Empirical Software Engineering, 5, 125–154 (2000)

Assessing and Understanding Efficiency and Success of Software Production

A. VON MAYRHAUSER avm@cs.colostate.edu
Computer Science Department, Colorado State University, Fort Collins, CO 80523, USA

C. WOHLIN claes.wohlin@telecom.lth.se
Dept. of Communication Systems, Lund University, Box 118, SE-221 00 Lund, Sweden

M. C. OHLSSON magnus.ohlsson@telecom.lth.se
Dept. of Communication Systems, Lund University, Box 118, SE-221 00 Lund, Sweden

Abstract. One of the goals of collecting project data during software development and evolution is to assess how well the project did and what should be done to improve in the future. With the wide range of data often collected and the many complicated relationships between them, this is not always easy. This paper suggests to use production models (Data Envelope Analysis) to analyze objective variables and their impact on efficiency. To understand the effect of subjective variables, it is suggested to apply principal component analysis (PCA). Further, we propose to combine the results from the production models and the analysis of the subjective variables. We show capabilities of production models and illustrate how production models can be combined with other approaches to allow for assessing and hence understanding software project data. The approach is illustrated on a data set consisting of 46 software projects from the NASA-SEL database (NASA-SEL, 1992). The data analyzed is of the type that is commonly found in project databases.

Keywords: Software project database, quantitative project assessment, production models, principal components analysis, analysis of subjective factors

1. Introduction

Increasingly, software productivity is a major concern in software development. Most new development methods, like object-oriented development (and structured development before it) are advocated because they promise higher levels of productivity. The same holds for tools and languages. Yet, many more factors influence productivity and it is usually not obvious how they interact. A key step is to analyze existing project data to assess and understand sources of inefficiencies in the projects that have been executed as well as factors which influence whether a project is viewed as successful or not. Furthermore, the analysis would help in quantifying relationships among the data, which can form the basis for identifying suitable actions to increase productivity. This requires the following:

- identification of relevant software production factors (process and product) that have a major influence on the characteristics of the software produced, on the efficiency of production and on the success of software projects.
- identification of relevant variables. For the software process, these relate to software quality measures and efficiency of its production. In this context, we consider software quality to include key desirable attributes of the software such as reliability, usability,

maintainability, etc. Efficiency of production relates to the consumption of resources in software production.

- modelling of the transformation of resources into production outputs. A classic approach is to use production models (Bessent et al., 1980; Charnes et al., 1978; Norman and Stoker, 1991). We propose to combine this approach with other statistical methods to assess and understand efficiency and success in software production.

While production models (or Data Envelope Analysis) have been used in Operations Research for quite a while, their use in the computing field has been limited (Banker et al., 1991; Roeseler and Mayrhauser, 1992). One study used production models to improve the efficiency of software maintenance (Banker et al., 1991). One reason for the limited use is that their success in the software development and maintenance domain requires appropriate parameterization (i.e., suitable metrics). In Roeseler (1991), Roeseler and Mayrhauser (1992) we did this parameterization for the computer system tuning domain. We developed a hierarchical set of factors and families of measures that, with the production model, make it possible to improve computer system services for users. Like computer systems, we can also consider software development and maintenance as a production process and model it accordingly. Inputs to the production model are various indicators of resources (effort, tools, capital, expertise, etc.). Outputs reflect the characteristics of software produced (size, quality, etc.). The production model then identifies which development activities were efficient and which factors are related to inefficiencies. This targets specific production activities for improvement.

The production model not only identifies all factors contributing to inefficiencies, it quantifies the degree of inefficiency. It also determines how much a factor level has to change to no longer cause inefficiency. At first glance, this would indicate almost a silver bullet for quantitative assessment, as it not only identifies inefficiencies, but also points to production factors that must improve. Root cause analysis can then further identify reasons for undesirable production factor values.

To make production models useful for quantitative assessment and improvement requires a hierarchy of metrics for further analysis of production model results.

The production modelling approach is complemented with a statistical analysis of subjective variables that are often collected in software projects (Wohlin and Ahlgren, 1995; Wohlin et al., 1995). The objective is to allow for a complementary analysis and hence provide a means for performing triangulation. The main emphasis is on the production models since they have been rarely applied in software engineering previously (Banker et al., 1991; Mayrhauser and Roeseler, 1993; Mayrhauser and Roeseler, 1993). The statistical methods used to analyze the subjective variables are an important complement to the production model results. The models introduced are illustrated in a case study where 46 software projects are evaluated using different types of methods to analyze objective and subjective data respectively.

The approach is outlined in Figure 1. The figure illustrates a common situation: project data include both objective and subjective metrics. The objective data is analyzed using production models to identify efficient and inefficient projects. Section 2 describes how production models work and how they analyze data. The methods to analyze the subjective

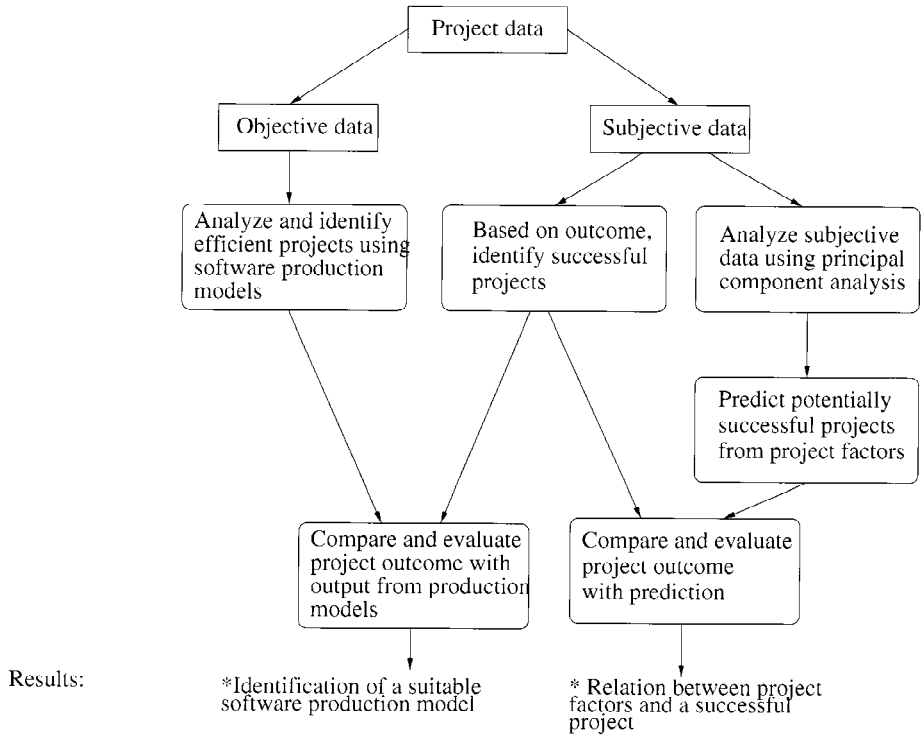

Figure 1. Outline of the evaluation approach.

data are introduced in Section 3. This includes both principal component analysis and ways of comparing agreement between different models. In particular, it is possible directly from the subjective data to assess which subjective factors are most important for project success. Section 4 presents an approach to perform a combined analysis. Section 5 illustrates and evaluates current viability of production models for process assessment on a case study of 46 projects. Moreover, the case study illustrates how production models may be combined with an analysis of subjective data to provide an improved understanding of software projects and their outcomes. The main objective of the case study is to illustrate how to use a combined approach that evaluates both productivity and success of projects based on objective and subjective variables. The results from the case study are promising and encouraging, but not spectacular, due to the type of data in the project data base. As with other methods, the type and quality of the data influences the results a model or analysis method is able to provide. The main lesson learned is that it is beneficial to apply different analysis techniques to different types of data and to combine the results. This provides a way of performing triangulation for project data. Finally, some conclusions are presented in Section 6.

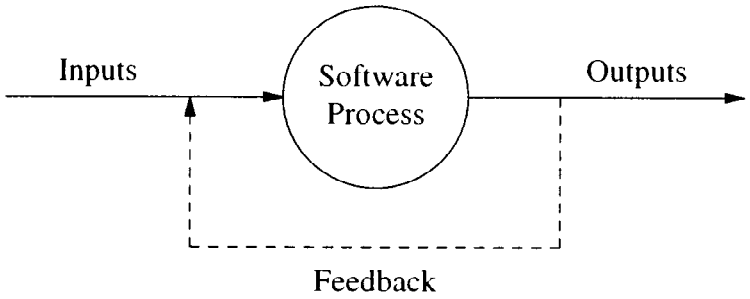

Figure 2. A simple causal model of the software process.

2. Production Models

Any software process defines the production of software or at least of deliverables that are part of the software production process (e.g. the testing process produces various test results). Technical and economic analysis perspectives are integrated by a causal model that describes software production in terms of *production systems*. Figure 2 depicts a production system model that describes how (possibly multiple) input factors (resources in the general sense) are transformed into (possibly multiple) output factors (deliverables, quality aspects, etc.) using a software development process. Feedback in the production system model is provided through managerial decision making (e.g., deciding on process, project plan and resources).

The motivating idea behind the production system model is the notion that the software development process transforms software feature requests into software products. The time to do this varies. So do the resources used.

Software productivity optimization requires identification of the most favorable set of operating conditions. Most favorable (optimal) production conditions are characterized by the attainment of *maximal* levels of outputs, given a set of production input quantities.

2.1. Production Functions

Instead of being "random," we assume that a functional relationship between software production process inputs and outputs exists. In the production system model, the production function f relates the inputs to the outputs and describes the resource transformation process. A general production function that transforms N inputs $Q_I = (Q_I^1, Q_I^2, \ldots, Q_I^N) \in R_+^N$ into M outputs $Q_O = (Q_O^1, Q_O^2, \ldots, Q_O^M) \in R_+^M$ is given by

$$f(Q_I) = Q_O \tag{1}$$

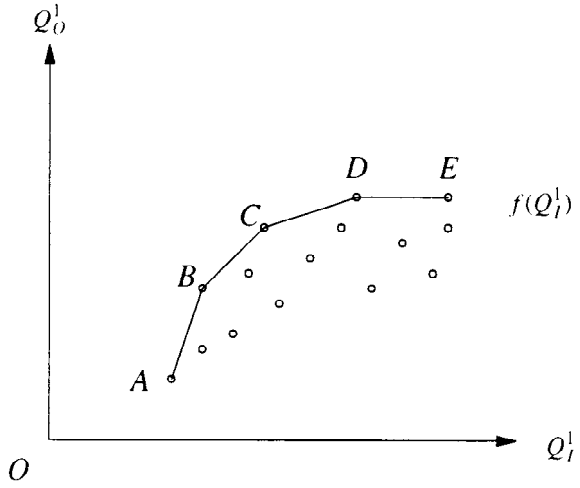

Figure 3. Estimation of the production function.

where:

- Q_O^m is the amount of the m-th output, $m = 1, \ldots, M$
- Q_I^n is the amount of the n-th input, $n = 1, \ldots, N$
- $f\colon R_+^N \mapsto R_+^M$ production function giving maximal output for given input.

We assume that the N inputs and M outputs are strictly positive and are measured on a ratio scale (hence $Q_O \in R_+^M$ and $Q_I \in R_+^N$). A production process is now modeled by a production function $f\colon R_+^N \mapsto R_+^M$, where Equation 1 gives the maximal output vector Q_O obtainable from an input vector Q_I.

Because of the complex interactions of software production process components, the analytic specification of the production function according to Equation 1 is rarely feasible. In the absence of quantitative means to determine the interactions and causalities of production components in the production process *directly*, we take an empirical approach based on historical production observations.

To illustrate the basic concept, assume for the moment a situation with one input and one output variable. Figure 3 depicts observations on a production process characterized by one input (Q_I^1) and one output (Q_O^1) factor (e.g. the traditional productivity metric based on effort (the input variable) and lines of code (the output variable)). For each input amount Q_I^1, the corresponding output amount Q_O^1 is shown. Given input interval and observation period, the points A, B, C, D, and E, represent the maximum outputs obtained and therefore represent the (piecewise linear) approximation to the production function f of the underlying (actual) production process. Because maximum output levels are not

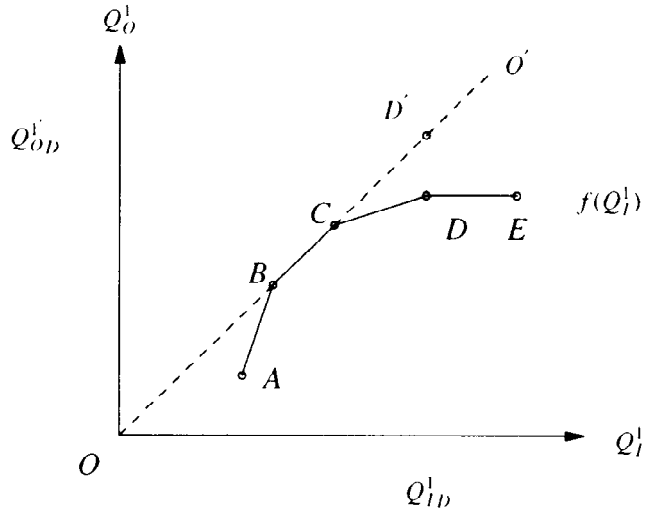

Figure 4. Production process resource allocation.

attained, output amounts $Q^1_{O_i} < f(Q^1_{I_i})$ in Figure 3 indicate inefficiencies in the production process for the ith period.

Consider a ray $\overline{OO'}$ through the origin and tangent to the production function f. This is the efficiency frontier. Points B and C now represent preferred operating conditions from a resource utilization point of view, since these points maximize the ratio output over input (see Figure 4). In particular, the input amount $Q^1_{I_D}$ employed at point D should result in output amounts $Q^1_{O_{D'}}$ associated with the hypothetical production point D', where D' lies on $\overline{OO'}$. (Conversely, output amounts $Q^1_{O_D}$ at D should only require input amounts $Q^{1'}_{I_D}$, where $Q^{1'}_{I_D}$ is obtained by reading straight down at the intersection of a horizontal line through D with $\overline{OO'}$). This example models a fixed return to scale, i.e., f is a straight line. See section 2.3 for how to treat a variable return to scale.

A measure of efficiency of the ith production period is now provided by the ratio of *observed* output amounts to *desired* output amounts (the desired output amounts are along the frontier); i.e.,

$$0 \leq \frac{f(Q^1_{I_i})}{Q^{1'}_{O_i} | Q^1_{I_i}} \leq 1 \tag{2}$$

where $f(Q_{I_i})$ and $Q^{1'}_{O_i} | Q^1_{I_i}$ denote the observed and the desired output quantities, respectively, given $Q^1_{I_i}$ amounts of input employed.

Given the observations above, economic principles in the production system model consider the quantity (value) of the deliverables (outputs) generated given the resources (inputs)

employed. Software production efficiency is then, naturally, defined by an organization's ability to produce outputs given inputs consumed. An organization is at its most productive when resources are optimally utilized (i.e., maximum outputs are obtained, given resources consumed). Knowledge of a process's relative (in)efficiency, amount of excess input, and desired output goals can then be used to:

- evaluate the efficiency of a software process. That is, are sufficient amounts of output produced? Output is characterized by factors considered desirable for the software. Such factors include size, performance, quality etc.

- decide on tactics to improve efficiency. Solutions of the production model point out which input factors must be improved to achieve desired output levels. So, for example, the model may suggest that design review effort (staff hours) may need to be increased by a certain amount to reduce inefficiency for the output factor software reliability (presumably too low a level).

2.2. Optimal Efficiency

A ratio definition of efficiency according to Equation 2 can be expressed in terms of a fractional programming model. Consider, again, a one input/one output situation, where Q_O^1 represents the output *lines of code delivered* (*LOC*) and O_I^1 represents the input *developer effort* (E). The ratio LOC/E provides an obvious efficiency metric.[1]

Let E_k and LOC_k be the observations on input and output values of the kth observation period, respectively, where $k = 1, 2, \ldots, n$. Further, let E_{opt} and LOC_{opt} be the observation-pair that maximizes the ratio $\frac{LOC_k}{E_k}$. By our definition, the corresponding period is the most efficient production period in the reference set of n periods.

Suppose, for the moment, that the output *LOC* is constant, i.e., $LOC_k = const$. An efficiency metric is now given by $Eff_k = LOC_k / LOC_{opt}$. In the case of $k = opt$, the result will be one, and maximum efficiency is achieved. In all cases where $LOC_k < LOC_{opt}$, the result will be less than one; that is, maximum efficiency is not achieved.

A productivity indicator based on one input and one output is extremely limited and problematic. There are many other input factors that influence software production in a significant way. Output also constitutes more than lines of code. First, software development produces a variety of non-code deliverables. Second, size alone does not adequately describe a software product. Therefore, let us now expand the simple one input, one output situation to the multi-input, multi-output case.

The above ratio definition of efficiency is a special case of the economic definition of productivity $P = Output/Input$ and has been extended to multiple input and multiple output situations for prediction and understanding of production processes (Charnes et al., 1978). Suppose the efficiency of a set of production periods, $j = 1, 2, \ldots, n$, is to be evaluated. For all periods, a common set of inputs X_j that contribute to the production of outputs Y_j

is defined, where

$$X_j = \begin{Bmatrix} x_{1,j} \\ x_{2,j} \\ \cdot \\ x_{i,j} \\ \cdot \\ x_{m,j} \end{Bmatrix}, Y_j = \begin{Bmatrix} y_{1,j} \\ y_{2,j} \\ \cdot \\ y_{r,j} \\ \cdot \\ y_{s,j} \end{Bmatrix}. \tag{3}$$

Assume that observations on X_j and Y_j are obtained. $x_{i,j} > 0$ represents the observed value of the ith input for period j, and $y_{r,j} > 0$ represents the observed value of the rth output.[2]

The nonlinear programming definition of Equation 4 generalizes the output/input ratio measure of efficiency by calculating the ratio of a weighted sum of outputs to a weighted sum of inputs. The fractional programming model according to Equation 4 is a mathematical representation of the efficiency metric Eff_k for situations where dimensional units of inputs and outputs are different.

$$\text{maximize} \quad h_k = \frac{\sum_{r=1}^{s} u_r y_{r,k}}{\sum_{i=1}^{m} v_i x_{i,k}}$$

subject to:

$$1 \geq \frac{\sum_{r=1}^{s} u_r y_{r,j}}{\sum_{i=1}^{m} v_i x_{i,j}}, \quad j = 1, 2, \ldots, n, \tag{4}$$

$$0 < u_r, \; r = 1, 2, \ldots, s,$$

$$0 < v_i, \; i = 1, 2, \ldots, m.$$

In Equation 4, $k = 1, 2, \ldots, n$, respectively, indexes the different production periods in the reference set, and h_k is the normalized efficiency index for the kth period. If $h_k < 1$, then the kth period is inefficient compared to the reference set of periods. In particular, if $h_k = .75$, then period k is only 75% as efficient as the most efficient period (or periods) in the reference set. If $h_k < 1$, some inputs are not fully utilized during the kth period or are employed in the wrong proportions, and some other periods are getting more output per unit of input for these resources. u and v represent marginal rates of conversion that vary for each period evaluated. The u and v are constrained to be strictly positive and represent an overall increase in efficiency of a period, if one less unit of an input is utilized or one more unit of an output produced. These conversion rates allow for situations where dimensional units of inputs and outputs are different (i.e., most practical cases).

Application of the duality relations of linear programming to Equation 4 and change of variables yields a linear programming model according to Equation 5 (Charnes et al., 1978; Norman and Stoker, 1991). This model provides the basis for establishing a period's

efficiency rating, and the desired input and output quantities that render an inefficient period as efficient as the most efficient period(s) in the reference set.

$$\text{maximize } z_k - \epsilon \left(\sum_{r=1}^{s} S_r^+ + \sum_{i=1}^{m} S_i^- \right)$$

subject to:

$$0 = y_{r,k} z_k - \sum_{j=1}^{n} y_{r,j} \lambda_j + S_r^+, r = 1, 2, \ldots, s, \tag{5}$$

$$x_{i,k} = \sum_{j=1}^{n} x_{i,j} \lambda_j + S_i^-, \quad i = 1, 2, \ldots, m,$$
$$0 \leq \lambda_j, S_r^+, S_i^- \qquad \forall j, r, i.$$

S_r^+ in Equation 5 represents nonnegative slack (i.e., additional output) associated with the outputs, and S_i^- represents nonnegative slack (i.e., excess resources) associated with the inputs. S_r^+ is the additional amount of the rth output that is be expected, and S_i^- is the amount by which the ith input has to be reduced, if the kth production period were to become efficient. At an optimum we have $h_k^* = 1/z_k^*$. Sources of inefficiency are indicated by $z_k^* > 1$, and/or $S_r^+ > 0$ or $S_i^- > 0$.[3]

The λ_j represent an optimal basis for inefficient production periods, and a hypothetical efficient period can be constructed from the weighted average of two (or more) of the observed efficient periods. Efficient periods are represented by $\sum_{j=1}^{n} \lambda_j \vec{P}_j$, where $\vec{P}_j$ is a vector consisting of the outputs and inputs for the jth period. ϵ in Equation 5 is a small quantity (e.g., 10^{-5}) that is always smaller than any positive value that may be assumed by λ_j. This quantity provides algebraic closure of the constraint set in linear programming, and it guarantees that optimal solutions are at finite, non-zero, extremal points (Charnes and Cooper, 1985).

No production period can be rated efficient unless both $z_k^* = 1$ and the slack variables $S_r^+ *$ and $S_i^- *$ are all zero. If the original observations on X_k and Y_k are applied to

$$\begin{aligned} x'_{i,k} &= x_{i,k} - S_i^{-*}, \quad i = 1, 2, \ldots, m, \text{ and} \\ y'_{r,k} &= y_{r,k} z_k^* + S_r^{+*}, \quad r = 1, 2, \ldots, s, \end{aligned} \tag{6}$$

new values $x'_{i,k}, y'_{r,k}$ are obtained that render period k efficient. That is, if X'_k and Y'_k were observed (instead of X_k and Y_k), then period k would be as efficient as the most efficient period (or periods) in the reference set. Notice that output augmentation and input reduction in Equation 6 may be required at the same time.[4]

The differences

$$\begin{aligned} \Delta x_{i,k} &= x_{i,k} - x'_{i,k}, \quad i = 1, 2, \ldots, m, \\ \Delta y_{r,k} &= y'_{r,k} - y_{r,k}, \quad r = 1, 2, \ldots, s, \end{aligned} \tag{7}$$

represent the estimated amounts of input and output inefficiencies, respectively, in the X_k, Y_k for the kth period (Charnes and Cooper, 1985).[5]

2.3. Extensions and Practical Considerations

The illustration of the problem formulation and solution mechanism so far covers the basic form of production models and Data Envelope Analysis (DEA) (Bessent et al., 1980; Charnes et al., 1978; Charnes and Cooper, 1985; Norman and Stoker, 1991). Several extensions and suggestions for dealing with practical issues exist.

Interval and rank data are frequently used in assessing various aspects of projects (Wohlin and Ahlgren, 1995; Wohlin et al., 1995). A data envelope analysis case study in Norman and Stoker (1991) uses "pitch" of a shop (two values, 1 or 2) in an analysis of the efficiency of 45 shops of a British national retailer. Current practice in DEA is to include such variables "with care." This means that sometimes they are used as if they represented ratio level data, other times they are dealt with in a separate analysis.[6]

Another issue arises with variable returns to scale. The basic form of the model assumes a fixed return to scale, as evidenced by the straight line $\overline{OO'}$ in Figure 4. An example where a model with diminishing returns to scale would be appropriate is the software testing phase. As testing progresses, the yield usually diminishes. DEA considers this situation by introducing a modified weighted sum ratio with an additional constant in the numerator of the objective function of Equation 5 (Norman and Stoker, 1991; Charnes et al., 1978). For estimating most productive scale size see Banker (1984).

Finally, the selection of which variables to include is an important issue as it affects the results of the analysis. In software development, key factors are size of code and effort of production. Besides these, we see a wide range of other variables that either relate to other types of items produced or to special production conditions that affect the production process. If we start with the basic variables first, then add other variables over the course of the analysis, we must understand the effects of adding more variables to the model. Nunamaker (1985) establishes the following effects on efficiency of existing data points (in this case the individual software projects to be analyzed) when variables are added to a model:

1. Efficient projects stay efficient.

2. Inefficient projects with respect to a model with fewer variables may become efficient when variables are added. This is true whether the new variables are correlated or not. It also holds when variables are disaggregated (as for example when breaking down overall effort into development versus maintenance effort).

To put it another way: additional variables consider more reasons why a project should be efficient, i.e., models with additional variables are more "forgiving." This makes a judicious selection of variables very important.

Variable selection is usually done via a panel of experts, prior statistical work, the analyst's knowledge of the decision environment, or a combination of the three approaches.

3. Principal Component Analysis and Agreement Index

Software production models work well for objective variables. However, many software project databases include subjective evaluations of projects. This can be handled in two ways:

- pretend the subjective rankings are objective and include them in the production model analysis (Norman and Stoker, 1991),
- ignore subjective variables.

Neither approach is desirable, so we propose a third. It is important to gain as much understanding as possible from the subjective variables. In this particular case, principal component analysis is used to further enhance the understanding and support the assessment of what constitutes an efficient and successful project. We do not describe PCA at the same depth as the software production models since descriptions are more generally available in books on statistics covering multivariate statistics. PCA has also been more widely applied in analyzing software engineering data than software production models. Some examples of PCA applied in software engineering can be found in Khoshgoftaar and Lanning (1994), Khoshgoftaar et al. (1996), Ohlsson and Wohlin (1998).

The data in too many databases are collected without a purpose (except to collect data). Therefore many measures are highly correlated because they measure almost the same thing. Various techniques exist to extract a useful subset. One is Principal Component Analysis (PCA) (Gorsuch, 1983; Kachigan, 1986; Ramsay and Silverman, 1997). PCA groups correlated variables or variables with the same behaviour into a number of factors where each factor accounts for the maximum possible amount of the variance for the variables being analyzed. The number of factors extracted may vary depending on the data set and the method chosen for extraction. PCA can help in reducing the number of variables, and it can also provide support in identifying variables that vary together. It can, for example, be applied to analyze similarities between software development projects (Khoshgoftaar and Lanning, 1994). It has also been used when building models for prediction of fault-prone software components, see for example (Khoshgoftaar et al., 1996).

There exist some problems using PCA. The analysis requires formally three or more interval variables because it is defined as a linear model. Non-parametric PCA methods have been developed, although they are not as commonly used. The results of applying linear versus non-parametric PCA often turn out to be the same and we can measure the quality of the PCA results by the significance level of the results (Briand et al., 1996). When applying PCA to non-interval variables, it is necessary to be a little cautious. This may, for example, include not accepting the results from the analysis without making sure that the results correspond to expectations.

Two objectives may be identified when applying PCA to the subjective variables in the analysis of project data. The first objective is to reduce the number of variables that need to be considered, and the second objective is to identify project factors (project characterization) that are closely related to the project outcomes (e.g. software quality, timeliness). Here, we distinguish project factors from the input factors in the production model. The project factors

Table 1. Illustration of the kappa statistics.

	Successful	Unsuccessful	Sum
Successful	p_{11}	p_{12}	p_{1+}
Unsuccessful	p_{21}	p_{22}	p_{2+}
Sum	p_{+1}	p_{+2}	

are a subjective characterization of the project, while the input factors in the production model are project resources in a general sense.

Project outcomes also are often collected as rank order subjective variables. The project output factors in the production model refer to objective output, while the project outcome factors in the PCA are related to subjective judgement of project results. In this case, PCA can identify which subjective project factors relate to outcome factors.

Given a threshold of what constitutes a positive outcome (e.g. of the timeliness of delivery, the quality of the delivered software, etc.), it is possible to identify successful projects. At this point it also becomes possible to evaluate which projects were both successful (as evidenced by the analysis of subjective variables) and highly productive (as defined by the efficiency index of the production model). The comparison between these two different models can be evaluated using, for example, an agreement index, often referred to as kappa statistic (Altman, 1991). In software engineering, the kappa statistic has been applied to interrater agreement of process assessments (El Emam, 1998).

Briefly, the kappa statistic can be explained as follows for the simple case with two raters (or models) and two levels (successful or unsuccessful project). Table 1 illustrates this type of problem in the form of a table, where the cells illustrate the proportions of the projects with a given rating according to model 1 and model 2. For example, $p_{11} = 0.20$ would mean that 20% of the projects are considered successful according to the subjective variables and successful (efficient) according to the production model. Similarly, p_{22} indicates the proportion of projects that both models consider unsuccessful. The columns and rows are summarized, which is indicated with p_+1 and so forth.

Based on Table 1, it is possible to derive an agreement index. Let P_A be the proportion in which there is agreement. Then, P_A becomes

$$P_A = \sum_{i=1}^{2} p_{ii} \tag{8}$$

This agreement includes that the agreement could have been obtained by chance. To take this into account, the extent of agreement that is expected by chance is defined as

$$P_E = \sum_{i=1}^{2} p_{i+} \times p_{+i} \tag{9}$$

The agreement index is then defined as

$$\kappa = \frac{P_A - P_E}{1 - P_E} \tag{10}$$

To be able to understand the degree of agreement, it is necessary to interpret the kappa statistic on a scale describing the degree of agreement. Several such scales exist, although they are mostly minor variations of each other. Three scales are presented in El Emam (1998). Here the benchmark suggested by Altman (1991) is used. The benchmark is shown in Table 2.

The agreement index requires comparable scales to determine successful projects in both models. This prevents setting thresholds for each model, since it is not possible to map one scale into another directly. To circumvent this problem, we use each model to rank the projects by success and identify the x most successful projects and then use the kappa statistic to compare.

The use of PCA and the agreement index for the subjective variables is illustrated in the case study in Section 5.

4. Combined Analysis

The production models provide information about which projects were judged to be efficient. This may be combined with an analysis of the subjective measures of project success. It is possible to relate the assessment of the production models to the analysis of the subjective variables. This allows for a combined analysis where the two analysis approaches work together and provide a better understanding and assessment than could have been achieved by only one of the approaches. In particular, two different scenarios are possible:

- The production models and the analysis of the subjective variables may pinpoint *the same projects*. In this case, the models support each other and either one of them could have been used for analysis.
- The production models and the analysis of the subjective factors may pinpoint *different projects or at least not exactly the same projects*. The models complement each other. This scenario means that the models address different aspects, and hence they should be used together.

The projects to focus on for the analysis are those that both approaches identifiy, i.e. projects which are both efficient and successful. In this scenario, these are the projects to learn from and try to emulate. It is possible to study whether the projects that were judged efficient by the production models also are judged successful based on the analysis of the subjective variables or vice versa. Thus, it is possible to identify which of the above scenarios applies to a particular data set. This may be done using a table such as presented in Table 1 and the agreement index presented above. It is particularly beneficial to be able to analyze both project efficiency and project success. This implies that it is possible to

Table 2. The Altman kappa benchmark.

Kappa statistic	<0.20	0.21–0.40	0.41–0.60	0.61–0.80	0.81–1.00
Strength of agreement	Poor	Fair	Moderate	Good	Very good

focus on projects which were both efficient and successful. The assessment of these aspects provides an excellent basis for understanding software success and efficiency, which then in the next step can form the input to improvement activities.

5. Case Study

5.1. Validate Model

Obviously, one case study cannot validate a general method. However, it can serve as an illustration and evaluation to which degree existing project data are likely to be useful when one tries to assess software projects' efficiency and success. Specifically, the following questions must be answered:

- does the production model identify efficient and inefficient periods of production?
- are efficient projects also normally successful projects?
- is the use of production models together with analysis of subjective variables a potentially useful approach?

Subsequently a case study is used to illustrate what the proposed approach can do and what it requires for successful use.

5.2. Project Data

The case study is based on data from the NASA-SEL database (NASA-SEL, 1992). Further information about NASA-SEL can be found in Basili et al. (1995). Project data in the database provide a rich set of project variables to choose from. They include parameters related to schedules and estimates, resource use (both manpower and computer use), a variety of product characteristics related to structure, size, growth, and change data. Subjective evaluations rank the projects in terms of problem complexity, schedule constraints, nature of requirements, team ability, management performance, discipline, software quality, etc.

In total, the database contains data from more than 150 projects. The projects span a five year period. Of these, we selected 46 for analysis, based on completeness of project data recorded. These 46 projects represented a variety of types of systems, languages, and approaches for software development. In all, there were 25 attitude systems, 8 attitude ground support systems, 10 simulators, 2 systems related to shuttle payloads, and one orbit system.

Table 3 summarizes the 46 projects used in the analysis. The first column provides a project identifier. We had complete objective data on the following factors that affect productivity.

Inputs

- TMH: Total technical and management hours expended on the project.
- SH: Total service hours expended on the project. Service hours measure the documentation efforts of the Technical Publication department (editing, reproduction, etc.).
- CPU: Total CPU hours used on all machines.

The first two relate to labor resources expended, the last is a measure of computing resources. While one could interpret CPU hours used as a pure use of a capital resource, the measure is said to have been used in reliability calculations, thus can be interpreted as relating to reliability growth efforts.

Outputs

- CHGS: Number of changes made to system components.
- PDOC: Number of pages of documentation produced for system.
- SLOC: Total SLOC (Source lines of code) for all components in system. This figure consists of (see below) new code (NewSLOC), modified code (ModSLOC), and reused code without modification (ReSLOC).
- NewSLOC: Total SLOC for all components in system classified as new.
- ModSLOC: Total SLOC for all components classified as modified.
- ReSLOC: Total SLOC for all components reused from other system without modification.

Note that SLOC $=$ NewSLOC $+$ ModSLOC $+$ ReSLOC (sum of all SLOC, whether new, modified, or reused). Table 3 shows the values of the primary inputs and outputs of the production model for all projects selected for analysis.

Besides these project data, we also decided to consider the subjective evaluation data of Table 4. Each item is ranked on a five point scale with the higher value denoting the more desirable outcome. The subjective variables can be grouped as indicated in Table 4. The categories are Problem, Team, Management, Execution, Infrastructure and Outcome.

5.3. Production Model Analysis of Objective Variables

We ran the production model with five possible parameterizations. Table 5 shows these five possible parameterizations of the production model (I-V). Table 6 summarizes results of running the models, listing all projects that were considered efficient or marginally inefficient by at least one of the five models. Table 6 shows projects that are efficient with a "+" in the table. Marginally inefficient projects are denoted by a "o" in the table. Efficient

Table 3. Project data included in production models.

Project	TMH	SH	CPU	CHGS	PDOC	NewSLOC	ReSLOC	ModSLOC
P1	12588	1109	382.3	1255	1613	45345	893	4673
P2	15760	4316	465	3228	2473	84729	7098	20041
P3	5791.5	1079	169.6	858	1120	20075	48618	6727
P4	14507.5	1231	402	2107	1793	49316	21825	4252
P5	14370.5	2744.4	497.2	2710	2458	76883	2834	5652
P6	15122.1	1926.5	309.5	2761	2695	61950	13266	14297
P7	1094.6	3.4	135.9	255	66	5540	99	0
P8	2520.7	527.6	79.3	275	300	5354	2449	1323
P9	19474.9	2846.2	987.3	2077	2107	45004	12616	9705
P10	17996.7	3266.9	810	1575	2360	44644	13016	8606
P11	4466	1194.2	63.2	255	763	14873	385	0
P12	2166.1	543.6	79.2	219	255	9627	18	527
P13	4633.2	695	180.5	541	760	10822	4118	2331
P14	20151.9	3819	790.6	3459	3017	98388	4416	7502
P15	3615.3	531	215.2	530	366	11878	1564	1323
P16	2942.7	338.1	58.6	660	527	12227	1065	1571
P17	1555.9	234.9	48.3	314	511	9568	3822	892
P18	4429.3	507.9	98.9	795	873	18680	6304	7838
P19	1004.9	201	14	103	136	2451	1099	1947
P20	1322.1	241.7	19.6	158	169	4160	365	0
P21	1140.5	128.4	35.9	300	284	7350	328	2049
P22	747.2	111.8	9.3	135	61	2052	0	0
P23	1293.1	162.7	22.9	289	163	4921	283	0
P24	42104.8	6808.1	1977.8	4193	5227	100470	29750	20642
P25	49476	5620	1337.1	2596	3870	137739	15635	5767
P26	15333.6	2207.3	138.13	261	1974	33196	6441	12067
P27	32083.4	4407.6	36.26	544	500	40201	134	1494
P28	12005	1524.5	248.8	412	1400	26986	2556	7363
P29	17057	1875	56.1	1776	1609	70951	10483	0
P30	54755.1	4718.5	177.2	1706	2500	194169	14109	28115
P31	49930.5	4312.9	155.46	1494	10690	141084	7934	29664
P32	123417	17157	729	5668	13122	292552	51366	17988
P33	28055.8	2124.9	911.46	983	2060	109807	7049	54246
P34	37806.4	2875.8	119.77	1325	4800	106834	5869	16156
P35	13657.5	1289.7	331.32	252	1406	59783	11450	20862
P36	12804	893.7	153.12	357	918	22175	5330	9538
P37	89513.8	7854.1	263.96	1404	4600	260382	11868	30876
P38	17975.5	1987.4	196.56	955	2100	63861	4399	38186
P39	11526	1034.5	410.87	297	1395	38327	11544	18277
P40	13214.6	1365.8	24.72	427	950	55289	11060	6063
P41	21657.7	2538.5	111.1	323	1200	41552	179016	50441
P42	18796.9	2126.4	701.51	341	2235	20859	39495	123663
P43	4726.5	855.1	250.23	47	650	2161	58591	5944
P44	11694.7	308	465.71	142	1035	12974	26275	28858
P45	4597.6	684.72	21.5	38	1500	10590	141006	2913
P46	2516	755.58	92.97	15	650	0	52026	9421

Table 4. Subjective evaluation factors.

Category	Factor	Explanation
Problem	COMP	Problem Complexity
	SCHE	Schedule constraints (loose=1, tight=5)
	RSTAB	Stability of requirements (unstable =1, stable =5)
	RQUA	Quality of requirements
	RDOC	Documentation requirements
	RREW	Rigor of requirements reviews
Team	TABI	Development team ability
	TAPP	Development team application experience
	TENV	Development team environment experience
	TSTAB	Stability of development team (unstable =1, stable =5)
Management	MPER	Management performance
	MAPP	Management application experience
	MSTAB	Stability of management team
	PROPL	Project planning discipline
	PPCOMPL	Degree to which plans were followed (compliance with plan)
Execution	PROGP	Use of modern programming practices
	FORC	Discipline in formal communication
	REMET	Discipline in requirements methodology
	DEMET	Discipline in design methodology
	TEMET	Discipline in testing methodology
	TEPLAN	Use of test plans
	QA	Discipline in quality assurance
Infrastructure	CM	Discipline in configuration management
	DEVSYS	Access to development system
	DEVTERM	Ratio of developers to terminals (low =5, high =1)
	MEM	Memory constraints
	RESTIM	System response time (poor =1, very good =5)
	SHWSS	Stability of hardware and support software
	EFFTOOL	Effectiveness of tools used
Outcome	AGGRE	Agreement of software with requirements
	QSOFT	Quality of software
	QDES	Quality of design
	QDOC	Quality of documentation
	TIMELI	Timeliness of delivery
	ACCTEST	Smoothness of acceptance testing

projects have an efficiency index of 1 while marginally inefficient projects have an efficiency index between .9 and 1.

All models clearly identified efficient and inefficient projects. Table 7 shows detailed efficiency indices for all projects and all models. It should be noted that the analysis should be conducted with care. It is especially important not to get into a situation where a lot of code or other documentation is produced to artificially inflate efficiency. Thus, it is suggested that this type of assessment is primarily used to understand software production rather than as assessment of personnel. If reasonable assumptions can be made about an

Table 5. Possible models.

	M-I	M-II	M-III	M-IV	M-V
Inputs:					
THM	*	*	*	*	*
SH			*	*	*
CPU	*	*	*	*	
Outputs:					
CHGS	*	*	*	*	*
PDOC		*	*	*	*
NewSLOC		*		*	*
ModSLOC		*		*	*
ReSLOC		*		*	*
SLOC	*		*		

legend: * means included in model

underlying quality assurance process that prevents artificially inflating production figures, this approach is credible.

Models I and II use the same inputs (technical/management effort and CPU hours), but differ in how they consider SLOC produced. Model I lumps all SLOC together into one output variable, while model II differentiates between new, modified and reused code. This allows the model to make implicit allowance for some types of SLOC to expend more effort than others. Not surprisingly, model II shows a larger number of efficient projects than model I. In particular, the last several projects (they reused code in a significant manner) are included in the efficient projects.

Neither model I nor model II consider service hours expended on a project. This implicitly assumes that service hours are proportional to management/technical effort. Should that not be an appropriate assumption, it is better to include this input factor in the model. Including service effort as an input variable should also consider number of pages of documentation produced as an output variable (the service effort measures the efforts of the technical publication department). Considering these two variables gives rise to models III and IV. They distinguish themselves in that model III lumps all SLOC produced into one variable, while model IV considers them separately. As with models I and II, model III is less "forgiving" than model IV, as it does not make allowances for the different types of SLOC and their possibly varying resource cost. Model IV, having the most parameters, results in the largest number of efficient projects.

A last consideration of the model relates to the use of CPU time as a production input. CPU, total CPU hours used on all machines, can be broken down into 13 types of machines. However, the majority of projects use at most two or three different types of machines. Logically, the best way to represent this resource usage as an input is to break it down into the 13 individual types. This creates sparse data across 13 inputs (and an ill-conditioned optimization problem). The results of model runs using the sparse data show unrealistic increases in efficiency for all projects. This is because the model now has 12 more input variables to the production model than before and thus is that much more forgiving. Strictly

Table 6. Efficient and marginally inefficient projects by model type.

Project	M-I	M-II	M-III	M-IV	M-V
P6		○		○	
P7			+	+	+
P16	○	+	+	+	
P17		+	+	+	+
P18		○		+	
P19		+		+	
P21	+	+	+	+	+
P22	+	+	+	+	
P23	+	+	+	+	
P29	+	+	+	+	
P30				+	
P31		+	+	+	+
P33				○	
P34				○	
P35				○	
P38		+		+	
P40		+		+	
P41		+		+	
P42		+		+	+
P44			+	+	+
P45	+	+	+	+	+
P46				+	+

legend: + efficient
○ marginally inefficient

speaking, the model does not allow zero values for specific variables, rather it evaluates the combined impact of a series of variables. Additionally, the model does not correlate well with otherwise identical runs using CPU as a single input. Thus, breaking down CPU usage by type of computer is not recommended.

In a situation like this, the modeler has two choices, (1) to use the aggregate variable CPU, or (2) to reject it outright. We show the result of rejecting CPU as a production variable in our model V. It does not model CPU resource usage, but distinguishes between different types of SLOC and amount of documentation as a production output. It is far less forgiving than model IV. It is in the nature of the technique that the greater the number of variables, the higher the proportion of units that achieve a relative efficiency of 1. Informally speaking, inefficiencies in production can be "explained away" by considering the effect of a set of environmental variables. Depending on the analyst's interest and what he or she considers the major productity drivers and outputs, one would choose one of these five models in step one of the data analysis.

Considering the results in table 6, two projects stand out, namely P21 and P45. They are robustly efficient, no matter which of the models are applied. What makes these projects so great from an efficiency perspective? To answer this question, we have to turn to the database for more information. P21 did not have any explanation in the message field of its database entry, but the P21 entry reports a high degree of verbatim code reuse (86%). Interestingly enough, P46, which had a similar code reuse ratio, is only considered efficient by models

Table 7. Efficiency indices for models I-V.

Project	M-I	M-II	M-III	M-IV	M-V
P1	.40	.57	.61	.68	.61
P2	.81	.86	.81	.86	.84
P3	.80	.82	.80	.82	.82
P4	.59	.61	.73	.77	.62
P5	.72	.83	.72	.83	.83
P6	.83	.91	.87	.91	.72
P7	.89	.87	1.00	1.00	1.00
P8	.42	.45	.44	.45	.45
P9	.41	.43	.42	.43	.43
P10	.36	.43	.42	.43	.43
P11	.33	.73	.57	.73	.54
P12	.43	.69	.44	.69	.69
P13	.44	.54	.54	.54	.54
P14	.65	.76	.65	.76	.76
P15	.56	.57	.56	.57	.57
P16	.99	1.00	1.00	1.00	.86
P17	.84	1.00	1.00	1.00	1.00
P18	.82	.99	.87	1.00	.82
P19	.59	1.00	.66	1.00	.61
P20	.62	.71	.69	.71	.49
P21	1.00	1.00	1.00	1.00	1.00
P22	1.00	1.00	1.00	1.00	.69
P23	1.00	1.00	1.00	1.00	.85
P24	.39	.43	.43	.43	.43
P25	.24	.46	.30	.46	.44
P26	.18	.63	.41	.63	.45
P27	.32	.62	.54	.62	.19
P28	.20	.45	.41	.45	.42
P29	1.00	1.00	1.00	1.00	.66
P30	.41	.85	.50	1.00	.58
P31	.38	1.00	1.00	1.00	1.00
P32	.40	.62	.55	.62	.38
P33	.28	.72	.41	.96	.83
P34	.41	.82	.83	.99	.66
P35	.25	.81	.45	.90	.79
P36	.20	.45	.43	.61	.42
P37	.26	.68	.35	.80	.48
P38	.41	1.00	.51	1.00	.71
P39	.25	.62	.55	.71	.68
P40	.62	1.00	.86	1.00	.68
P41	.42	1.00	.49	1.00	.59
P42	.34	1.00	.46	1.00	1.00
P43	.44	.52	.44	.54	.54
P44	.20	.50	1.00	1.00	1.00
P45	1.00	1.00	1.00	1.00	1.00
P46	.73	1.00	.79	1.00	1.00

IV (the most forgiving one), and model V (which does not consider CPU resources). The models which consider P46 not efficient either do not consider service effort (M-I, M-II), or do not break down SLOC so that SLOC due to reuse could be considered (M-I, M-III). Checking the message entry for P46 in the database provides a clue: it turns out that service time had to be prorated between P45 and P46 with two thirds of that effort going to P46. This could explain the difference in efficiency rating. Indeed, checking the production model results indicate that for P46 the model considers the service effort 380 hours higher than it should have been. Thus, the differences in efficiency between P45 and P46 might be a matter of apportionment of service hours rather than differences in efficiency of code reuse.

Looking at the "weakest of the best" (P30, P33, P34 and P35) provides insight into possible improvements. They only become efficient (P30) or marginally inefficient (P33-P35) in model IV, which considers the most parameters. Comparing model results in models II and IV (difference: model II does not consider service effort), an interesting situation emerges: Model II and IV consider applied efforts appropriate (no need to reduce them). Model II, however, expects significantly more output in the following areas: number of pages of documentation (PDOC, increase by 74%) and code reuse (ReSLOC, increase by 200%). In order to evaluate whether these are the areas in which to become more efficient, one should indeed use model IV, since it directly models service hours, a production variable that measures the effort of the technical publication department. When these factors are explicitly considered, model IV identifies the project as efficient. Thus improvement in this area is only necessary if one wants to improve beyond the current frontier of efficiency. A similar argument can be made for the remaining marginally inefficient projects.

Consider now the "worst" projects for each model. They are for model I: P26, for models II and IV: P9, for model III: P25, and for model V: P28. Unlike the "best" projects, no clear picture emerges, except that they don't do well in any of the models. The suggestions for improvement range from output augmentation alone to a combination of output augmentation and input reduction.

Table 8 shows the detailed production model results for model IV. The first column shows the project index. The second column identifies the efficiency index for the project using model IV. The next three columns show excess inputs for the three input variables TMH, SH, and CPU. The rightmost five columns show desired outputs for each output variable (CHGS, PDOC, NewSLOC, ReSLOC, and ModSLOC). The efficient projects of course, show neither input reduction, nor output augmentation. The marginally inefficient projects (with an efficiency index of between .9 and 1.0) suggest a reduction (and thus increased efficiency) in technical and management hours of effort rather than other inputs. Conversely, at the low end of the spectrum, the suggested input reduction to achieve optimum efficiency concerns service hours and CPU hours rather than technical and management hours.

Considering desired outputs, the projects at the low end of the efficiency scale suggest output augmentation for the output variables between a factor of 2 (for PDOC) and 45 (for ReSLOC). For marginally inefficient projects, this drops to a range of less than 1% to a factor of almost 4 (for code reuse). Output augmentation is fairly evenly distributed, suggesting a need for improvements across the board, rather than for a specific production variable. The next step is to identify other potential factors. For this, we now have to turn to the subjective evaluation variables.

Table 8. Detailed production model results for model IV.

Project	EffInd	TMH	SH	CPU	CHGS	PDOC	NewSLOC	ReSLOC	ModSLOC
P1	.68	2167.72	0	0	2723.25	2484.09	66339.74	2881.23	17667.93
P2	.86	0	2546.64	0	3898.19	3763.61	98963.47	8290.46	26913.27
P3	.82	0	358.32	37.92	1042.47	1588.47	29508.5	59070.87	8210.53
P4	.77	3143.41	0	0	2753.85	2758.00	68761.68	28525.27	18149.60
P5	.83	0	1126.54	44.85	3781.21	3579.5	92644.01	4134.06	25820.17
P6	.91	0	210.37	0	3037.1	2964.5	79914.58	23165.34	15726.70
P7	1.00	0	0	0	255	66	5540	99	0
P8	.45	0	226.14	3.60	609.12	664.5	15603.95	5424.53	3964.67
P9	.43	0	600.22	394.53	4891.34	4961.98	122229.1	12616	33283.13
P10	.43	0	524.58	241.63	3647.7	5465.76	103918.8	58496.16	19931.50
P11	.73	0	601.01	0	421.57	1039.21	20257.03	37247.46	2432.39
P12	.69	0	299.74	11.02	569.77	539.38	13959.15	622.96	3891.55
P13	.54	0	40.57	35.92	1008.42	1416.64	28682.49	9147.91	4344.98
P14	.76	0	1549.67	156.28	5300.11	5019.53	129872.2	5829.12	36186.28
P15	.57	0	121.92	102.92	936.51	904.63	23063.72	2763.59	6429.16
P16	1.00	0	0	0	660	527	12227	1065	1571
P17	1.00	0	0	0	314	511	9568	3822	892
P18	1.00	0	0	0	795	873	18680	6304	7838
P19	1.00	0	0	0	103	136	2451	1099	1947
P20	.71	0	87.90	0	221.36	236.77	5828.16	4823.22	803.61
P21	1.00	0	0	0	300	284	7350	328	2049
P22	1.00	0	0	0	135	61	2052	0	0
P23	1.00	0	0	0	289	163	4921	283	0
P24	.43	0	1275.07	672.71	9727.76	12126.64	263625	69020	50386.99
P25	.46	0	94.86	0	11274.05	11129.44	300959.7	34162.47	78961.53
P26	.63	0	295.98	0	799.72	3132.74	52682.05	187920.90	19150.33
P27	.62	16643.52	2777.66	0	872.58	1226.82	64482.4	11765.52	4697.58
P28	.45	0	48.10	0	1970.54	3085.6	59477.14	115356	16228.05
P29	1.00	0	0	0	1776	1609	70951	10483	0
P30	1.00	0	0	0	1706	2500	194169	14109	28115
P31	1.00	0	0	0	1494	10690	141084	7934	29664
P32	.62	0	2005.21	0	9210.5	21323.25	475397	1082844	37279.69
P33	.96	1775.43	0	0	4338.4	4912.92	114528.7	27597.91	56578.58
P34	.99	7422.83	0	0	1332.95	4828.8	107475	13103.68	17565.03
P35	.90	2207.55	0	0	2528.27	2595.02	66179.78	12675.15	23094.23
P36	.61	4641.74	0	0	1061.04	1513.78	36566.57	8789.17	15728.16
P37	.80	13889.79	0	0	3398.12	6169.1	325737.90	61018.42	40229.68
P38	1.00	0	0	0	955	2100	63861	4399	38186
P39	.71	0	0	0	2075.54	2280.84	53887.76	16230.86	25697.46
P40	1.00	0	0	0	427	950	55289	11060	6063
P41	1.00	0	0	0	323	1200	41552	179016	50441
P42	1.00	0	0	0	341	2235	20859	39495	123663
P43	.54	0	0	162.5	86.86	1306.03	8019.3	108276.20	10984.51
P44	1.00	0	0	0	142	1035	12974	26275	28858
P45	1.00	0	0	0	38	1500	10590	141006	2913
P46	1.00	0	0	0	15	650	0	52026	9421

5.4. Analysis of Subjective Variables

The objective of the analysis of the subjective variables is to investigate which projects were regarded successful. Moreover, the aim is to identify which project factors during development are most closely related to the success of a project. The first step in the analysis is to perform a PCA for all subjective variables (i.e., project and outcome variables) to obtain principal components that include both. The main reason is that the focus of the analysis should be on how the project factors correlate with the outcome of the projects.

The analysis used is a standard principal components analysis with an Orthotran/Varimax transformation to extract orthogonal factors with the highest possible correlations (Gorsuch, 1983; Kachigan, 1986; Ramsay and Silverman, 1997). Factors having an eigenvalue above one are extracted. Table 9 shows the results from the analysis including only the highest loadings. A higher loading means a larger influence on the principal component.

The PCA generated ten factors. It is preferable if the principal components can be interpreted in the context of what the factors stand for. Some of the factors in Table 9 can be interpreted in this way, while others seem primarily a statistical artifact. An example of a factor with an easy interpretation is factor 4, which clearly relates to the experience of the development team. The most interesting factor given the objective of the analysis is the first factor, which includes all outcome variables and three of the project variables. In summary, the first factor includes:
Project variables:

- Stability of requirements (RSTAB)
- Project planning discipline (PROPL)
- Degree to which plans were followed (PPCOMPL)

Outcome variables:

- Agreement of software with requirements (AGGRE)
- Quality of software (QSOFT)
- Quality of design (QDES)
- Quality of documentation (QDOC)
- Timeliness of delivery (TIMELI)
- Smoothness of acceptance testing (ACCTEST)

What this tells us is that only three of the project factors vary the same way as the outcome variables. Stability of requirements, project planning, project execution and all of the subjective ratings of project outcome: agreement with requirements, quality indicators, meeting deadlines and smooth acceptance testing are in the same principal component. To put it another way: low rankings in these three project variables will guarantee low ranking in the outcome variables (an unsuccessful project). The obvious advice is to watch these

Table 9. PCA for all subjective evaluation factors.

Factor	1	2	3	4	5	6	7	8	9	10
COMP		.637								
SCHE										.500
RSTAB	.448									
RQUA								.686		
RDOC			.686							
RREW			.914							
TABI		.497								
TAPP				.863						
TENV				.722						
TSTAB									.917	
MPER										.468
MAPP						.647				
MSTAB						.614				
PROPL	.818									
PPCOMPL	.815									
PROGP								.478		
FORC		.893								
REMET		.898								
DEMET							.562			
TEMET							.729			
TEPLAN							.730			
QA			.655							
CM		.707								
DEVSYS		.867								
DEVTERM		.697								
MEM			.505							
RESTIM		.648								
SHWSS					.912					
EFFTOOL		.921								
AGGRE	.584									
QSOFT	.828									
QDES	.811									
QDOC	.597									
TIMELI	.749									
ACCTEST	.739									

project variables closely to ensure project success. Given the nature of these variables, it is reasonable to assume at least some causality between these project factors and the project outcomes.

The kappa statistic evaluates how good this assumption is: we rank project success based on project variables (model 1) and outcome variables (model 2). Only factors with a loading above 0.7 are used in the ranking to avoid including variables with low affinity to the factor. Then we evaluate the agreement using the kappa statistics. Key in this analysis is to define how many projects are considered successful. We chose 17, because the data indicated a clear ranking difference at this point. The agreement index is now calculated for the two different rankings. The proportions of projects are shown in Table 10. The rows identify whether a project was considered successful or unsuccessful based on ranking of project

Table 10. Projects predicted as successful projects versus those that actually were judged as being successful.

	Successful	Unsuccessful	Sum
Successful	11/46	6/46	17/46
Unsuccessful	6/46	23/46	29/46
Sum	17/46	29/46	

characteristics while the columns classify projects as successful or not based on success variables. This type of matrix shows agreement between both sets of indicators in the diagonal, disagreement in the other fields. It is also called a diffusion matrix.

This results in $\kappa = 0.44$ with a correlation (Spearman) between the project rankings of 0.59. In other words, the agreement is moderate and hence there is a moderate chance that projects with stable requirements, good project planning and where the project plan is followed will become successful projects. Thus, the subjective variables provide insight into which parameters are most important to provide a good basis for a successful project. It is particularly interesting for the development organization to note that stable requirements are important (Wohlin and Ahlgren, 1995). This is an important fact that should be communicated clearly to the customer, whether internal or external. The importance of the project plan stresses the need for good project management. Given these relationships, it appears possible to judge the expected success of a project based on the three project variables.

5.5. Combined Analysis: Objective and Subjective Variables

The next step compares the analysis results of the subjective and objective variables. As before, the projects are ranked based on both approaches (degree of success and efficiency index). The objective is twofold. Firstly, it may help to decide which of the five production models should be used in this particular organization. Secondly, it helps in identifying which of the efficient projects were also regarded as successful. The latter helps to determine which projects represent "best practice." The efficient and successful projects are obviously the projects that we would like to learn from, emulate or use as a baseline when trying to improve.

In this particular case, there were 19 highly efficient projects (for model V, which indicated the largest number of efficient projects of the five production models). 18 of the projects had an efficiency index of 1, the 19th had an efficiency index of .99. We compared these 19 most efficient projects to the 19 most successful ones. The agreement index for all five models is poor and the correlation is fairly low. The highest correlation is between the efficiency indices of model II and the subjective outcome factors. The correlation is 0.31, which is still rather low.

Given that the correlation was highest for model II, we used it to illustrate how the result of the production model may be combined with the analysis of the subjective data. The

Table 11. Successful projects versus efficient projects according to model II.

	Efficient	Inefficient
Successful	9	10
Unsuccessful	10	17

diffusion matrix is shown in Table 11. It shows the number of projects according to their success and efficiency classification.

Nine projects are both efficient and successful. These projects represent best practice and are the ones to emulate. Efficiency is an important company internal attribute. Success is an important external attribute. Thus, the combination of efficiency and success is crucial. The nine projects are: P2, P17, P19, P23, P31, P40, P41, P45 and P46. Seven of these are regarded as efficient for model II in Table 6, i.e., they have an efficiency index of 1.0. Note that P21, which is viewed as efficient by all five production models, is not regarded as a successful project. The rank of P21 is 20 (together with five other projects), i.e., just below the projects that were considered successful. The five highest ranked projects with respect to success are: P17, P11, P8, P31 and P46. Based on this analysis, one possible approach would be to

- Analyze in more detail the nine projects that were both efficient and successful.
- Focus in particular on projects P17, P31 and P46. What makes these both efficient and successful?
- Do an analysis of P21 to find out why it is regarded as efficient, but not really a success.

In general, we need to look into detailed project data after this type of analysis. Some important questions, emphasizing the complementary nature of the production models and the analysis of the subjective variables, include:

- Q1: What do the successful and efficient projects have in common? (Look at size, subjective outcome factors and so on).
- Q2: What can be improved in the successful projects? (Look at projects that are successful, but inefficient. These represent situations where improvement is possible.)
- Q3: Do unsuccessful projects which are efficient have low quality? (This could be one reason why they were efficient.)
- Q4: Could inefficient/unsuccessful projects be targeted for major improvements?

The combined analysis shows that there is much to learn from combining different analysis approaches. Different views are important when assessing and understanding software projects. Increased understanding makes it possible to focus improvement work.

6. Conclusion

Software productivity and project success are complex to analyze. Both depend on a large number of related factors. Reliable quantitative methods paired with engineering judgement appear most promising. This paper described a multi-method approach that utilizes quantitative methods to evaluate efficiency and success.

We used a case study to illustrate and evaluate the combined approach. The results are encouraging, although clearly challenges exist. These must be overcome for this combined analysis to be practically useful:

1. The production models worked well. They clearly identified efficient versus inefficient projects in the same database.

2. Although the project database is considered quite good and one of the most extensive in existence, we believe that the production models could have been more helpful if better quantitative data had been collected. Indeed, the data available was largely disappointing. Specifically, better output measures of quality of all project deliverables, of user satisfaction and the like would have been much more useful in evaluating projects. Such measures exist, but had not been collected in the database. Instead, subjective measures were collected. Of course, subjective measures pose their own problems. In general, subjective project data so far has shown limited value in prediction and evaluation, but the results shown here demonstrate that this need not be so. These subjective measures can provide valuable information when treated properly. Some researchers have questioned the validity of subjective data (Valett and Condon, 1993), and it is true that such problems may exist, e.g. due to low inter-rater reliability (few metrics programs evaluate this). We do not know how valid our data set was, except that a statistical test clearly rejected that the data is random. Because of these issues, we emphasized use of the analysis techniques over the interpretation of the specific results.

3. It is unlikely that all relevant project input and output factors can be described in quantitative terms. This makes it necessary to combine production model analysis with a method that is capable of analyzing subjective factors of the type used in the project database. We suggested and applied a method to do this.

4. In the course of the analysis, we identified several projects for further analysis.

Due to the limited knowledge about the projects analyzed, interpretation of some of the results was not possible. This is not a deficiency of the models, rather it highlights the indispensable need for analyst knowledge to complement the quantitative analysis. No model or collection of data should be expected to interpret itself. Only expert knowledge can do that.

Acknowledgments

We are grateful to Chris Graves and Armin Roeseler for their assistance with prior modeling efforts. It set the infrastructure for part of this analysis. We also thank Frank McGarry for giving us access to the NASA-SEL data which we used as a case study to illustrate our approach.

Notes

1. We are well aware of the problematic nature of this metric. Indeed, its inherent problems motivated our use of multi-input, multi-output production models.
2. Notice the formal requirement that the observations on X_j and Y_j have non-zero elements, that is, they are defined to be strictly greater than zero. To comply with this requirement, it has been suggested to add a small value (e.g., 0.01) to observations which have legitimate zero values. The introduction of these (small) constants will not appreciably alter the solutions obtained (Bessent et al., 1980).
3. Throughout, optimal values of variables are denoted with an asterisk.
4. In general, *augmentation* is the desired direction for outputs, and *reduction* (or *diminution*) is the desired direction for inputs.
5. The $\Delta x_{i,k}$ are also called *input slack*.
6. Mathematical extensions that could be included in the model (e.g. Wagner, 1969; Hillier and Lieberman, 1967) relate to integer linear programming and mixed integer programming.

References

Altman, D. 1991. *Practical Statistics for Medical Research*, Chapman-Hall.

Banker, R. 1984. Estimating most productive scale size using data envelope analysis. *European Journal of Operational Research* 17: 35–44.

Banker, R., Datar, S., and Kemerer, C. 1991. A model to evaluate variables impacting the productivity of software maintenance projects. *Management Science* 37(1): 1–181.

Basili, V., Zelkovitz, M., McGarry, F., Page, J., Waligora, S., and Pajerski, R. 1995. SEL's software process-improvement program. *IEEE Software* November: 83–87.

Bessent, M., and Bessent, E. W. 1980. Determining comparative efficiency of schools through data envelopment analysis. *Educational Administration Quarterly* 16(2): 57-750.

Briand, L., El Emam, K., and Morasca, S. 1996. On the application of measurement theory in software engineering. *Journal of Empirical Software Engineering* 1(1): 61–886.

Charnes, A., Cooper, W. W., and Rhodes, E. 1978. Measuring the efficiency of decision making units. *European Journal of Operational Research* 2(6): 429–449. See also Corrections. *op. cit.*, 1979. 3(4): 339.

Charnes, A., and Cooper, W. W. 1985. Preface to topics in data envelopment analysis. *Annals of Operations Research* 2: 59–94.

El Emam, K. 1998. Benchmarking Kappa for Software Process Assessment Reliability Studies. International Software Engineering Research Network Technical Report, ISERN-98-02.

Gorsuch, R. L. 1983. *Factor Analysis, 2nd ed*. Hillsdale, New Jersey: Laurence Erlbaum Associates.

Hillier, F. S., and Lieberman, G. J., 1967. *Introduction to Operations Research*. San Francisco, CA: Holden-Day, Inc.

Kachigan, S. K. 1986. *Statistical Analysis: An Interdisciplinary Introduction to Univariate and Multivariate Methods*. New York: Radius Press.

Khoshgoftaar, T. M., and Lanning, D. L. 1994. Are the principal components of software complexity stable across software products? *Procs. International Symposium on Software Metrics*, 61–72, October 1994, London, United Kingdom.

Khoshgoftaar, T. M., Allen, E. B., Goel, N., Nandi, A., and McMullan, J. 1996. Detection of software modules with high debug code churn in a very large legacy system. *Procs. International Symposium on Software Reliability Engineering, ISSRE '96*. White Plains, New York, USA, 364–371.

NASA-SEL. 1992. *Software Engineering Laboratory Database Organization and Users Guide, Revision 2; NASA-SEL Series, SEL-89-201*. Greenbelt, MD: Goddard Space Flight Center. Oct.

Norman, M., and Stoker, B. 1991. *Data Envelopment Analysis*. New York, NY: Wiley and Sons.

Nunamaker, T. R. 1985. Using data envelopment analysis to measure the efficiency of non-profit organizations: a Critical evaluation. *Managerial and Decision Economics* 8(1): 50–58.

Ohlsson, M. C., and Wohlin, C. 1998. Identification of green, yellow and red legacy components. *Procs. International Conference on Software Maintenance, ICSM '98*. Washington, D.C., USA, 6–15.

Ramsay, J. O., and Silverman, B. W. 1997. *Functional Data Analysis*, New York: Springer Verlag.

Roeseler, A. 1991. *A production-based approach to performance evaluation of computing technology*. PhD Thesis. Illinois Institute of Technology.

Roeseler, A., and von Mayrhauser, A. 1992. A production-based approach to performance evaluation of computing technology. *Journal of Systems and Software*.

Valett, J., and Condon, S. E. 1993. The (mis)use of subjective process measures in software engineering. *Proceedings of the Eighteenth Annual Software Engineering Workshop. Software Engineering Laboratory Series SEL-93-003*. NASA Goddard Space Flight Center, 161–168.

von Mayrhauser, A., and Roeseler, A. 1993. Software process assessment and improvement using production models. *Procs. COMPSAC '93*. Phoenix, AZ.

von Mayrhauser, A., and Roeseler, A. 1993. Assessing efficiency of software production for NASA–SEL Data. *Procs. 18th Annual Software Engineering Workshop, Software Engineering Laboratory Series SEL-93-003*, 144–158.

Wagner, H. M. 1969. *Principles of Operations Research*. Prentice-Hall International Series in Management. Englewood Cliffs, NJ: Prentice-Hall.

Wohlin, C., and Ahlgren, M. 1995. Soft factors and their impact on time to market. *Software Quality Journal* No. 4, pp. 189–205.

Wohlin, C., Xie, M., and Ahlgren, M. 1995. Reducing ttime to market through optimization with respect to soft factors. *Procs. IEEE Annual International Engineering Management Conference*. Singapore, 116–121.

Dr. Anneliese von Mayrhauser is a Professor at Colorado State University. She is also Director of the Colorado Advanced Software Institute, a consortium of businesses and Colorado research universities that supports collaborative Technology Transfer Research projects.

Dr. von Mayrhauser is the author of a text book and numerous articles in the area of Software Engineering, particularly software testing and maintenance.

Dr. von Mayrhauser holds an MS and PhD from Duke University and a Dipl.-Inf. from the Technical University of Karlsruhe. She is currently Editor in Chief of the IEEE Transactions on Software Engineering. She has also served on several other editorial boards including the IEEE Transactions on Reliability, the Empirical Software Engineering Journal and the Journal of Software Maintenance.

She contributes to many conferences in various capacities including general or program chair, or as a member of the steering or program committee.

Dr. Wohlin is a professor of software engineering at the Department of Communication Systems, Lund University. Prior to this, he was a professor of software engineering at Linköping University. He has a Ph.D. in Communication Systems from Lund University. He is the founder and director of the research group in software engineering consisting of 12 people at Lund University. Dr. Wohlin is responsible for the education and research in the area of software engineering at the department. He is also vice head of the department. His research interests include empirical methods in software engineering, software metrics, software quality and systematic improvement in software engineering. Claes Wohlin is the principal author of the book "Experimentation in Software Engineering—An Introduction" published by Kluwer Academic Publishers in 1999. He is on the editorial board of the journal of Information and Software Technology. Dr. Wohlin is the program chair for the IEEE International Symposium on Software Metrics in London, UK in 2001.

Magnus C. Ohisson is a doctoral student at the Department of Communication Systems, Lund University. He received his Master of Science in Software Engineering from University of Karlskrona/Ronneby in 1996, and his Licentiate in Engineering from the Department of Communication Systems, Lund University, in 1999. His main research areas are the use of experience to build prediction models and models for process improvement. Most of the work has been carried out within effort prediction and maintenance. Magnus C. Ohisson is a co-author of the book "Experimentation in Software Engineering—An Introduction" published by Kluwer Academic Publishers in 1999.