</td>
<td>0.539</td>
<td>0.030</td>
<td>$&lt; 10^{-15}$</td>
</tr>
<tr>
<td>log(TeamSize)</td>
<td>0.672</td>
<td>0.037</td>
<td>$&lt; 10^{-15}$</td>
</tr>
<tr>
<td>Language3GL</td>
<td>-0.345</td>
<td>0.254</td>
<td>0.175</td>
</tr>
<tr>
<td>Language4GL</td>
<td>-0.770</td>
<td>0.258</td>
<td>0.003</td>
</tr>
<tr>
<td>LanguageApG</td>
<td>-0.761</td>
<td>0.276</td>
<td>0.006</td>
</tr>
<tr>
<td>PlatformMR</td>
<td>-0.140</td>
<td>0.074</td>
<td>0.059</td>
</tr>
<tr>
<td>PlatformMulti</td>
<td>-0.120</td>
<td>0.165</td>
<td>0.466</td>
</tr>
<tr>
<td>PlatformPC</td>
<td>-0.363</td>
<td>0.080</td>
<td>$6.3 \times 10^{-4}$</td>
</tr>
<tr>
<td>DevTypeNew</td>
<td>0.258</td>
<td>0.064</td>
<td>$5.6 \times 10^{-4}$</td>
</tr>
<tr>
<td>DevTypeRe</td>
<td>0.533</td>
<td>0.145</td>
<td>$2.6 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

NB: the default language type is 2GL, default platform is Mainframe, and the default development type is Enhancement. Multiple $R^2$-squared = 0.735. Residual standard error is 0.663 on 566 degrees of freedom.

According to Table IV, the model is fitted as:

$$
\log(\text{Effort}) = 4.380 + 0.539 \times \log(\text{AFP}) + 0.672 \times \log(\text{TeamSize}) + \alpha_i \Phi(\text{Language}_i) + \beta_j \Phi(\text{Platform}_j) + \gamma_k \Phi(\text{DevType}_k) \\
i = 1, 2, 3, 4; \ j = 1, 2, 3, 4; \ k = 1, 2, 3
$$

Here log() is the natural log operation with base $e$ ($e = 2.718$). The function $\Phi$ is an indicator function with binary values of 1 or 0 (value of 1 indicates the relevant development technique in the parentheses is used, otherwise the value is 0). The default development techniques used are: 2GL for development language ($\alpha_1=0$), Mainframe for development platform ($\beta_1=0$), and Enhancement for development type ($\gamma_1=0$). The coefficients $\alpha_i$, $\beta_j$, and $\gamma_k$ can be obtained from Table IV. The fitted model can be used to estimate the work effort required for the development. For instance, suppose one particular project is a new development for mainframe platform, with functional size 1000 and average team size 10, using fourth-generation language. Then the effort can be estimated as:

$$
\log(\text{Effort}) = 4.380 + 0.539 \times \log(1000) + 0.672 \times \log(10) - 0.770 + 0 + 0.258 = 9.14
$$

$$
\text{Effort} = 9320
$$

Hence a total of 9320 man-hours are estimated for the development.

4. Risks in Software Development

The explanatory power of the fitted model is quite high at $R^2 = 73.5\%$. The standard error of the residuals is 0.663 on 566 degrees of freedom.

In a true linear model, suppose that all the independent variables $x$’s are fixed, but the dependent variable is random and generated by the model:

$$
y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon
$$

Where $\epsilon$ follows a normal distribution $\epsilon \sim N(0, \sigma^2)$. The variance $\sigma^2$ can be estimated with $RSS$ (residual sum of squares): $\sigma^2 = RSS/(n-p)$, where $n$ is the valid
sample size and \( p \) is the number of parameters to be estimated. Therefore, in our case \( \sigma^2 \) equals to the square of standard error, or \( \sigma = 0.663 \). We can use \( \sigma \) to estimate the confidence interval for \( \log(\text{Effort}) \) based on the true linear model when all the independent variable are fixed and there are fluctuations in \( y \) due to the random effect of \( \varepsilon \). Suppose \( \log(\text{Effort}) \) is estimated to be \( u_0 \), the 95% confidence interval for \( \log(\text{Effort}) \) can be written as: 

\[
u_0 \pm 1.96 \times \sigma\]

Based on the 95% confidence interval the maximum effort possible for the development is \( \exp(u_0+1.96*\sigma) \) and the minimum possible effort is \( \exp(u_0-1.96*\sigma) \), compared with the base effort as \( \exp(u_0) \). The adjustment factor is \( \exp(1.96*\sigma) = 3.67 \). Comparing the maximum effort with the minimum effort, we get 

\[
\frac{\exp(u_0+1.96*\sigma)}{\exp(u_0-1.96*\sigma)} = \exp(2 \times 1.96 \times 0.663) = 13.4
\]

Therefore, we can conclude the incompetent development teams can spend as many as 13 times of the effort taken by proficient teams for the development. This demonstrates that there are significant risks in predicting and managing software development.

### 5. The Effect of Project Size and Team Size on Development Risk

Given this high level of risk measured by a wide confidence interval, we further investigated to determine which factors will increase or decrease risk.

Table 1 showed that average team size, project size and language type are the three most significant factors for development effort. Accordingly, we divided the dataset into 32 groups by grading each record according to three criteria. For average team size, we divide the data into four groups: T1 (average team size < 3), T2 (3 \( \leq \) average team size \( \leq \) 5), T3 (6 \( \leq \) average team size \( \leq \) 10), and T4 (average team size > 10). For project size, we also divided the data into four groups: A1 (project size \( \leq \) 100 afp), A2 (100 < project size \( \leq \) 200 afp), A3 (200 < project size < 400 afp), and A4 (project size \( \geq \) 400). For Language Type, we kept the two main language groups: 3GL and 4GL. For each of the 32 groups, we calculated the standard deviation of Effort in the group members. We got 32 standard deviations which will be used to assess the factors that contribute to its large values. We used regression analysis which can control the effect of other factors when we examine the effect of one specific factor. Table 2 below shows the regression results. Note that Language Type is not significant and thus removed.

When we interpret the effect of project size (measured with Adjusted Function Points) on the risk, the effect of average team size needs to be fixed in advance. The linear regression we used can inspect the effect of project size when the effect of team size is controlled. In Table 2, the default group for project size is A1 and the default group for average team size is T1. We can see that the standard deviation increases along with project size for a given team size. Therefore, in cases when the development team is chosen beforehand, it is preferable to choose a smaller project for the purpose of reducing development risk. Likewise, when the project size is given in advance, the project with larger team size has larger standard deviation thus more risk.

| Table 2 Regression Results of Standard Deviation on AFP and Average Team Size |
|-------------------------------|----------------|---------------|------------------|
| Value | T Value | Pr(>|t|) |
| (Intercept) | -976.156 | -0.898 | 0.378 |
| AFPA2 | 515.244 | 0.443 | 0.661 |
| AFPA3 | 1212.956 | 1.044 | 0.307 |
| AFPA4 | 5645.496 | 4.859 | 0 |
| TeamSizeT2 | 1018.398 | 0.876 | 0.389 |
| TeamSizeT3 | 1853.136 | 1.595 | 0.123 |
| TeamSizeT4 | 5186.159 | 4.463 | 0 |
| Multiple R-Squared | 0.676 |

The default group for AFP is A1 and the default group for Team Size is T1.

### 6. Conclusion

In this study we investigated the amount of risk in terms of the range of possible productivity that can be expected in a portfolio of software development...
projects. Working on the large database with 4106 projects from across the world, this study revealed that inefficient development teams can spend as many as 13 times of the effort taken by proficient teams for the development. This study further examined the effects of project size and team size on the variances in software development. The results showed that to reduce the risks of cost overruns, a small project is better than a large project when the team size is chosen in advance, and a small team is preferable to a large team for a project of a certain fixed size.

References