SOFTWARE DEVELOPMENT PRODUCTIVITY FACTORS IN PC PLATFORM

Abbas Heiat, College of Business, Montana State University-Billings, Billings, MT 59101, 406-657-1627, aheiat@msubillings.edu

ABSTRACT

CRT and ANN algorithms used for determining the most important variables (attributes) of software development in PC environment. While accuracy of classification of productive versus non-productive cases by CRT and ANN are relatively close (72% vs 69%), their ranking of important variables are different. CRT ranks the Programming Language as the most important variable and Function Points as the least important. On the other hand, ANN ranks the Function Points as the most important followed by team size and Programming Language.

INTRODUCTION

Identifying the most relevant factors influencing project performance is essential for implementing business strategies by selecting and adjusting proper improvement activities. There is, however, a large number of potential influencing factors. There is, however, a large number of potential influencing factors. This paper proposes a novel approach for identifying the most relevant factors influencing soft- ware development productivity. The method elicits relevant factors by integrating data analysis and expert judgment approaches by means of a multi-criteria decision support technique. Empirical evaluation of the method in an industrial context has indicated that it delivers a different set of factors compared to individual data- and expert-based factor selection methods. Moreover, application of the integrated method significantly improves the performance of effort estimation in terms of accuracy and precision. Many software organizations are still proposing unrealistic software costs, work within tight schedules, and finish their projects behind schedule and budget, or do not complete them at all [12]. This illustrates that reliable methods for managing software development effort and productivity are a key issue in software organizations.

One essential aspect when managing development effort and productivity is the large number of associated and unknown influencing factors or productivity factors [33]. Identifying the right productivity factors increases the effectiveness of productivity improvement strategies by concentrating management activities directly on those development processes that have the greatest impact on productivity. On the other hand, focusing measurement activities on a limited number of the most relevant factors reduces the cost of quantitative project management [6].

Software development productivity is an important project management concern. One study reports that a 20% improvement in software productivity will be worth $45 billion in the U.S and $90 billion worldwide [1]. As a result, a number of empirical studies of software productivity have appeared in the literature over the past three decades. Scacchi has published a report that examines empirical investigations in relation to software development attributes, tools, techniques, or some combination of these that have a significant impact on productivity of software production [2]. These studies focus on the development of large scale software

development. Twelve major software productivity measurement studies are reviewed including those at IBM (Albrecht [3, 4]), TRW (Boehm [1, 5, 6, 7]), NASA (Bailey and Basili [8]), ITT (Vosburg et.al [9]), and international projects (Lawrence [10], Cusumano and Kemerer [11]). In addition, Scacchi examines a number of other theoretical and empirical studies of programmer productivity, cost-benefit analysis, and estimation of software cost (Thadhani [12], Lambert [13], Cerveny and Joseph [14].

MEASURE OF PRODUCTIVITY

In general, productivity is understood as a ratio of outputs produced to inputs used. However, researchers may use different outputs and inputs for measuring productivity. IEEE Standard 1045 calculates productivity in terms of effort as an input and lines of code or function points as output [16]. The two most common methods for measuring complexity or size of a software development project are Function Points and Lines of Code.
The main limitation of the LOC model is that it depends on the accuracy of an early estimate of lines of code. This estimate is usually based on the past experience of the systems analyst. Certainly, most organizations would find it difficult, if not impossible, to locate experienced analysts who could come up with an accurate estimate of the system size using a LOC model [17]. A third problem with LOC model is that it does not take into account the resources available to the systems development team. These include among other things the types of language used in coding, software tools, the skills and experiences of the team itself [18].

An alternative method for estimating systems development effort was developed by Albrecht [3]. Albrecht introduced the concept of Function Points (FP) to measure the functional requirements of a proposed system. In FP modeling the size of the system is determined by first identifying the type of each required function in terms of inputs, outputs, inquiries, internal files, and external interface files. To calculate the value of function points for each category, the number of functions in each category is multiplied by the appropriate complexity weight. The total systems effort is then calculated by multiplying the sum of function points for all categories by the Technical Complexity Factor (TCF). The TCF is determined by assigning values to 14 influencing project factors and totaling them. Readers unfamiliar with the FP model are referred to Albrecht and Gaffney [4]. Albrecht argued that FP model makes intuitive sense to users and it would be easier for project managers to estimate the required systems effort based on either the user requirements specification or logical design specification [3]. Another advantage of the FP model is that it does not depend on a particular language. Therefore, project managers using the FP model would avoid the difficulties involved in adjusting the LOC counts for information systems developed in different languages. In this paper I have used the following equation for calculation of productivity: Productivity = Effort/Function Points

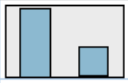
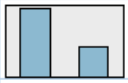
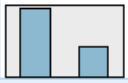
METHODOLOGY

Data Mining may be defined as the process of finding potentially useful patterns of information and relationships in data. As the quantity of clinical data has accumulated, domain experts using manual analysis have not kept pace and have lost the ability to become familiar with the data in each case as the number of cases increases. Improved data and information handling capabilities have contributed to the rapid development of new opportunities for knowledge discovery. Interdisciplinary research on knowledge discovery in databases has emerged in this decade. Data

mining, as automated pattern recognition, is a set of methods applied to knowledge discovery that attempts to uncover patterns that are difficult to detect with traditional statistical methods. Patterns are evaluated for how well they hold on unseen cases. Databases, data warehouses, and data repositories are becoming ubiquitous, but the knowledge and skills required to capitalize on these collections of data are not yet widespread. In this research As a First step I used Auto-Classification tool in SPSS Modeler which applies 11 different algorithms shown in Figure 1. The most efficient algorithms with highest accuracy rates are displayed based on current data set used for analysis.

The following is a brief description of the algorithms suggested and displayed by Auto-Classification as the most accurate models as shown in Figure 1.

Figure 1. The Most Efficient Algorithms

Use?	Graph	Model	Build Time (mins)	Max Profit	Max Profit Occurs in (%)	Lift(Top 30...	Overall Accuracy (%)	No. Fields Used	Area Under
☑		C&R Tree 1	< 1	5	2	1.271	72.059	7	0.603
☑		Neural Net 1	< 1	0	1	1.02	69.118	7	0.602
☑		C5 1	< 1	0	1	1	70.588	7	0.5

Decision Trees- Decision trees and rule induction are two most commonly used approaches to discovering logical patterns within medical data sets. Decision trees may be viewed as a simplistic approach to rule discovery because of the process used to discover patterns within data sets.

Decision tree is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches. Initially, you start with a training set in which the classification label (say, "productive" or "non-productive") is known (pre-classified) for each record. All of the records in the training set are together in one big box. The algorithm then systematically tries breaking up the records into two parts, examining one variable at a time and splitting the records on the basis of a dividing line in that variable (say, FP> 30 or FP<=30). The object is to attain as homogeneous set of labels (say, "productive" or "non-productive ") as possible in each partition. This splitting or partitioning is then applied to each of the new partitions. The process continues until no more useful splits can be found. The heart of the algorithm is the rule that determines the initial split rule [14].

Artificial Neural Networks (ANN) - Artificial neural networks are defined as information processing systems inspired by the structure or architecture of the brain (Caudill & Butler, 1990). They are constructed from interconnecting processing elements, which are analogous to neurons. The two main techniques employed by neural networks are known as supervised learning and unsupervised learning. In unsupervised learning, the neural network requires no initial information regarding the correct classification of the data it is presented with. The neural network employing unsupervised learning is able to analyze a multi-dimensional data set in order to discover the natural clusters and sub-clusters that exist within that data. Neural networks using this technique are able to identify their own classification schemes based upon the structure of the data provided, thus reducing its dimensionality. Unsupervised pattern recognition is therefore sometimes called cluster analysis [3, 16, and 17].

DATA

The data used in this research project was collected by The International Software Benchmarking Standards Group (ISBSG). The group gathered information from 1238 software projects from around the world. Projects cover a broad cross-section of the software industry. In general, they have a business focus. The projects come from 20 different countries. Major contributors are the United States (27%), Australia (25%), Canada (11%), United Kingdom (10%), Netherlands (7%), and France (7%). Major organization types are insurance (19%), government (12%), banking (12%), business services (10%), manufacturing (10%), communications (7%), and utilities (6%).

Projects types include enhancement projects (50%), new developments (46%), and 4% are re-developments. Application types consist of Management Information Systems (38%), transaction/production systems (36%), and Office Information Systems (5%). Nearly 3% are real-time systems.
Over 70 programming languages are represented. 3GLs represent 57% of projects, 4GLs 37%, and application generators 6%. Major languages are COBOL (18%), C/C++ (10%), Visual Basic (8%), Cobol II (8%), SQL (8%), Natural (7%), Oracle (7%), PL/I (6%), Access (3%), and Telon (3%). Platform for projects include mainframe projects (54%), midrange (24%), and microcomputers (22%).
Sixty-two (62%) of projects use a standard methodology that was developed in-house), 21% use a purchased methodology. Only 12% do not follow a methodology. The use of CASE tools ranges from 21% of projects using upper CASE, down to 10% for integrated CASE tools. CASE tools of some type are used in 51% of projects. Traditional system modelling techniques (data modelling, process modelling, event modelling, business area modelling) are used in 66% of projects. They are the only techniques listed in 27% of projects; 39% use a combination of traditional modelling and other techniques. The most common single technique is data modelling, used in 59% of projects. RAD/JAD techniques are used in 28% of projects. Object oriented techniques are used in 14% of projects. Prototyping is used in 29% of projects.
Data in the ISBSG database had to be cleaned and pre-processed in order to get, relevant and complete data for analysis. Records with missing value of attributes were excluded and the character values of text attributes or variables were transformed to numeric values. Function points count, total work effort in hours, team size, development platform (mainframe, mid-size, PC), language type (3GL, 4GL, Application Generator etc.), whether a software development

methodology was used, programming language and development type (new, enhancement, etc.) attributes were considered for analysis. Productivity attribute was calculated by dividing total work effort in hours by count of function points. Once the data was pre-processed, 468 usable projects were available for analysis.

Data in the ISBSG database had to be cleaned and pre-processed in order to get, relevant and complete data for analysis. Records with missing value of attributes were excluded and the character values of text attributes or variables were transformed to numeric values. Function points count, team size, development platform (mainframe, mid-size, PC), language type (3GL, 4GL, Application Generator etc.), whether a software development methodology was used, programming language and development type (new, enhancement, etc.) attributes were considered for analysis. Productivity attribute was calculated by dividing total work effort in hours by count of function points. Once the data was pre-processed, 468 usable projects were available for analysis.

DECISION TREE ANALYSIS

Before using this algorithm data has been balanced and partitioned into training and testing samples. You can use Balance nodes to correct imbalances in dataset. In our dataset balancing is used in order to make the number of productive and non-productive cases close to equal. SBGI Dataset is partitioned into 70% for training and 30% for testing the models. Figure 4 shows that programming language is the most important variable and function point is the least important. While the importance of programming language is theoretically make sense and confirms the previous findings, the lack of importance for function points is surprising. The function point represents the complexity and size of a software project and you may expect that it should have a great influence on productivity of software development. Both the confusion matrix and the gain chart indicates CRT as a good model with 72% accuracy.

ANN ANALYSIS

The same balancing and partition options used for ANN analysis. A Multilayer Perceptron network was used for this analysis. The ANN model resulted in a different ranking of important variables. Function Points is the most important followed by Team Size and Programming Language. This result seems to be similar to the results of previous results. Confusion matrix shows lower accuracy for ANN model (69%) than accuracy for CRT model (72%).

CONCLUSION

The two major classification algorithms CRT and ANN that were recommended by the Auto Classifier tool in SPSS Modeler used for determining the most important variables (attributes) of software development in PC environment. While the accuracy of classification of productive versus non-productive cases are relatively close (72% vs 69%), their ranking of important variables are different. CRT ranks the Programming Language as the most important variable and Function Points as the least important. On the other hand, ANN ranks the Function Points as the most important followed by team size and Programming Language.

Let us consider the results of CRT which is more accurate in terms of classification. In CRT model methodology, application type, language type, team size, and specially function points which represents the size and complexity of the software development are not indicated as important variable. Lack of or poor use methodology, which is also listed as the least important variable in ANN model, has clearly an effect on productivity of software development and may explain the lower productivity in PC platform application. However, the low importance of team size and function points is puzzling.

There is a need for using a larger sample size and may be from different repositories to validate or reject the results of this study. Conducting research on Mainframe platform and comparing the results with the results of the PC platform would also clarify further the productivity issue.

References Are Available Upon Request