Knowledge Management of Software Productivity and Development Time

James A. Rodger, Pankaj Pankaj, Ata Nahouraii

MIS and Decision Sciences, Eberly College of Business & Information Technology, Indiana University of Pennsylvania, Indiana, USA.
Email: jrodger@iup.edu

Received April 15th, 2011; revised September 20th, 2011; accepted September 30th, 2011.

ABSTRACT

In this paper, we identify a set of factors that may be used to forecast software productivity and software development time. Software productivity was measured in function points per person hours, and software development time was measured in number of elapsed days. Using field data on over 130 field software projects from various industries, we empirically test the impact of team size, integrated computer aided software engineering (ICASE) tools, software development type, software development platform, and programming language type on the software development productivity and development time. Our results indicate that team size, software development type, software development platform, and programming language type significantly impact software development productivity. However, only team size significantly impacts software development time. Our results indicate that effective management of software development teams, and using different management strategies for different software development type environments may improve software development productivity.

Keywords: Software Engineering, Software Metrics, Fourth Generation Languages, CASE Tools, Field Study, Tools and Techniques, Models And Principles, Team Size, Volatility, Development Platform and Type, Productivity, Elapsed Development Time

1. Introduction

Competition in software industry has increased significantly. One of the ways software companies can stay competitive is to improve software development productivity of their software products. However, despite the advances in software development tools, development methodology and programming languages, research shows that productivity improvements have either remained the same or declined substantially [1].

Several studies in the literature have measured factors impacting either software productivity or software development time [2,3]. Blackburn et al. [4] argue that software productivity and software development time are not the same. For example, low productivity organizations can reduce the software development time by increasing the software development team size. While increasing the number of software developers to reduce software development time is an interesting option, Fried [5] argues that large teams increase the non-productive time due to increased communication and coordination requirements.

Very few researchers have focused on developing models to understand the primary antecedents of software development productivity and software development time. In fact, we are not aware of any study that uses real-world data and investigates the impact of certain variables on both software development productivity and software development effort. For example, the Blackburn et al. [4] study uses survey data and measures managerial perceptions.

Management of both software development productivity and software development time are of paramount importance. Effective management of software development productivity and software development time leads to a better competitive position for an organization [6]. In certain cases, managing software development time may lead to a lower likelihood of schedule overrun and litigation due to violations of contractual agreements.

In this paper we investigate the impact of team size, ICASE tools, software development platform, software development type, and type of programming language on software development productivity and software development time. We use a real-world data set of 130 different software development projects. The projects in the data set were completed between years 1989-2001 in over...
seven different countries. The data are used in many other studies and is publicly available from the International Software Benchmarking Standards Group [7-9].

The rest of the article is organized as follows. First, using the software engineering literature, we identify the factors that may impact the software development productivity and software development time. Second, we describe our data and empirically test the impact of the identified factors on software productivity and software development time. In the end, we provide a summary, limitations and future extensions of the research.

1.1. Relevant Literature and Hypotheses

Very few researchers have focused on developing models to understand the primary antecedents of software development productivity and software development time. Subramanian and Zarnich [10] proposed a theoretical model that causally predicts the software productivity. The Subramanian and Zarnich [10] model consists of three independent variables: ICASE tools, systems development method and ICASE tool experience. Using real-world data on several software projects from an organization, Subramanian and Zarnich [10] empirically validated their model. Foss [11] proposed four essential aspects for reducing software development time tools, methodology, people and effective management. Given that tools, methodology and people impact both software productivity and development time, we investigated the impact of these factors on software productivity and development time.

1.1.1. Tools and Methodology

Programming methods and tools are known to have an impact on the software development effort. Programming methods consist of the programming language, the development platform and the development methodology [10, 12,13].

Programming, project management and design tools—hereafter called development tools—do have an impact on software productivity and development time. Development tools have been used to improve analyst and programmer productivity, improve software quality, reduce maintenance, and increase management control over the software development process. Automated software development tools fall into three categories: programming support tools, design technique tools and project management tools [14]. There is qualitative data available that supports the development tool type as having an impact on the software effort and productivity [15]. Other researchers have supported these claims [16-18].

Programming languages are the primary methods for creating software. The basic challenge for business software builders is to build reliable software as quickly as possible. Fourth generation languages automate much of the work normally associated with developing software applications [19]. The literature on the impact of language type on software productivity is inconclusive. Blackburn et al. [4] reported that language type does have an impact on software development productivity. However, Blackburn et al. [2] reported that language type does not have an impact on productivity. One of the reasons why programming languages might not have an impact on effort is that some of the programming languages, such as C++, might be more complex than some of the other 3GLs. 4GLs and recent object-oriented programming languages, while complex, provide many functionalities that might lead to lower effort. For example, Microsoft Foundation Classes (MFC) in Visual C++ and JAVA Swing classes in Java programming provide several reusable classes that might be used to design graphical user interfaces efficiently. 3GL languages don’t provide such extensive capabilities; some of the complex visual interfaces are only possible in 4GL languages.

This leads to the following hypothesis:

Hypothesis 1: The use of 4GL programming language will increase software development productivity and reduce software development time.

Integrated CASE (ICASE) tools are designed to provide support for all phases of the systems development life cycle [10]. The capabilities of ICASE tools include the following:

1) Graphical capabilities for modeling user requirements, and error and consistency checking.
2) Prototyping and system simulation capabilities.
3) Code generating capabilities.
4) Code testing, code validation, and code reuse capabilities.
5) Reengineering, reverse engineering, data dictionary and database interface capabilities.
6) Management information acquisition, storing, managing and reporting capabilities.

Banker and Kauffmann [20] showed that the use of ICASE tools has a significant impact on productivity. Subramanian and Zarnich [10], confirming the positive impact of ICASE tools on productivity, showed that no significant differences in productivity are observant for different types of ICASE tools. Subramanian and Zarnich [10] mentioned that programmer experience with ICASE tools is an important factor in improving productivity. Vessey et al. [21] argued that the use of ICASE tools alone cannot warrant productivity improvements, and programmers trained in the use of ICASE tools are crucial for productivity improvements. Blackburn et al. [2] speculating on the impact of CASE tools, mentioned that increasing project complexity and size are obscuring the advantages that CASE tools bring. We propose following hypothesis:
Hypothesis 2: The use of ICASE tools will increase software development productivity and lower software development time.

1.1.2. Team Size
Team size, as a factor impacting software effort and productivity, has been used in several studies [3,7,22-25]. While team size seems to play a role, its impact is not clearly established. In a global survey of different countries, Blackburn et al. [2] argued that smaller teams might be more productive. However, the authors said that the assertions about small team size and productivity are rarely supported by anecdotal evidence. Microsoft used a strategy of employing small teams of star developers and found that the strategy, when confronted with the market realities of marketing, developing, and maintaining large mass-market applications, does not work well [26]. Large team size might inhibit productivity due to inefficiencies created by the problems of coordination and communication between the members of the team [27,28]. However, larger team size during the customers’ requirements phase might avoid ambiguity, which might improve productivity. Banker and Kemerer [29] argued that software projects might benefit from larger team size as specialized personnel with expertise in certain areas might improve overall productivity.

Smith et al. [12], in their empirical study on the impact of team size on software effort, using an object-oriented programming language-based system, showed that team size does not have a significant impact on software effort. However, Angelis et al. [7], in multi-organizational and multi-project data, claimed that team size does have an effect on software development effort. Since our data is similar to Angelis et al. [7] data, we have the following hypothesis:

Hypothesis 3: An increase in team size will decrease software development productivity and increase software development time.

1.1.3. Computer Platform
Computer platform, as a factor impacting software development time and productivity, has been used in several studies [30,31]. The computer platform refers to the both the machine complex and infrastructure software and is a function of execution time constraints, main storage constraints and platform volatility [30]. The platform characteristics in which application software development programming needs to be accomplished is determined by a target machine such as a mainframe, minicomputer, or personal computer [32]. Platform difficulty (factors) is rated from very low to very high and can be used to determine software development productivity and elapsed time [30].

In the modern client-server architecture, personal computers are used as clients and small or mid-range computers are used as servers [33]. Mainframe computers continue to be used for centralized data management functions midrange computers have become popular in distributed data processing [34]. While the older legacy systems are run on mainframes, the newer systems running on the personal computer or midrange platforms function to interact with the legacy systems. Based on the foregoing discussion, we propose following hypothesis:

Hypothesis 4: An increase in computer platform complexity will increase software development productivity and lower software development time.

1.1.4. Software Development Type
It is well documented that the costs of enhancing software applications to accommodate new and evolving user requirements is significant [35]. Software development can fall into three major types. These categories include new, redevelopment and enhancement software types. According to ISBSG standards, new development types mean that a full analysis of the application area is performed, followed by the complete development life cycle, (planning/feasibility, analysis, design, construction and implementation). An example of a new development type may be a project that delivers new function to the business or client. The project addresses an area of business, (or provides a new utility), which has not been addressed before or provides total replacement of an existing system with inclusion of new functionality. In the re-development of an existing application, the functional requirements of the application are known and will require minimum or no have no changes. Re-development may involve a change to either the hardware or software platform. Automated tools may be used to generate the application. This includes a project to re-structure or re-engineer an application to improve efficiency on the same hardware or software platform. For re-development, normally only technical analysis is required. Enhancement changes are development types made to an existing application where new functionality has been added, or existing functionality has been changed or deleted. This would include adding a module to an existing application, irrespective of whether any of the existing functionality is changed or deleted. Enhancements do not have errors but require significant costs for system upgrades [36]. Adding, changing and deleting software functionality to adapt to new and evolving business requirements is the foundation of software enhancements [35]. Software volatility is a factor that drives enhancement costs and errors [37,38]. Further, there is an opportunity to introduce a new series of errors every time an application is modified [39]. We propose following hypothesis:
Hypothesis 5: An increase in software volatility will decrease software development productivity and increase software development time.

Manova is used to test the resulting model seen in Figure 1.

2. Data and Experiments

We obtained the data on 1238 software projects from International Software Benchmarking Standards Group (ISBSG). The ISBSG (release 7) data are used by several companies for benchmarking software projects and are available in the public domain. The ISBSG procedures encourage software development teams to submit their project data to the repository in return for a free report, which graphically benchmarks their projects against similarly profiled projects in the ISBSG repository [7]. The software project data typically are submitted by the software project manager, who completes a series of special ISBSG data validation forms to report the confidence he/she has in the information he/she provides. ISBSG has developed a special mutually exclusive data quality rating that reflects the quality of data related to any given project. Each project is assigned a data quality rating of A, B, and C to denote the following:

- A = The project data satisfies all the criteria for seemingly sound data.
- B = The project data appears fundamentally sound, but some data attributes might not be fundamentally sound.
- C = The project data has some fundamental shortcomings.

Companies participating in ISBSG benchmarking acquired project data in several different ways. FP data on the projects were acquired mostly by an automated process (about 40%) or obtained from the development tools (about 21%). The software effort data were mostly recorded (about 59%). In certain cases, the software effort data were derived from the actual project cost (about 13%). In many cases, the data acquisition procedures were missing or unknown.

The software projects in ISBSG release 7 data came from 20 different countries. Figure 2 illustrates the major data-contributing countries. The top three known contributing countries were the United States, Australia, and Canada. Over 97% of the projects were completed between the years 1989-2001. Most of the projects (about 50%) were completed between the years 1999-2001. Figure 3 illustrates the data quality rating of the ISBSG release 7 data on the 1238 projects, and Figure 4 illustrates the industry type distribution for the 1238 projects.

The ISBSG data set included data on integrated CASE tools, programming languages, development type, development platform, elapsed time, productivity and team size. Of the total 1238 software projects, only 138 projects had complete data on all five independent and dependent variables for investigating elapsed time and productivity. For the elapsed time and productivity model, we used all 138 projects, respectively, in our analysis.

![Figure 2. ISBSG release 7 project data origin by country.](image1)

![Figure 3. Data quality distribution of the ISBSG release 7 project data.](image2)
Figure 4. Industry type distribution in the ISBSG release 7 project data.

Figure 5 illustrates the distribution of projects by different programming languages and Figure 6 illustrates CASE tool types. The majority of the projects used a fourth generation programming language. ICASE tools were used in only about 7.9% of the elapsed time model and 7.8% for productivity model. All other projects used upper CASE tools, no CASE tools or lower CASE tools.

Figure 7 illustrates project distribution by industry type and Figure 8 illustrates data quality for the 138 projects in the elapsed time and productivity model. The majority of projects, about 22.2%, were from the banking industry for the elapsed time and productivity model. When comparing the industry distribution and data quality for the 138 projects with the original set of 1238 projects, we see that the data quality distribution for the 138 projects is very similar to the data quality distribution for the original 1238 projects. For the elapsed time and productivity model, 65.9% was A quality and 34.1% was B quality.

Figure 9 illustrates project distribution by platform type for the elapsed time and productivity models. Figure 10 illustrates development type for the 138 elapsed time and productivity projects respectively. The majority of platforms, about 47.62%, were mainframes for the elapsed time and productivity model. The majority of development types, about 69%, were new development for the elapsed time and productivity model.
We used the Multiple Analysis of Variance (MANOVA) procedure to test all hypotheses. Table 1 illustrates the results of the multivariate tests for the five independent and the two dependent variables, elapsed time and productivity. The Pillar’s Trace, Wilk’s Lambda, Hotelling’s Trace and Roy’s Largest Root were significant at the 0.05 level of significance for development type.

Pillar’s Trace, Wilk’s Lambda, Hotelling’s Trace and Roy’s Largest Root were significant at the 0.000 level of significance for development platform and team size and language generation. Pillar’s Trace, Wilk’s Lambda, Hotelling’s Trace and Roy’s Largest Root were not significant for I-CASE.

Table 2 illustrates the tests of between-subjects effects.
Table 2. Tests of between-subjects effects.

<table>
<thead>
<tr>
<th>Source</th>
<th>Dependent Variable</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrected Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project Elapsed Time</td>
<td>1770.704a</td>
<td>27</td>
<td>65.582</td>
<td>1.945</td>
<td>0.009*</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>115.786b</td>
<td>27</td>
<td>4.288</td>
<td>8.359</td>
<td>0.000*</td>
</tr>
<tr>
<td>Intercept</td>
<td>Project Elapsed Time</td>
<td>1658.895</td>
<td>1</td>
<td>1658.895</td>
<td>49.204</td>
<td>0.000*</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>83.106</td>
<td>1</td>
<td>83.106</td>
<td>161.990</td>
<td>0.000*</td>
</tr>
<tr>
<td>Development Type</td>
<td>Project Elapsed Time</td>
<td>104.390</td>
<td>2</td>
<td>52.195</td>
<td>1.548</td>
<td>0.217</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>4.802</td>
<td>2</td>
<td>2.401</td>
<td>4.680</td>
<td>0.011**</td>
</tr>
<tr>
<td>Development Platform</td>
<td>Project Elapsed Time</td>
<td>89.171</td>
<td>2</td>
<td>44.585</td>
<td>1.322</td>
<td>0.271</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>26.252</td>
<td>2</td>
<td>13.126</td>
<td>25.585</td>
<td>0.000*</td>
</tr>
<tr>
<td>I-CASE tool</td>
<td>Project Elapsed Time</td>
<td>6.344</td>
<td>1</td>
<td>6.344</td>
<td>0.188</td>
<td>0.665</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>0.479</td>
<td>1</td>
<td>0.479</td>
<td>0.934</td>
<td>0.336</td>
</tr>
<tr>
<td>Generation</td>
<td>Project Elapsed Time</td>
<td>28.023</td>
<td>1</td>
<td>28.023</td>
<td>0.831</td>
<td>0.364</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>4.705</td>
<td>1</td>
<td>4.705</td>
<td>9.171</td>
<td>0.003*</td>
</tr>
<tr>
<td>Team Size</td>
<td>Project Elapsed Time</td>
<td>1295.088</td>
<td>21</td>
<td>61.671</td>
<td>1.829</td>
<td>0.024**</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>48.736</td>
<td>21</td>
<td>2.321</td>
<td>4.524</td>
<td>0.000*</td>
</tr>
<tr>
<td>Error</td>
<td>Project Elapsed Time</td>
<td>3708.616</td>
<td>110</td>
<td>33.715</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>56.433</td>
<td>110</td>
<td>0.513</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>Project Elapsed Time</td>
<td>15219.960</td>
<td>138</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>589.846</td>
<td>138</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>Project Elapsed Time</td>
<td>5479.320</td>
<td>137</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>172.219</td>
<td>137</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*significant at the .01 level; **significant at the .05 level; a R Squared = 0.323 (Adjusted R Squared = 0.157); b R Squared = 0.672 (Adjusted R Squared = 0.592).

in the elapsed time and productivity model. It also illustrates the results of the overall model fit. The results indicate that the overall model fit was satisfactory. The F-value was 8.359 for productivity and 1.945 for elapsed time. The model fit was significant at the 0.01 level of significance for both elapsed time and productivity. The R-square for elapsed time was 0.323. This indicates that the independent variables explain about 32% of the variance in the dependent variable. The R-square for productivity was 0.672. This indicates that the independent variables explain about 67% of the variance in the dependent variable.

The results provide support for hypothesis one. The coefficient for 4GL is significant for productivity (p = 0.003) and not significant for elapsed time (p = 0.364). This indicates that the use of 4GL programming languages do reduce software elapsed time and increase productivity. For hypothesis two, no significant impact of ICASE tools was found on the software elapsed development time or productivity. This indicates that use of ICASE tools do not have an impact on the software development elapsed time or productivity. Hypothesis three was supported at the 0.05 level of significance, for elapsed time, indicating that the increase in team size will lead to an increase in the software development elapsed time (p = 0.024). However, productivity was also supported at the 0.000 level of significance indicating that the increase in team size will lead to an increase in productivity (p = 0.000). These results may suggest a nonlinear relationship is at work here. Hypothesis four was supported. The coefficient for platform is significant for productivity (p = 0.000) and not significant for elapsed time (p = 0.271). This indicates that the platform used has an impact of reducing software elapsed time and increase productivity. Hypothesis five, which investigated development type volatility, was supported, indicating that enhanced development lead to increases in software development elapsed time. The coefficient for volatility is significant for productivity (p = 0.011) and not significant for elapsed time (p = 0.217). This indicates that the development type volatility has an impact of reducing software elapsed time and increase productivity.

In order to increase the confidence on the pair-wise comparisons for development type volatility and platform type, the Tukey method was utilized in the elapsed time and productivity model. Post hoc tests are not performed for generation because there are fewer than three groups. Post hoc tests are not performed for LNSIZE because at least one group has fewer than two cases. These results can be seen in Tables 3 and 4.
Table 3. Mean differences for development type volatility for productivity.

<table>
<thead>
<tr>
<th></th>
<th>Redevelopment</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancement</td>
<td>0.330 (p = 0.548)</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>0.960 (p = 0.026)**</td>
<td>0.270 (p = 0.127)</td>
</tr>
</tbody>
</table>

** The mean difference is significant at the 0.05 level.

Table 4. Mean difference for platform type for productivity.

<table>
<thead>
<tr>
<th></th>
<th>Main Frame</th>
<th>Personal Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Computer</td>
<td>1.304 (p = 0.000)*</td>
<td></td>
</tr>
<tr>
<td>Mid-Range</td>
<td>0.321 (p = 0.115)</td>
<td>0.983 (p = 0.000)*</td>
</tr>
</tbody>
</table>

* The mean difference is significant at the 0.01 level.

3. Discussion, Limitations and Conclusions

We have investigated the factors impacting the software elapsed time and productivity. Using the existing literature, we identified several variables that might impact software elapsed time and productivity. Further, using a data set of 138 projects for elapsed time and productivity, we empirically tested the impact of several factors on these dependent variables.

Tabachnick and Fidell [40], state that for multiple continuous dependent variables, multiple discrete independent variables and some continuous independent variables, a researcher should run Factorial MANCOVA. MANOVA works best with either highly negatively correlated dependent variables or moderately correlated dependent variables when correlation is less than 0.6. Since our correlation between productivity and elapsed time is a negative number, greater than 0.60, the use of MANOVA is more powerful than using two ANOVAs.

The results provide support for hypothesis one. The coefficient for 4GL is significant for productivity (p = 0.003) and not significant for elapsed time (p = 0.364). This indicates that the use of 4GL programming languages do reduce software elapsed time and increase productivity. For hypothesis two, no significant impact of ICASE tools was found on the software elapsed time and productivity. This indicates that the use of 4GL programming languages do reduce software elapsed time and increase productivity. Hypothesis three, which investigated development type volatility, was supported, indicating that enhanced development lead to increases in software development elapsed time. The coefficient for volatility is significant for productivity (p = 0.011) and not significant for elapsed time (p = 0.217). This indicates that the development type volatility has an impact of reducing software elapsed time and increase productivity.

ICASE tools have been known to have significant impact on productivity [10, 20]. In our case, the non-significant impact of ICASE tools on elapsed time and productivity could be because of several reasons. First, over 90% of our data set did not contain ICASE tools, and the limited number of ICASE tools might have jeopardized the statistical significance. Second, we did not have information on the programmers’ ICASE tool experience. Subramanian and Zarnich [10] indicated that ICASE tool experience is one of the contributing factors for lower productivity in ICASE tool projects. Kemerer [41], highlighting the importance of ICASE tool experience, wrote the following.

Integrated CASE tools have raised the stakes of the learning issue. Because these tools cover the entire life cycle, there is more to learn, and therefore the study of learning—and the learning-curve phenomenon—is becoming especially relevant.

Thus, we believe that lack of information about ICASE tool experience might have impacted the significance results between the ICASE tool and software project elapsed time and productivity effort hypotheses.

The type of programming language did not have an impact on software project elapsed time (p = 0.364), but did impact productivity (p = 0.003). The descriptive statistics of the data indicate that about 51% of the projects were developed in 4GL languages, and 49% of the projects were developed in 3GL programming languages. Thus, we believe that our data was not very biased for any particular generation of programming languages. The insignificance of programming language on software project elapsed time could be due to several reasons. The first reason might be that the programmers’ experience in programming language might play a role. A few languages are more difficult to learn than others. Second, the complexity of a language type might compensate for any other advantages that it might offer, such as code and design reuse. We observed very interesting results in regard to team size. First, an increase in team size generally
leads to higher software project elapsed time and decreased productivity. This increase in software project elapsed time might be due to increased communication requirements that in turn lead to decreased overall productivity.

The heterogeneity of software project elapsed time, development effort recording techniques, and data quality of projects improve the external validity of our study at the expense of internal validity of the study. Given that our data came from multiple projects and multiple organizations, heterogeneity was expected. We do, however, note that there may be certain limitations related to the internal validity of the study. There may be other factors that may limit the generalization of the results of our study. First, we had very few ICASE tools in our data set, which might have had an impact on both internal and external validity of hypotheses related to ICASE tools. Second, we did not have programmers’ experience information on ICASE tools and programming languages, which is known to have an impact on the software development effort. Third, the non-parametric dataset and parametric regression model might have provided us a lower fit, and the regression results might in fact be improved by using non-parametric models. Since our data is available in the public domain, we believe that future research may address some of these issues.

REFERENCES


