Handelshøyskolen BI

**Discussion Paper**
**4/2002**

# Identifying High Performance ERP Projects

Erik Stensrud
*The Norwegian School of Management*
erik.stensrud@bi.no
&
*Stensrud Consulting*
erik.stensrud@ieee.org

Ingunn Myrtveit
*The Norwegian School of Management*
ingunn.myrtveit@bi.no

## Abstract

*Learning from high performance projects is crucial for software process improvement. Therefore, we need to identify outstanding projects that may serve as role models. It is common to measure productivity as an indicator of performance. It is vital that productivity measurements deal correctly with variable returns to scale and multivariate data. Software projects generally exhibit variable returns to scale, and the output from ERP projects is multivariate. We propose to use Data Envelopment Analysis Variable Returns to Scale (DEA VRS) to measure the productivity of software projects. DEA VRS fulfils the two requirements stated above, and to our knowledge, it is the only method complying with them. The results from this empirical study of 30 ERP projects extracted from a benchmarking database in Accenture identified six projects as potential role models. These projects deserve to be studied and probably copied as part of a software process improvement initiative. The results also suggest that there is a 50% potential for productivity improvement, on average. Finally, the results support the assumption of variable returns to scale in ERP projects. We recommend DEA VRS be used as the default technique for appropriate productivity comparisons of software projects. Used together with methods for hypothesis testing, DEA VRS is also a useful technique for assessing the effect of alleged process improvements.*

## Index Terms

Software process improvement, benchmarking, best practice identification, software project management, multivariate productivity measurements, data envelopment analysis (DEA), software development, enterprise resource planning (ERP), software metrics, economies of scale, variable returns to scale.

## 1. Introduction

Learning from high performance projects is crucial for software process improvement. Therefore, we need to identify outstanding projects that may serve as role models. A minimum prerequisite for identifying these best practice projects, is the ability to *measure* the performance. If you cannot measure it, you cannot possibly know which projects are best, and you cannot know whether you have improved or not. Also, if you are able to identify the best projects, they may serve as role models guiding you on *how* to improve. For practitioners, identifying and studying the best practice projects is an invaluable source of learning. Last, but not least, by measuring project performance, you create incentives that likely will yield higher performance. Indeed, Weinberg [37] demonstrated many years ago that the proverb "You get what you measure" also is highly valid in the software engineering field.

In addition to identifying the best practice projects, several stakeholders are interested in the related problem of benchmarking the projects. (In this context, benchmarking means to measure the project performance against some established performance standard, or alternatively, against an observed best practice frontier.)

As practitioners, we experience an increasing demand from our c*ustomers* that performance benchmarks of past performance be included in proposals. Therefore, *consultants* must provide benchmarks to stay competitive. *Organisations* use benchmarks internally as input to compensation schemes and promotions, and thus needs to identify best performers. Finally, as already stated, p*roject managers* and *methodologists* need to identify best practice processes and technologies to improve project methodologies and software processes.

It is not trivial to correctly identify the outstanding, best performing software projects. First, we need to establish criteria for what we actually mean by qualitative words like "outstanding", "high performance", "best", and so on, and then we must find appropriate quantifiable measures. Next, it is vital that the comparisons of individual software projects deal correctly with variable returns to scale and multivariate data because it is likely that software projects exhibit variable returns to scale, in general, and in addition, the output from ERP projects is multivariate. (ERP projects are a subclass of software projects.)

In this paper, we measure the *productivity* and use it as a performance indicator. In other words, we use the productivity as the criterion to judge software projects as "high performance" or "best". For software projects, productivity is relatively easy to measure. Also, it is a common performance indicator in software engineering. It is, however, not unproblematic to reduce the task of measuring performance to the subtask of measuring solely productivity of software projects. This issue is discussed in section 7.1.

The most widely applied productivity model in software engineering (See e.g.[12] [15] [22] [26] [9]) is the following univariate, constant returns to scale (CRS), model (P=productivity, x=input, y=output):

$$P = \frac{y}{x} \qquad (1)$$

Equation 1. A univariate, CRS (linear) productivity model

Common output measures (i.e. y) in software projects are source lines of code (SLOC), function points (FP) or object points, and the usual input measure (i.e. x) is effort, e.g. the number of personmonths (PM). So, Equation 1 states that the productivity equals the number of FP developed per PM. That is, the more FP per PM, the higher the productivity. Equation 1 therefore seems like a reasonable productivity model.

There is, however, one serious drawback with the productivity model in Equation 1. The productivity model (Equation 1) assumes *constant returns to scale* (CRS) in software projects. In other words, CRS assumes a *linear* relationship between input and output. This assumption is inconsistent with the assumptions made by important cost estimation models like COCOMO 1.0 or 2.0 [13][14]. COCOMO assumes the contrary, namely that software projects exhibit *variable returns to scale* (VRS). That is, they assume a *non-linear* relationship between input and output. Provided VRS cost models like COCOMO are right in their assumptions, CRS productivity models like Equation 1 would simply pick the smallest project as the most productive project, ultimately misleading us to draw erroneous conclusions regarding which project is the most productive. Cost estimation models like COCOMO generally have the following form (P=productivity, x=effort, y=FP or SLOC, B>1):

$$x = \frac{1}{P} y^{B} \qquad (2)$$

Equation 2. A VRS (non-linear) cost model

When B>1 (as in COCOMO), Equation 2 states that software projects have *decreasing* returns to scale (DRS). DRS is a special case of VRS. On the other hand, Equation 2 would describe an *increasing* returns to scale (IRS) cost model if B<1. IRS is also a special case of VRS. When software projects exhibit VRS, it means they might be either IRS or DRS or both. Finally, if B=1, Equation 2 describes a CRS cost model (Figure 1). (In the paper, we use the term *economies of scale* as a synonym to IRS and *diseconomies of scale* as a synonym to DRS and *(dis)economies of scale* as a synonym to VRS, and VRS to mean either IRS, DRS or both.)

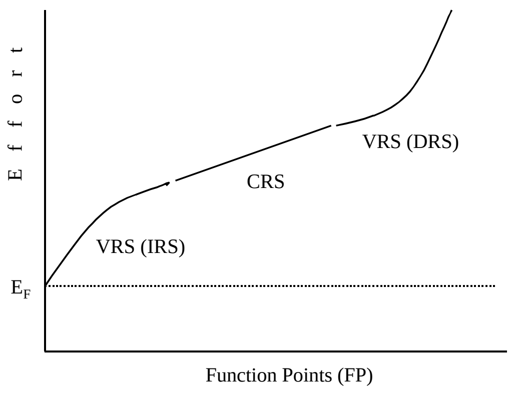

Figure 1. CRS and VRS models

Small and large software projects likely exhibit VRS (IRS and DRS, respectively) whereas medium software projects probably exhibit CRS. To see this, it is useful to divide software development into two parts, *application* development and *technical infrastructure* (TI) development. The application is the part of the system containing the user functionality. The technical infrastructure consists of hardware, network, operating system, compilers, editors, database management system, transaction processing monitors, window managers, programming standards, and other third party software needed to support the application.

Function points measure mostly application size, but account inadequately for TI size. Therefore, using FP as the output measure and effort as input, small projects likely exhibit IRS because they incur a relatively large TI development effort compared to the application development effort. A small project by definition develops few FP. Therefore, the average effort per FP is high since the TI effort is distributed across a low number of FP. Therefore, when small projects develop more FP, the TI effort is distributed across more FP, and consequently, the average effort per FP decreases.

At the other end of the spectrum, large projects likely exhibit DRS because they incur increasing overhead effort (Larger project teams and more modules needed to be integrated will result in more coordination effort.) Therefore, it is reasonable to expect that the average effort per FP increases. Finally, medium projects probably are CRS (i.e. they have a linear relationship between FP and effort) because the IRS and DRS effects cancel each other out.

In addition to the CRS/VRS issue that likely is general to the broad class of software projects, ERP projects (a subclass of the class of software projects) produce *multivariate* outputs. Therefore, *univariate* productivity models like Equation 1 are inapplicable since they can take one input and one output, only (e.g. input=effort and output=FP). As for ERP[1] projects, the output is a multivariate measure (output={Users, Sites, Plants, Companies, Interfaces, EDI, Conversions, Modifications, Reports, Modules}, see Table 1). Hence, the multivariate ERP output measure differs from custom software development (CSD) projects where the output often is a univariate measure like FP or SLOC.

The output of ERP projects is, and has to be, multidimensional since ERP projects are part of *business transformation* initiatives and not stand alone CSD projects. This implies that the projects not only deliver developed software but also deliver reengineered business processes and organisational structures. The business reengineering is performed partly to improve business performance and partly necessitated by the ERP package because the functionality of a package to some extent dictates how you have to do your business. Of course, one could also perform business process reengineering activities in connection with CSD projects. The difference is that you do not *have to* since, in a CSD project, you can always customise the functionality to an existing organisation and the way it does its work. Using an ERP software package, the functionality is largely given (unless you rewrite the existing functionality). Thus, as a user you must adapt your work processes so that they align with the given software functionality. Thus, there is always *some* business process reengineering activities carried out in these ERP projects.

To appropriately measure and compare the productivity of individual ERP projects exhibiting VRS and multivariate outputs, we propose to use *Data Envelopment Analysis, Variable Returns to Scale* (DEA VRS) to measure the productivity of software projects in general, and of ERP projects, in particular. DEA VRS ensures that large projects are compared with other large projects and small projects with small projects. Furthermore, DEA is suitable for productivity comparisons of ERP projects because it handles multivariate inputs and outputs. In fact, as far as we know, DEA is the *only* method complying with these two requirements that we consider crucial to perform correct productivity assessments in software engineering.

The paper is organised as follows. Section 2 presents the issue of measuring productivity in general. Section 3 presents DEA. The presentation emphasises the strengths as well as the limitations of DEA in the context of identifying best practice ERP projects. We believe it is the first time DEA is used to analyse ERP projects. Furthermore, we believe it is the first time DEA is used to test hypotheses and where significance levels are reported when analysing software projects[2]. Section 4 presents related work using DEA to analyse software projects. Many papers have been published on DEA[3]. However, to our knowledge, only four papers have used DEA to analyse software projects [5][7][8][28]. It is unfortunate that DEA VRS has not gained more widespread use in the software engineering community since productivity assessments are widely conducted and reported in research studies, using CRS productivity models e.g.[12] [15][22] [26][9] that probably are inappropriate. It seems inappropriate (and meaningless) to compare the productivity of a small project with the productivity of a large project e.g. compare a one-person project with a 100-person project if the aim is to identify appropriate role models as sources of learning for software process improvement.

We show that the four papers using DEA to analyse software projects partly suffer from methodological flaws and partly use DEA where simpler methods could have been used. Section 5 describes the ERP data used in the analysis. Section 6 presents the results of analysing the Albrecht-Gaffney CSD data set using DEA as well as the results analysing the ERP data with DEA. The main purpose of analysing the Albrecht-Gaffney data set is to provide an intuitive example (for *univariate* cases) of the use of DEA CRS and VRS.

[1] In earlier papers we have preferred the term PER (package-enabled reengineering) to ERP (enterprise resource planning). Actually, PER *projects* implement ERP *systems*. However, the term ERP has become more established in magazines, e.g. in Communications of the ACM, April 2000 issue. Therefore, we have opted for the term ERP in this paper.

[2] However, some results were published in an earlier version at METRICS'99 [27].

[3] We found 285 hits in the INSPEC Electronics & Computing database 1989 - Oct 97 using the search term «data envelopment analysis» of which a large majority were in operational research journals.

Section 7 discusses some important assumptions underlying performance and productivity assessments regardless of whether DEA VRS or other CRS productivity models like Equation 1 are used. Also, some issues that are particular to DEA are discussed in this section. Section 8 concludes recommending that DEA VRS should be adopted as the default productivity model in software engineering for comparison (benchmarking) of individual projects and identification of the most productive projects.

## 2. MEASURING PRODUCTIVITY

As stated in the Introduction, the productivity (P) is generally defined as the output (y) over input (x) ratio in the *univariate CRS* case (as in Equation 1). In software engineering, we are accustomed to depicting the output (FP or SLOC) along the horizontal axis and input (effort) along the vertical axis (as in Figure 1). In economics in general, and in DEA in particular, the axes are usually switched. We have adopted the DEA convention in the figures in this paper since the paper is on DEA (despite being in a software engineering journal). We observe, therefore, that the axes in Figure 1 are switched compared with the axes in Figure 3.

Using the DEA convention, we have plotted the Albrecht-Gaffney [2] data set in Figure 2 (see also Table 2) where we observe that project 23 appears to have the highest productivity when we apply a *univariate CRS* model (P=199/0.5=398). Alternatively, we may present the productivity results on a normalised scale, i.e. a scale from zero to one, by dividing all numbers with the highest, $P_{MAX}$. For the Albrecht-Gaffney data set $P_{MAX}$ =398 (i.e. the productivity of project 23). Project 23 thus has a normalised productivity equal to 1. Using this normalised CRS productivity scale, the productivity of e.g. project 20 *relative* to project 23, is:

$$\frac{P_{20}}{P_{MAX}} = \frac{\frac{1572}{61.2}}{\frac{199}{0.5}} = \frac{25.7}{398} = 0.06$$

Equation 3: A normalised, univariate, CRS productivity measure

We observe that project 20 appears extremely unproductive in the CRS scheme. Inspecting Figure 2, it seems likely that the software projects in the Albrecht-Gaffney data set exhibit diseconomies of scale (VRS of the DRS type), and therefore, CRS productivity models are inappropriate for comparing e.g. a large project like number 20 with a small project like number 23. Project 20 is a 61 PM project and therefore likely is a multi-person project with a team size of, say, 5-10 developers (The largest projects in the Albrecht-Gaffney data set, projects 1 and 2, likely have 10+ developers). As opposed to these large projects, project 23 is a small, two workweeks, one-person project probably with no overhead costs and likely insignificant fixed costs (because in two weeks it would be impossible to install the technical infrastructure and thereafter develop and test an application of 398 FP). Figure 2 thus clearly illustrates how misleading a simple CRS productivity model can be for productivity comparisons unless it takes diseconomies of scale (VRS) into account. The plot of the Albrecht-Gaffney data set reveals a pattern clearly suggesting VRS (of type DRS or diseconomies of scale). Therefore, it is not surprising that the smallest project (project 23) is deemed the most productive. Thus, an obvious objection against using a CRS productivity model is that it is not reasonable to compare a small (0.5 workmonths) project with a large (61 workmonths) project. Still, we observe that this is routinely done in software engineering productivity studies (e.g. [15][22][14][26][9]). In general, it would seem more reasonable to compare a project with other projects of similar size. That is, it seems more appropriate to apply a VRS model comparing the productivity of small with small and big with big, since there are good reasons, analytical as well as empirical, to believe there are economies as well as diseconomies of scale in software projects i.e. VRS [5][8].

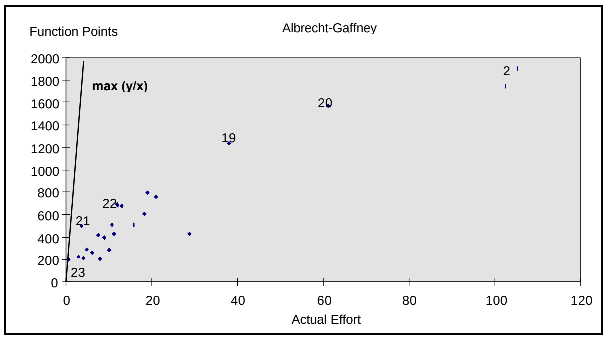

Figure 2. Benchmarking Albrecht-Gaffney projects assuming constant returns to scale (CRS). The straight line is the CRS frontier.

If we assume VRS rather than CRS, one pragmatic approach is to define a *non-parametric best practice frontier* in this two-dimensional space. This idea is illustrated in Figure 3 where the dotted line represents the CRS best practice frontier, and the solid line represents the VRS best practice frontier. In this VRS scheme, project 23 is no longer the only fully productive project. Rather, in the VRS scheme, project 20 also is on the front (P=1.0) in stead of being highly unproductive (P=0.06) in the CRS scheme. Similarly, e.g. project 10 is benchmarked against the line segment between projects 19 and 22 in stead of against the dotted CRS line where project 23 is the only reference. Intuitively, a VRS model (comparing small with small and large with large) seems more reasonable for the Albrecht-Gaffney projects. Also, a VRS productivity model would definitely be more appropriate than a CRS productivity model for the COCOMO data since the cost model assumes that software projects comply with a VRS model.

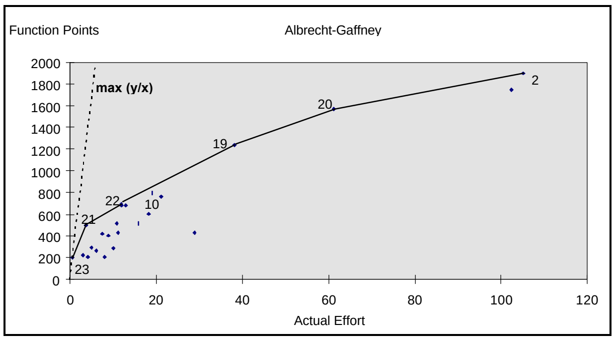

Figure 3. Benchmarking Albrecht-Gaffney projects assuming VRS and using a non-parametric frontier. The dotted straight line is the CRS frontier. The broken line is the VRS frontier.

Next, proceeding from the *univariate* Albrecht-Gaffney CSD data set to *multivariate* ERP projects, we observe that univariate productivity models like Equation 1 can not be used. In Table 1, all the variables {Users, Sites, Companies, etc.} define the output. In other words, this multivariate ERP output is analogous to the univariate FP (or SLOC) output used in CSD projects. In this multivariate case, it seems reasonable to construct a productivity model similar to Equation 4 (Note that it is still CRS rather than VRS, though):

$$P = \frac{\sum_{j=1}^{n} a_j Y_j}{\sum_{k=1}^{m} b_k X_k}$$

Equation 4: A multidimensional CRS productivity model

In Equation 4, $a_j$ and $b_k$ are weights reflecting the relative importance of the different outputs and inputs, respectively. The normalised productivity can still be defined in a way similar to Equation 2, i.e. $P/P_{MAX}$.

We observe that although Equation 4 improves over Equation 1 (and Equation 3) in that it allows for multivariate productivity models, it still does not handle VRS, but only CRS. In this paper we therefore propose to use DEA to measure productivity and benchmark software projects (including ERP projects) because DEA addresses multivariate CRS as well as *multivariate VRS* productivity measurements. In other words, DEA tackles the problem of comparing projects of similar size with each other in a normalised, multivariate space.

## 3. Data Envelopment Analysis

The initial publication on Data Envelopment Analysis (DEA) method is credited to Charnes, Cooper and Rhodes [18] handling CRS (constant returns to scale), only. Afriat [1] laid the foundations for VRS, which later have been enhanced by several authors including Banker, Charnes and Cooper [6] and Førsund and Hjalmarson [21].

When performing DEA, the first step is to decide whether to use a CRS or a VRS model since DEA gives you the choice. For software projects in general, and ERP projects in particular, it is prudent (and it makes sense, see the Introduction and Figure 1) to assume VRS. The VRS assumption is supported by e.g. Boehm [13], Brooks [16] and Banker, Chang and Kemerer [5]. For example, the VRS assumption is explicitly stated in cost models like COCOMO 1.0 and 2.0 [13][14] where the exponent of the size variable is greater than one ($x=Ay^B$, where B>1, x=Effort, A includes a selection of cost drivers, and y=FP, SLOC, or object points). Thus, in software engineering and software cost estimation, it is not controversial to assume that software projects exhibit VRS.

Technically, there are two alternative algorithms to calculate the VRS *efficiency* using DEA. (Using the DEA terminology, we use the term *efficiency* in stead of the term *productivity*. In the paper, they are used as synonyms.) We may either use an *input reducing* efficiency or alternatively an *output increasing* efficiency measure. These two measures are illustrated in Figure 3 for project C where AB/AC and EC/ED are the input decreasing and output increasing efficiencies, respectively. Both are reasonable approaches in the context of software engineering. We can either measure how much less effort that could have been used to produce the same amount of project output (keeping project size constant), or alternatively, we can measure how much more project output that could have been produced with the same amount of effort (keeping project effort constant). In this paper we use the input reducing efficiency measure to illustrate the DEA method.

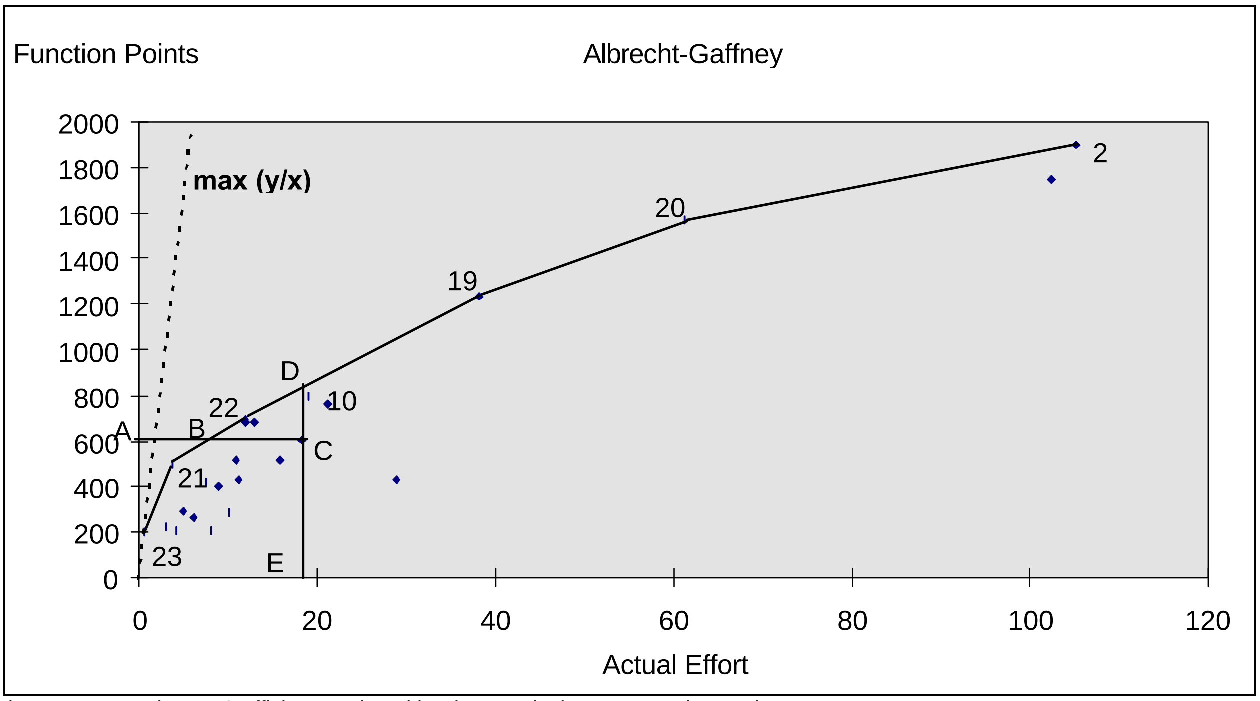

Figure 4: Measuring VRS efficiency using either input reducing or output increasing measures.

Using project C as example, we attempt to find the *minimal* effort required to produce the same amount of output as C produces. That is, we ask how much effort it would take for a best practice project to produce just as much output as C. This minimal effort is the effort at the point B, which is a linear combination of the two frontier projects 21 and 22. These latter are termed *reference projects*. Thus, the idea is to move horizontally from C and towards the left until we hit the line segment at B. This is a minimisation problem, which can be solved using linear programming.

The formal problem thus becomes to minimise the objective function:

$$E_i = \min q_i \tag{1}$$

subject to the constraints:

$$\sum_j l_{ij} Y_{kj} \geq Y_{ki}, \forall k \tag{1.1}$$

$$q_i X_{mi} \geq \sum_j l_{ij} X_{mj}, \forall m \tag{1.2}$$

$$\sum_j l_{ij} = 1 \tag{1.3}$$

$$l_{ij} \geq 0, \forall j \tag{1.4}$$

The constraint in (1.3) is the VRS constraint, and furthermore:

- $E_i$ - is the efficiency score for observation i
- $\theta_i$ – is the efficiency score variable to be determined for observation i
- $\lambda_i$ – are the weights to be determined for observation i
- $X_{mi}$, $Y_{ki}$ – are inputs and outputs of observation i - is the current observation
- j - is all the other observations with which observation i is compared
- m - is the number of inputs, in our case effort, only
- k - is the number of outputs, i.e. the multidimensional size metric for the ERP projects

The technicalities for solving the DEA problem in a computationally efficient manner on a computer is beyond the scope of this paper and is thus not discussed here. The algorithmic issues in DEA are, however, similar to the issues to consider in linear programming.

## 4. Related work

Banker and Kemerer [8] use DEA to test whether software projects exhibit VRS and to identify the optimal project size with respect to maximising productivity. They apply the DEA CRS model on eight univariate (single input - single output) data sets, including the Albrecht-Gaffney data set [2]. Regarding the Albrecht-Gaffney data set, they find that the most productive project is project 23 in Figure 1 (199 function points, 0.5 workmonths. See Table 2). The merit of Banker and Kemerer is that they introduce DEA in software engineering.

However, for this trivial univariate CRS case, we observe that the same result could have been found with simpler methods than DEA such as visual inspection of the scatter plot in Figure 1 or by calculating all the simple y/x ratios and then sorting them in a spreadsheet.

Banker, Chang and Kemerer's paper [5] is an extension of [8] employing the DEA-based F-test of Banker and Chang to verify their previous results of VRS in software projects.

Banker, Datar and Kemerer [7] employ a variant of basic DEA CRS that is extended in two orthogonal directions. The first extension is called Stochastic DEA (SDEA). SDEA is stochastic in the sense that in addition to productivity related deviations, it also allows for the impact of random errors. The second extension extends DEA to analyse the effects of several alleged productivity factors such as using or not using «peer reviews». This doubly extended DEA model is used to evaluate the effect of five productivity factors on 65 software maintenance projects.

The idea behind a stochastic DEA is conceptually appealing, and we acknowledge Banker's, Datar's and Kemerer's work on this issue. A stochastic DEA that can incorporate random errors would certainly be welcome by statisticians, and it would also improve our faith in the results drawn from DEA. Stochastic DEA remains, however, a formidable challenge. We do not see how the problem of distinguishing model specification errors and measurement errors from inefficiency is solved in Banker et al.'s paper. Concerning model specification errors, we still find it more intuitive to perform a sensitivity analysis on the model specification and study the effects on the frontier as well as on the individual and average efficiency scores. Concerning measurement errors, we find it more intuitive to remove one project at a time from the frontier and again study the effects on the frontier as well as on the individual and average efficiency scores.

Parkan, Lam and Hang [28] use DEA to measure the performance of individual projects where DEA is used as a part of an organisation's reward structure. They apply the VRS model on one data set with eight projects. The data set has four inputs and one output. However, they have not commented on the fact that they use a VRS model, and why. With four inputs and only eight projects, three out of the eight projects are efficient. The robustness of this result is not commented. Few projects and many dimensions will result in too many projects being on the frontier, making it meaningless to identify role models.

Below follows a summary of DEA papers in IT areas other than software engineering. We consider this work as somewhat remotely related to software engineering. Nevertheless, we have included them to provide a broader account of related work due to the scarcity of work reported in software engineering.

Fisher and Sun [20] use DEA to evaluate the individual performance of 22 e-mail packages using the VRS model. The data set has five inputs and four outputs. Using all inputs and outputs they find four efficient e-mail packages. One project is in the reference set for all but two of the 22 packages. Fisher and Sun do not comment on the rationale for choosing a VRS rather than a CRS model. Also, they do not comment on why one single package serves as reference for almost all other packages, nor do they do a sensitivity analysis by removing this package which obviously is extreme in one or more of the output dimensions.

Thore, Phillips, Ruefli and Yue [34] use DEA to rank the efficiency of 44 U.S. computer companies using six inputs and three outputs. They find that 11 companies are efficient using both CRS and VRS models. The robustness of this result is not discussed. Sensitivity analysis is not done.

Mahmood [24] uses DEA to evaluate organisational efficiency of IT investments using a data set with 81 firms and eight inputs and ten outputs per firm. The results indicate that two-thirds of the firms are efficient. It is not documented whether a CRS or a VRS model is used. However, using any of these two models, there will likely be many firms on the frontier because of the large number of dimensions. The robustness of the results is not discussed. Mahmood also compares the efficient group of firms with the non-efficient group based on differences in means but without testing the significance of these results.

Doyle and Green [19] use DEA to benchmark 22 microcomputers using one input and four outputs. The merit of their paper is in providing a good presentation of DEA and a comparison of DEA with regression analysis.

In summary, previous studies suffer from several major flaws.

- They use DEA for univariate CRS data sets where we have shown that simpler productivity models (like Equation 1) could have been used.
- They use CRS models where a VRS model would have been more appropriate.
- When using a VRS model in multivariate data sets, it is applied to too small data sets compared to the number of variables. In such a case, the results are not particularly informative as too many projects will be on the DEA frontier.
- Sensitivity analysis is not a routine part of DEA analysis in empirical software engineering papers. Sensitivity analysis of outliers as well as of model specification must be done when using DEA because productivity comparisons, not limited to DEA, are extremely sensitive to outliers and model specification, in general.
- Last, but not least, the rationale for applying either a CRS or a VRS model is not reported.

## 5. ERP DATA

The original data set used for this validation consists of 48 completed ERP projects. All the ERP projects in the sample implement the same ERP software package, SAP R/3, i.e. it is a homogeneous data set. The data have been gathered since 1990, and it is an ongoing effort. All the projects are industrial projects spanning from 100 to 20.000 workdays, and there are ten output factors. All the ten variables in Table 1 are therefore candidate *output* metrics. The obvious *input* metric is Effort (not reported in Table 1). These 11 (ten outputs, one input) metrics constitute the intra-organisational benchmarking standard[1] in Accenture (formerly Andersen Consulting). A more detailed description and explanation of these metrics is beyond the scope of this paper since the main focus is on the DEA technique rather than on the metrics. Readers interested in the metrics are referred to [31][32].

**Table 1: Descriptive statistics for the ten ERP outputs**

| Variable | N | Mean | Min | Max |
|---|---|---|---|---|
| Users | 48 | 346.5 | 7 | 2000 |
| Sites | 48 | 10.25 | 0 | 98 |
| Plants | 48 | 7.35 | 0 | 98 |
| Companies | 48 | 2.833 | 1 | 35 |
| Interfaces | 46 | 13.07 | 0 | 50 |
| EDI | 35 | 1.857 | 0 | 10 |
| Conversions | 37 | 18.38 | 1 | 93 |
| Modifications | 39 | 9.74 | 0 | 30 |
| Reports | 44 | 44.16 | 0 | 100 |
| ModulNo | 48 | 4.500 | 1 | 8 |

We observe that we have a relatively large number of output factors (10) compared to the number of projects (48). In addition, there were missing values for some of the observations. Therefore, we had to reduce the number of output factors, and at the same time use variables giving us the largest possible sample. We primarily used expert knowledge to determine which of the ten outputs to include in the model. Best subset regression analysis [11][25] was used to assist the expert in this selection process. We landed on a model with three outputs (Users, EDI, Conversions) and one input (Effort). This resulted in a final usable data set of 30 observations.

There were no specialised units in the data set, i.e. projects that e.g. have a high number of Users and zero in EDI as well as Conversions. (Specialised units will tend to be on the front.)

## 6. RESULTS

In this section, we provide the results for the Albrecht-Gaffney data set as well as for the ERP data set. We have included the Albrecht-Gaffney CSD data set mostly because it is instructive to discuss the results of the DEA method using a univariate data set, and a data set that presumably is familiar to a software engineering audience. Also, it is interesting to compare our VRS frontier results to Banker and Kemerer's [8] CRS frontier results since they also analysed the Albrecht-Gaffney data set. Also, it is interesting to compare the DEA measures with the more familiar univariate, non-normalised, CRS productivity measure. We have therefore reported three different productivity measures:

- Univariate non-normalised CRS productivity (P)
- DEA CRS efficiency ($E_{CRS}$)
- DEA VRS efficiency ($E_{VRS}$)

### 6.1 Albrecht-Gaffney Results

**Table 2: Efficiency results for Albrecht-Gaffney data set**

| Project ID | Actual Effort | Function Points | P* | $E_{CRS}$ | $E_{VRS}$ |
|---|---|---|---|---|---|
| 1 | 102.4 | 1750 | 17 | 0.04 | 0.83 |
| 2 | 105.2 | 1902 | 18 | 0.05 | 1 |
| 3 | 11.1 | 428 | 39 | 0.1 | 0.26 |
| 4 | 21.1 | 759 | 36 | 0.09 | 0.71 |
| 5 | 28.8 | 431 | 15 | 0.04 | 0.10 |
| 6 | 10 | 283 | 28 | 0.07 | 0.14 |
| 7 | 8 | 205 | 26 | 0.07 | 0.07 |
| 8 | 4.9 | 289 | 59 | 0.15 | 0.29 |
| 9 | 12.9 | 680 | 53 | 0.13 | 0.87 |
| 10 | 19 | 794 | 42 | 0.11 | 0.88 |
| 11 | 10.8 | 512 | 47 | 0.12 | 0.38 |
| 12 | 2.9 | 224 | 77 | 0.19 | 0.26 |
| 13 | 7.5 | 417 | 56 | 0.14 | 0.37 |
| 14 | 12 | 682 | 57 | 0.14 | 0.94 |
| 15 | 4.1 | 209 | 51 | 0.13 | 0.15 |
| 16 | 15.8 | 512 | 32 | 0.08 | 0.26 |
| 17 | 18.3 | 606 | 33 | 0.08 | 0.44 |
| 18 | 8.9 | 400 | 45 | 0.11 | 0.29 |
| 19 | 38.1 | 1235 | 32 | 0.08 | 1 |
| 20 | 61.2 | 1572 | 26 | 0.07 | 1 |
| 21 | 3.6 | 500 | 139 | 0.35 | 1 |
| 22 | 11.8 | 694 | 59 | 0.15 | 1 |
| 23 | 0.5 | 199 | 398 | 1 | 1 |
| 24 | 6.1 | 260 | 43 | 0.11 | 0.18 |
| Mean | 21.9 | 648 | 60 | 0.15 | 0.56 |

*) P = Function Points divided by Actual Effort.

In Table 2, we show the results of the three productivity measures on the Albrecht-Gaffney data set. Assuming that the software projects exhibit CRS, we observe that the most productive project in the Albrecht-Gaffney data set is project 23 (P=398). Applying DEA CRS (i.e. a normalised CRS productivity scale), we observe it is the only project that is fully efficient ($E_{CRS}$=1.0). Project 23 is the same project that Banker and Kemerer [8] found to be the most efficient project. They also used DEA CRS. Still assuming CRS, we further observe that the productivity for project 23 largely exceeds any of the other projects.

[1] This is the standard as per 1997. However, there is continuous research to improve the metrics. The data have been reported in a Lotus Notes repository by "knowledge champions" (project team members responsible for contributing to knowledge management within the firm. There are several hundreds of them). Also, they have persons responsible for maintaining the repository, and the repository is accessible from all over the world.

To assess the validity of this CRS result, it is necessary to observe that this project is the smallest project in terms of effort (0.5 PM) and functionality (199 FP). In terms of effort, project 23 is significantly smaller than the next smallest project (project 12) which is almost six times as large as project 23. Therefore, it is not unreasonable to deem project 23 an outlier and therefore as not representative of a typical Albrecht-Gaffney project. In the univariate CRS case, outliers like this project can easily be detected by scatter plots (like Figure 2). However, in the multivariate CRS (or VRS) case, scatter plots are inapplicable, and one has to use other techniques like sensitivity analysis to detect such outliers and other sources of error.

**Table 3: VRS reference set for Albrecht-Gaffney data set**

| Project ID | 2 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|
| 1 | 0.54 | 0 | 0.46 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0.76 | 0 | 0.24 |
| 4 | 0 | 0.12 | 0 | 0 | 0.88 | 0 |
| 5 | 0 | 0 | 0 | 0.77 | 0 | 0.23 |
| 6 | 0 | 0 | 0 | 0.28 | 0 | 0.72 |
| 7 | 0 | 0 | 0 | 0.02 | 0 | 0.98 |
| 8 | 0 | 0 | 0 | 0.30 | 0 | 0.70 |
| 9 | 0 | 0 | 0 | 0.07 | 0.93 | 0 |
| 10 | 0 | 0.18 | 0 | 0 | 0.82 | 0 |
| 11 | 0 | 0 | 0 | 0.94 | 0.06 | 0 |
| 12 | 0 | 0 | 0 | 0.08 | 0 | 0.92 |
| 13 | 0 | 0 | 0 | 0.72 | 0 | 0.28 |
| 14 | 0 | 0 | 0 | 0.06 | 0.94 | 0 |
| 15 | 0 | 0 | 0 | 0.03 | 0 | 0.97 |
| 16 | 0 | 0 | 0 | 0.94 | 0.06 | 0 |
| 17 | 0 | 0 | 0 | 0.45 | 0.55 | 0 |
| 18 | 0 | 0 | 0 | 0.67 | 0 | 0.33 |
| 19 | 0 | 1 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 1 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 1 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 1 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 1 |
| 24 | 0 | 0 | 0 | 0.20 | 0 | 0.80 |

Next, assuming VRS rather than CRS, we find six efficient projects (i.e. where $E_{VRS}$=1) as opposed to one for the CRS case (Table 2, column $E_{VRS}$). Among the six efficient projects, two are at the very end of the frontier, the smallest (project 23) and the largest (project 2). Examining the scatter plots (Figure 4), we also observe that two other frontier projects (projects 19 and 20) do not have any other projects in their neighbourhood and that the cluster of projects is between 200 and 700 FP. Only in this area are the results reasonably robust. Outside this area, the results are less reliable. That is, we should be more careful concluding that the smallest project (23) or the largest projects (19, 20 and 2) are fully efficient. We also observe that visual inspection of scatter plots still can be used as a method to identify the VRS frontier for a univariate data set.

The assumption of VRS seems justified by the results of the average efficiency numbers of $E_{CRS}$ and $E_{VRS}$ in Table 2. It is more reasonable that the average efficiency is around 60% than 15% for a group of homogenous projects conducted by the same organisation. Also, a large project like project 2 was highly inefficient when compared with the frontier line determined by project 23 in the CRS model. In the VRS model, project 2 has become efficient. The latter result seems more reasonable although not robust since there are too few observations above 20 PM.

Table 3 contains the VRS *reference* projects for each project in the Albrecht-Gaffney data set. (A reference project always is selected among one of the frontier projects. Furthermore, in the VRS case a reference project will always be an efficient project of similar size to the project that references it). The column headings show the IDs of the six reference projects, the same six projects that are VRS efficient ($E_{VRS}$=1.0) in Table 2 (projects 2, 19, 20, 21, 22 and 23). Reading a row in Table 3, we can identify the reference projects for a given project. For example, inspecting project 7, we find it has two projects in its reference set: projects 21 and 23 (Table 3). Especially, we observe that project 23 is a more important reference than project 21 (98% vs. 2%). The figures in the cells are weights indicating the relative importance of the reference projects. The practical benefit of this information is that the project manager of project 7 can identify which projects he ought to consult and probably copy to improve his performance. We also observe that it is reasonable to compare project 7 against project 23 and project 21 since these three projects are of a similar size (205 FP, 199 FP, 500 FP, respectively).

Finally, we also observe that for this univariate data set, the reference projects could have been just as easily identified by visual inspection of the scatter plot in Figure 3 in stead of using DEA (but we would not have obtained any quantitative efficiency scores, though). Also, the weights could in principle have been determined by measuring with a ruler on the scatter plot diagram.

## 6.2 ERP results

The full potential of DEA first becomes apparent when the inputs or outputs are multivariate and the projects exhibit VRS. For multivariate data sets, visual inspections can no longer be used to detect the frontier, and for VRS data sets, it is incorrect to use simple CRS models to calculate efficiency scores. Our ERP data set is such a multivariate VRS data set having 10 outputs (See Table 1).

The average VRS efficiency ($E_{MEAN}$), standard deviation (SD), minimum VRS efficiency ($E_{MIN}$) and the number of efficient projects ($N_{EFF}$) for Albrecht-Gaffney and the ERP data set are shown in Table 4. We observe that the figures are almost identical for the two data sets except that the ERP set has nine efficient projects vs. six for Albrecht-Gaffney. This is as we would expect since there are more outputs for the ERP data set than for the Albrecht-Gaffney data set (three and one, respectively).

**Table 4: Average efficiency results using DEA VRS**

| | N | $E_{MEAN}$ | SD | $E_{MIN}$ | $N_{EFF}$ |
|---|---|---|---|---|---|
| Albrecht-Gaffney | 24 | 0.56 | 0.36 | 0.07 | 6 |
| ERP | 30 | 0.56 | 0.36 | 0.06 | 9 |

From a process improvement perspective, these average efficiency figures tell us that there is a potential for improvement of such projects between 40 and 50% compared with the "best-in-class" projects.

**Table 5: VRS efficiency and reference set for ERP data set**

| ID | $E_{VRS}$ | 48 | 101 | 111 | 133 | 137 | 140 | 142 | 158 | 168 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.13 | 0.52 | 0 | 0.48 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.41 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | 0.24 | 0.29 | 0 | 0.58 | 0 | 0 | 0 | 0.13 | 0 | 0 |
| 48 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 63 | 0.18 | 0.47 | 0 | 0.53 | 0 | 0 | 0 | 0 | 0 | 0 |
| 73 | 0.28 | 0.62 | 0 | 0.26 | 0 | 0 | 0 | 0.13 | 0 | 0 |
| 101 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 109 | 0.48 | 0.21 | 0 | 0 | 0 | 0.16 | 0.57 | 0 | 0 | 0.05 |
| 110 | 0.90 | 0.31 | 0 | 0.44 | 0 | 0 | 0 | 0.25 | 0 | 0 |
| 111 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 112 | 0.22 | 0.04 | 0.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0.81 |
| 113 | 0.15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 127 | 0.40 | 0.95 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.05 |
| 133 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 136 | 0.84 | 0 | 0.29 | 0 | 0.14 | 0.57 | 0 | 0 | 0 | 0 |
| 137 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 140 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 142 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 145 | 0.33 | 0.33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.67 |
| 146 | 0.11 | 0.47 | 0 | 0.53 | 0 | 0 | 0 | 0 | 0 | 0 |
| 147 | 0.06 | 0.74 | 0 | 0.26 | 0 | 0 | 0 | 0 | 0 | 0 |
| 151 | 0.72 | 0 | 0 | 0 | 0.18 | 0.31 | 0 | 0 | 0 | 0.51 |
| 154 | 0.29 | 0.22 | 0 | 0.78 | 0 | 0 | 0 | 0 | 0 | 0 |
| 155 | 0.40 | 0.25 | 0 | 0 | 0 | 0 | 0 | 0.63 | 0 | 0.13 |
| 158 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 159 | 0.59 | 0.50 | 0 | 0 | 0 | 0 | 0 | 0.37 | 0 | 0.12 |
| 163 | 0.51 | 0 | 0 | 0 | 0 | 0 | 0 | 0.13 | 0.73 | 0.13 |
| 168 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 172 | 0.19 | 0.74 | 0 | 0.26 | 0 | 0 | 0 | 0 | 0 | 0 |
| 174 | 0.23 | 0.83 | 0 | 0.17 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5 shows the individual VRS efficiency scores as well as the reference projects for each project in the ERP data set. Nine projects are fully efficient ($E_{VRS}$=1). We observe that in multivariate data sets an inefficient project may have more than two projects in its reference set. For example, project 47 has three reference projects (48, 111 and 142).

## 6.3 ERP results - Sensitivity analysis of outliers

DEA identifies best practice rather than the average or say the best 10 %, which makes the technique very sensitive to extreme observations. It is, therefore, necessary to do a sensitivity analysis of outliers. There are several techniques (e.g. superefficiency [3] and analysis of reference units [36]) each with their strengths and limitations depending on the purpose of the DEA analysis. The purpose of our DEA analysis is twofold, first to identify best practice projects as well as the reference projects for individual projects and second, to determine the average efficiency of the ERP projects to quantify the overall potential for productivity improvement. For this double purpose, the simplest, and probably most reasonable sensitivity analysis is to remove all the frontier projects one by one and study the effect on the mean efficiency. We may also study the effect on the efficiency of a given project and the stability of the frontier and the reference projects for individual projects. We concentrate on the first part, presented in Table 6. The other part should be fairly obvious.

The ERP data set has nine units on the front. We do the sensitivity analysis by removing each of these nine projects one at a time. We observe that none of the frontier projects are extreme, in the sense that their removal hardly influences the average efficiency. We observe this by comparing $E_{MEAN}$ in Table 4 and Table 6. That is, there still is a potential improvement of around 40%.

Furthermore, it is useful to identify the most influential reference units, i.e. those reference units, or *peers*, that are referenced most. This has a double purpose. First, it may be used to assess the robustness of the frontier (An efficient project that is not referenced at all must be in an area with few observations, such as projects 19 and 20 in the Albrecht-Gaffney data set). Second, it may be used to identify the most worthy role models (by distinguishing the efficient projects that few or no projects reference from those efficient projects that many projects reference). The projects that are referenced most are more likely to be appropriate role models.

One method to quantify the *degree of influence* of an efficient project is by computing the *peer index* [36]. The larger the data set and the number of reference units, the more helpful this technique is as part of a sensitivity analysis. The peer index, ρ, is defined as follows.

$$r_j^m = \frac{\sum l_{ij}\left(x_{mi} - x_{mi}^p\right)}{\sum\left(x_{mi} - x_{mi}^p\right)}$$

where in our case with no slack

$$x_{mi}^p = x_{mi}E_{mi}$$

and

- $\rho^m{}_j$ – is the peer index for reference unit j and input m
- $\lambda_{ij}$ – is the determined weight for observation i with respect to reference unit j
- $X_{mi}$ – is the input m of observation i
- $X_{mi}{}^p$ – is the *potential* input m of observation i, had it been efficient, i.e. on the DEA frontier
- j - is the number of reference units
- m - is the number of inputs, in our case effort, only
- k - is the number of outputs, i.e. the multivariate size metric for the ERP projects

It is beyond the scope of this paper to discuss the technicalities of the general peer index formula. For a full account, see [36].

The idea behind the peer index is as follows in the univariate case. All the inefficient projects are evaluated relative to two efficient projects. Consider e.g. project C in Figure 4. The efficiency of this project is assessed relative to the "virtual" project B. The "virtual" project B is as linear combination of the two efficient projects 21 and 22. Furthermore, assume that project B is closer to 22 than to 21, say dividing the line segment between 21 and 22 into a 40/60 ratio. In this case, project C contributes to increasing the peer index of project 22 by 0.6 points and of project 21 by 0.4 points. If there are many inefficient projects between the line segment between projects 21 and 22, these two latter projects will get a high peer index. In other words, projects 21 and 22 would be the role models (or reference units in DEA terminology) for a large percentage of the projects. Oppositely, projects 2 and 20 in Figure 4 do have only one single project referencing them, and project 19 has none, thus getting a peer index of zero. This also tells us that in the region where we find projects 19, 20 and 2, there are few observations and therefore, there is a smaller degree of confidence in the frontier. Now, in the univariate case as in Figure 4, we do not really need the peer index to draw these conclusions. We could just as well examine the scatter plot figure. (However, we would not get quantitative figures without the peer index). The usefulness of the peer index becomes evident for multivariate data when figures such as Figure 4 are no longer an option.

The ERP results in Figure 5 suggest that project 48 is an influential reference unit. (It is referenced by 37% of the projects). This confirms our results in Table 6 where we observe an increase in $E_{MEAN}$ from 0.56 (in Table 4) to 0.61. However, the frontier did not change (i.e. no new projects appeared on the frontier) when removing it. Therefore, even though it is an influential reference unit, we do not consider it an outlier that should be removed. From Table 6, we also observe that $E_{MEAN}$ generally remains reasonably unchanged around 56% when removing one of the frontier projects at a time.

The results in Figure 5 also reveals that project 133 is not an influential reference unit (It is referenced by only 1% of the projects), telling us that there must be few observations in this region.

**Table 6: Results of sensitivity analysis of ERP data set**

| Project ID | $E_{MEAN}$ | New ID |
|---|---|---|
| 48 | 0.61 | None |
| 101 | 0.54 | None |
| 111 | 0.55 | None |
| 133 | 0.56 | 151,136 |
| 137 | 0.54 | None |
| 140 | 0.54 | None |
| 142 | 0.55 | None |
| 158 | 0.54 | None |
| 168 | 0.59 | 145 |

Project ID – ID of removed project, $E_{MEAN}$ – mean of $E_{VRS}$, New ID - New projects on the frontier

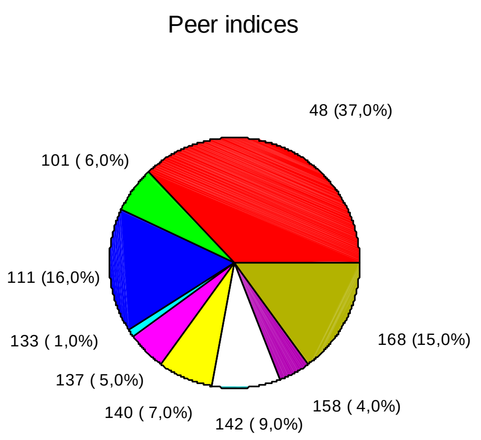

Figure 5: Pie chart of peer indices, ERP

## 6.4 ERP results - Sensitivity analysis of model specification

The original model consisted of the three input variables (Users, EDI, and Conversions). The model was determined based on expert judgement. In this model, we were in doubt whether to use EDI or Interfaces as EDI is a kind of interface and also is correlated with Interfaces. Therefore, it seems reasonable to do a sensitivity analysis of the model specification by substituting Interfaces for EDI to check if the front changes, and in case how much it changes, and also check how much the average VRS efficiency changes.

Table 7 shows the results using the modified model, i.e. output={Users, Interfaces, Conversions}. The $E_{MEAN}$ of the modified model is 0.50 which is fairly close to the average value in the original model which is 0.56 (see Table 4). We also observe that the front has changed to some degree. There are now 8 projects on the front vs. 9 in the original model. Projects 137, 142 and 158 are no longer on the front, and the latter two have got a very low efficiency. Newcomers on the front are projects 127 and 172.

**Table 7: VRS efficiency for modified model**

| Project ID | $E_{VRS}$ |
|---:|---:|
| 1 | 0.15 |
| 2 | 0.41 |
| 47 | 0.28 |
| 48 | 1 |
| 63 | 0.27 |
| 73 | 0.18 |
| 101 | 1 |
| 109 | 0.08 |
| 110 | 0.35 |
| 111 | 1 |
| 112 | 0.22 |
| 113 | 0.15 |
| 127 | 1 |
| 133 | 1 |
| 136 | 0.78 |
| 137 | 0.78 |
| 140 | 1 |
| 142 | 0.14 |
| 145 | 0.33 |
| 146 | 0.11 |
| 147 | 0.06 |
| 151 | 0.95 |
| 154 | 0.77 |
| 155 | 0.08 |
| 158 | 0.23 |
| 159 | 0.11 |
| 163 | 0.16 |
| 168 | 1 |
| 172 | 1 |
| 174 | 0.35 |

More important, when we inspect the front, we observe that six of the most influential peers remain efficient in both models. We therefore conclude that there is enough stability in our results to claim that the study has revealed that these six projects are truly efficient projects and therefore appropriate role models that are worthwhile being studied by project managers of other, less efficient, projects.

It is comforting to a practitioner that DEA is in good agreement with expert opinion in selecting the role models. Prior to this DEA analysis, we knew that project 48 stood out as a particularly successful project. In fact, this project, which implemented an ERP system in the healthcare industry, was deemed a role model long before we identified it using DEA. The implementation process of this project has been widely studied and reported within the organisation from which this data is extracted. As we observe from Figure 6, this project stands out in the DEA analysis as well since almost half of the projects (37%) have this as one of their reference units.

The expert evaluation of project 48 as a role model is based, we believe, on rather simple observations: it implemented a lot of SAP modules, and several interfaces and conversions, for many users with relatively little effort and short duration. The expert evaluation is, we believe, therefore similar to an informal application of a multivariate productivity model. DEA only formalises and makes explicit the expert judgement and provides quantitative figures rather than qualitative expert assessments.

We have not personally checked what project 48 did differently, but from our own experience as an ERP project manager (One of the authors is an experienced ERP project manager) we can propose an educated guess. We think project 48 adapted a different strategy to requirements specification: "work smarter, not harder". In stead of asking the users what they needed (by gathering requirements in the usual manner), they were proactive and simply convinced the users: "This is what you need" and rolled out a preconfigured SAP solution. This strategy has later been incorporated in the Accelerated SAP (ASAP) methodology to speed delivery of SAP implementations. (ASAP did not exist in the early 1990'ies when most of these projects were conducted adopting a traditional waterfall model). This approach works when the consultant has a lot of authority and knows the business domain as well as the SAP functionality well. The analogy with traditional software projects is as follows. The usual software methodology is a waterfall like model: specify requirements by asking the user, analyse requirements, then design, code, test and deploy the system. A different methodology would be as follows: present an existing solution to the users, then convince the users that this is what they need, train them and deploy. Actually, this is no different from what Microsoft does with their Office suite: First, develop what you think the users need. Next, persuade the users that Office is exactly what they need. Whatever you do, do not ask ten million users what they need. It would just drive you crazy reconciling all the diverse requirements.

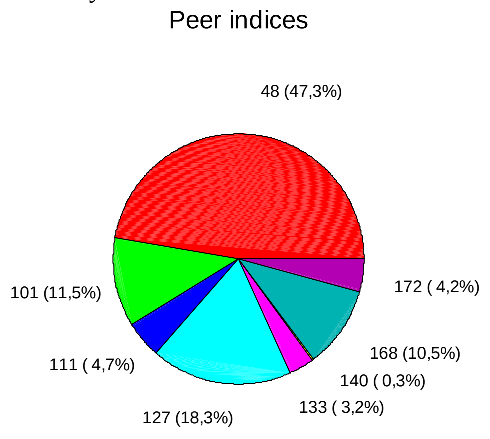

Figure 6: Peer indices of modified model, ERP

## 6.5 ERP results - Hypothesis testing

Apart from telling us how much room there is for productivity improvement, average figures of DEA results (such as in Table 4) may be used to identify what characterises the most efficient projects. This can be used to test hypotheses about the alleged superiority of a certain technology or a certain process improvement technique. For example, one can test whether a certain programming language or a database product or a process technique such as peer reviews improves efficiency or not.

We illustrate the hypothesis testing technique used in conjunction with DEA by investigating whether or not the average efficiency varies with industry. (This is probably not the most exciting hypothesis to state. Since the purpose, however, merely is to illustrate hypothesis testing in conjunction with DEA, it will do.) If efficiency varies with industry, it might be unfair and unreasonable to compare the productivity across industries for evaluation purposes. On the other hand, this information may be used to discover what ERP projects in a certain industry have in common that apparently make them more efficient. The industries in our sample are shown in Table 8.

**Table 8: Average VRS efficiency per industry in ERP data set**

| Industry | N | Mean | Median |
|---|---|---|---|
| Manufacturing | 11 | 0.56 | 0.51 |
| Energy | 3 | 0.46 | 0.19 |
| Process | 7 | 0.36 | 0.28 |
| Consumer | 7 | 0.79 | 0.9 |
| Unclassified | 2 | 0.57 | -- |

The preliminary results in Table 8 suggest that projects in the Consumer industry are the most efficient ($E_{VRS}$= 0.79) and that projects in the Process industry are the least efficient ($E_{VRS}$= 0.36). The significance tests in Table 9 confirm the preliminary result that the Consumer industry is more efficient than the other industries combined. We have used analysis of the variance (ANOVA) of the mean and Mann-Whitney of the median. Both tests are significant at the 5% level.

A problem with hypothesis testing with the DEA measures is the fact that the distributions are truncated (at 1). ANOVA assumes a normal distribution and is therefore not completely correct, and the results from ANOVA should therefore be treated with care. Mann-Whitney does not have any such requirement concerning the distribution and is therefore more suited in this case. A third alternative is a DEA adjusted F-test developed by Banker [4]. This test, however, requires a large number of observations in the sample. Still another alternative for hypothesis testing would be to use Tobit regression analysis [35] that handles truncated distributions such as the DEA efficiency scores that are truncated at 1.

The significance tests in Table 10 confirm the preliminary result that the Process industry is significantly less efficient than the other industries combined. Process industry projects should therefore be compared with projects from other industries with caution.

Of course, the point with this investigation is not to encourage everybody to leave the Process industry and join the Consumer industry if you are into SAP implementations. The point is to show that one must exercise much care to avoid unfair and meaningless productivity comparisons. We have already demonstrated that it is meaningless to compare the productivity of a small project with a large project provided software projects assume VRS. This latter result also suggests that it may be unfair to compare the productivity of SAP projects across industries if the productivity measurements are used e.g. as a basis for compensation schemes. In short, one must control for as many factors as possible to ensure valid productivity comparisons and infer correct conclusions.

**Table 9: Efficiency of Consumer industry vs. the others using ANOVA and Mann-Whitney significance tests**

| Industry | ANOVA | Mann-Whitney |
|---|---|---|
| | Mean | Median |
| Consumers | 0.79 | 0.9 |
| The Others | 0.48 | 0.33 |
| Significance level of difference | 0.04 | 0.03 |

**Table 10: Efficiency of Process industry vs. the others using ANOVA and Mann-Whitney significance tests**

| Industry | ANOVA Mean | Mann-Whitney Median |
|---|---|---|
| Process | 0.35 | 0.28 |
| The Others | 0.62 | 0.59 |
| Significance level of difference | 0.09 | 0.05 |

## 7. Discussion

There are a number of assumptions underlying performance and productivity models like DEA and univariate CRS models. Most of the assumptions are general and apply to any performance and productivity model. Only a few assumptions are particular to DEA. We find it important to differentiate between general assumptions underlying all kinds of productivity models and assumptions that are specific to DEA. If one does not differentiate between the assumptions underlying all productivity measurements and the assumptions specific to DEA, one runs the risk of unjustly criticising DEA for making unrealistic assumptions and hence reject the use of DEA on false grounds.

We argue in this section that the major objections one may raise against assumptions underlying productivity measurements of individual observations actually concern assumptions made by *all* productivity models, not limited to DEA. This includes the simple univariate CRS models that are widely applied in software engineering and against which we have hardly seen any critique raised in software engineering studies on productivity regarding their underlying assumptions.

We therefore present the most important assumptions underlying productivity measurements and discuss the extent to which these assumptions are valid in software engineering.

### 7.1 Performance and productivity

*It is hard to find appropriate* ***performance*** *indicators, and it may ultimately entail comparing apples and pears*. Measuring performance is a difficult task. Ideally, performance assessments should include *productivity* indicators as well as *quality* indicators, and it should also take other *external factors* into account such as schedule constraints. (It is well established that projects with compressed schedules require more effort to produce the same amount of software [13][16].) In practice, it is, unfortunately, often too difficult to assess projects based on combined productivity and quality measures because it requires comparing apples and pears. For example, assume that project A produces 100 function points per personmonth and has 10 defects per 1000 lines of code. Next, assume that another project B produces 200 function points per personmonth and has 20 defects per 1000 lines of code. Obviously, B has higher productivity but lower quality than A. Which project rates highest in overall performance? The answer generally is: "Don't know", unfortunately.

The pragmatic answer to this problem is to simplify the measuring task by measuring the *productivity*, only, and use the productivity measures as indicators of project performance acknowledging that productivity assessments do not tell the whole story. Productivity comparisons may be quite unfair unless one takes precautions to ensure that the projects to compare are reasonably equal and homogeneous in the other dimensions such as quality.

*It is hard to find appropriate* ***productivity*** *indicators, too.* So, performance comparisons are difficult. Does that mean that the problem is solved if we restrict ourselves to comparing productivity rather than overall performance? Unfortunately, the answer is a partial "no". Consider projects A and B once more. Assume A produces 100 FP per PM and B produces 200. Is B more productive than A? The answer is: "It depends". If one believes that the number of FP per PM is a fair indicator of productivity, B is twice as productive as A. However, let us assume that project A had to develop the technical infrastructure first whereas project B has the technical infrastructure in place and can focus on producing functionality from day one. Would we still consider B as the most productive project, and would the number of FP per PM be a reasonable and just indicator of comparison? Not necessarily.

We summarise the performance and productivity issues as follows. First, we resort to *productivity* comparisons as a substitute for *performance* comparisons because the latter often may require comparing apples and pears and is too difficult to carry out. This simplification entails the implicit assumption that productivity and performance are reasonably correlated, an assumption that does not always hold completely in software engineering. Second, we use productivity indicators that are easy to collect and count, such as function points and effort (or multivariate outputs like Users, Sites, etc. for ERP projects), assuming that these indicators are valid measures for productivity comparisons of individual software projects. This assumption does not always hold either.

We would like to repeat that these objections are general to all productivity models and not an objection against DEA, only. Specifically, it should be observed that the choice of appropriate productivity indicators such as FP or Users, Sites, etc. is *not* linked with whether or not DEA is used. DEA just *enables* you to apply multivariate productivity indicators.

## 7.2 General assumptions of productivity measurements

*Productivity models aimed at comparing* ***individual*** *units (individual projects or individual programmers) have to be non-stochastic, or deterministic*. DEA is incorrectly criticised because it is a *deterministic* model as opposed to stochastic models like regression analysis that allow for random errors. In the following, we argue that determinism is not particular to DEA but rather is a property of productivity models in general. In particular, univariate CRS models like Equation 1 also are deterministic. Consider the simple and widely applied univariate CRS productivity model in Equation 1 (e.g. y=FP, x=effort). Assume project A has x=100, y=100, and project B has x=100, y=200. The productivity of A and B is P=1 and P=2, respectively. If you state that B is twice as productive as A, one implicitly assumes that there are no random errors, i.e. that there are no model errors and no measurement errors. That is, you assume that the model is correct i.e. that FP is an appropriate indicator of software project output, and actual effort is an appropriate indicator of project input, and furthermore, that there are no counting errors of FP or effort. All the studies comparing the productivity of *individual* projects, or individual programmers, that we are aware of in software engineering have implicitly assumed a deterministic model. Otherwise, one could not have concluded that project B is more productive than project A, or benchmark an individual project (like e.g. [10][26][15] do), or in other studies, conclude that programmer X is more productive than programmer Y (like e.g. [29][17] do).

This is very different from traditional statistical methods like regression analysis that make the opposite assumption. Statistical, stochastic techniques are applicable when comparing *samples*, not individuals. In regression analysis, the stochastic error term is included to account for everything that is imperfect such as *model* errors (e.g. that FP does not account adequately for project output) and *measurement* errors (e.g. that FP counts are inaccurate). Productivity differences are likewise embedded in the stochastic error term. Thus, in a stochastic model like regression analysis, it is impossible to distinguish model and measurement errors from true productivity differences. They are all put in the same basket, the stochastic error component.

Acknowledging that productivity models like Equation 1 are deterministic, it is legitimate to ask how valid a deterministic assumption is for productivity assessments of *individual* software projects.

*Measurement errors*. We certainly know that there are measurement errors in FP counts. See e.g. Kemerer's study [23]. In addition, from personal experience we know there are measurement errors in effort reporting, too. Project members report time differently, and project managers manipulate effort reports to fit the budget. No studies similar to Kemerer's exist for ERP projects. Thus, we do not know the extent of measurement errors for the multivariate project output metric (Users, EDI, Interfaces, Conversions, etc.)

*Model errors*. We also know there are model errors in univariate CRS software productivity models like Equation 1. FP counts do not account appropriately for e.g. technical infrastructure development, compressed schedules and quality. Some projects do a lot of technical infrastructure work whereas other projects start with an existing technical infrastructure and can concentrate on producing functionality. Some software projects have severe schedule constraints and strict quality requirements (usability, MTBF, response time) whereas other projects have relaxed schedule and quality requirements.

As for ERP projects, there probably are smaller model errors than for CSD projects. First, ERP projects are more homogeneous than CSD projects in general. The user interface is given and constant (i.e. same usability), the MTBF is given and constant (since it is the same software package) and response time is given (to improve it, the usual solution is to purchase more hardware, rather than reprogram the software package). In addition, in the company Accenture (from where the ERP data originate), a standard methodology is employed for ERP projects implementing the SAP software package. Therefore, time reports are presumably more homogeneous than we would generally find in software data sets coming from diverse companies.

Regarding the three output measures (Users, EDI, Conversions) used for ERP projects in this study, we believe that they are at least as appropriate productivity indicators as FP is for the broad class of CSD projects. These ERP output indicators are certainly not perfect. For example, let us assume that the users in project A are hostile to the proposed business process reengineering (BPR) that is part of any ERP project whereas the users in project B are not. If this is the case, project A will appear less productive than project B based on counts of Users, Sites, Interfaces, Conversions, Modules, etc., since this "degree-of hostility-to-BPR" factor is not entered explicitly in the model. In this case, the productivity comparison will be unfair, and we run the risk of identifying only lucky projects (with a high "friendly-to-BPR" factor) as role models.

The wisdom to draw from this discussion is that productivity benchmarks ought not to be used uncritically in, say, compensation schemes. Still, productivity measurements might add value by *assisting* us in identifying role model projects that other project managers may want to take a closer look at. Nevertheless, we observe that it is difficult to be certain that we have found a deserving role model and not just a lucky fluke due to model or measurement error.

So, we may ask, what is the alternative when facing a complicated world and there only exist simplistic, imperfect models of this complex world? One solution is to strive to improve the models. We may define *more rigorous counting rules* so as to reduce measurement errors, and we may identify *more and better performance and productivity indicators* so as to reduce model errors. We may also strive to ensure that the projects we compare are reasonably homogeneous in the dimensions that are not explicitly entered into the model (e.g. quality, schedule constraints, inclusion of other products such as technical infrastructure).

Another, less attractive alternative is to give up, i.e. to refuse to measure productivity on the grounds that productivity measurements are imperfect and thus that productivity comparisons might be unfair. Yet another, apparently more attractive alternative is to account for stochastics in productivity models such as SDEA aims at. However, to do that one must be able to separate productivity from other components of the stochastic error term. Still another alternative is to perform more *controlled experiments* where one has better control over all the other external factors so that one may more rightly assume that a deterministic model is valid. In such a case, you rule out data from real world projects as a source of information and learning. Also, customers require productivity benchmarks from real projects, not from controlled experiments. Therefore, productivity assessments of controlled experiments can complement, but not replace, productivity assessments of real projects.

*Productivity comparisons assume that there is a benchmark against which to compare project productivity*. Consider the case of projects A and B above where P=1 and P=2, respectively. If we state that A is half as productive as B, we have implicitly used the productivity of B as a reference, or best practice frontier, benchmark. Another alternative is to compute the average productivity of the projects and use this average as the benchmark. Yet another alternative is to use a theoretically based benchmark. This is done in other disciplines, e.g. economics. This alternative does not seem viable in software engineering since theories regarding the theoretical maximum project productivity are non-existent.

*Productivity techniques generally assume that we know the correct variables*. Unlike e.g. stepwise and best subset regression analysis, productivity techniques like DEA do not assist us in *selecting* the most important variables based on some goodness of fit criterion. Rather, one must rely on a hypothesis or a theory to select variables. In practice, a pragmatic approach must be adopted in software engineering. We measure the things that are measurable and use them as indicators. For example, we measure the amount of software functionality by counting FP since it is countable, but we do not have good indicators to account for the amount of technical infrastructure development, so we simply leave it out of the model. For ERP projects it is no different. It is easy to count the number of Users, Sites, EDI, Interfaces and Conversions compared to measuring "user hostility to business process reengineering".

For the productivity comparisons of this ERP data set, we used a subset of the outputs available. There were ten measures, and we used only three in the DEA model. This assumes that the three variables selected are correlated with the others. Generally, this assumption is valid for ERP projects. A project with many Users usually also produces many Reports, Modules, etc. Of course, there are exceptions to the rule. We selected the three variables based on expert knowledge. (One of the authors is an experienced ERP project manger). He was also supported by best subset regression analysis.[1] We therefore believe that this assumption is reasonably valid.

VRS productivity models based on comparisons with observed best practice assume that observed best practice projects are representative role models and not just some extreme outliers. This assumption can be tested as we have done with sensitivity analysis. Using DEA, we can in addition justify this assumption by recourse to techniques such as the peer index that assist us in identifying robust parts of the frontier from less robust, "thin" parts of the frontier.

## 7.3 DEA specific assumptions

*Multivariate data*. DEA does in fact not make many specific assumptions other than an assumption about how to handle multivariate cases. DEA proposes a method of obtaining a *single* productivity score for multivariate cases. The alternative would be to use a series of simple y/x ratios e.g. Users/Effort, EDI/Effort and Conversions/Effort. In some cases, this is an alternative method, and in other cases one runs into difficulties when adopting this latter method. Consider the projects A and B in Table 11. For A, we have that Users/Effort=1 and EDI/Effort=0.2. For B, the figures are Users/Effort=2 and EDI/Effort=0.1. Therefore, A is most productive in Users and B in EDI. Which project is overall most productive, A or B? DEA offers a solution to this problem by creating a single efficiency or productivity score (in stead of three ratios). DEA makes one assumption in doing this. It assumes that all dimensions have equal weight in normalised space. This is an assumption made by many other multivariate measures e.g. *multivariate euclidean distance* as implemented in the estimation by analogy tool ANGEL [30].

**Table 11. Example of ERP projects**

| Proj ID | Effort | Users | EDI | Conversion |
|---|---|---|---|---|
| A | 1000 | 1000 | 200 | 100 |
| B | 1000 | 2000 | 100 | 100 |
| C | 1 | 1 | 1 | 1 |

*Returns to scale*. The DEA CRS model assumes constant returns to scale just as simple productivity models like Equation 1. Therefore, the DEA CRS is no different from Equation 1 in this respect. It differs from Equation 1 in that it handles multivariate productivity indicators, only. Unlike DEA CRS and the CRS model in Equation 1, the DEA VRS model assumes variable returns to scale. The latter assumption seems more appropriate for ERP projects, and many software data sets exhibit VRS properties. This includes the COCOMO as well as the Albrecht-Gaffney data sets. Consider again projects A, B and C in Table 11. A and B have equal effort. In this case, it seems justified to compare them in a CRS scheme. Next, consider A and C in the User dimension only. Both have the same CRS productivity ($P=y/x=1$). Using a univariate CRS productivity model, they would therefore be rated as equally productive. Is the CRS model a fair model for comparison in this case? Probably not. Project A likely is a multi-person project whereas project C is a single-person project. Therefore, A incurs a lot more overhead costs in coordinating team members and resolving conflicts of interest among the users. Thus, it seems more reasonable to conclude that A is in fact a lot more productive than C. Therefore, the CRS model is an invalid model when it comes to comparing the productivity of small and large projects. Still, the (univariate) CRS model is widely applied in software engineering studies without any consideration for scale (See e.g.[12] [15][22] [26][9]). DEA VRS seems a much more correct and reasonable model than a CRS model when the data set includes small as well as large projects.

## 7.4 Other considerations

*Multivariate DEA VRS*. In the multivariate DEA *VRS* case, we require that the number of observations is much larger than the number of variables and that there are not too many specialised units. A considerable number of observations are characterised as efficient unless the sum of the number of inputs and outputs is small relative to the number of observations. Specialised units will typically be on the frontier. For example, an ERP project that delivers Conversions, only, will probably produce more Conversions than any

[1] Best Subsets is an efficient way to select a group of "best subsets" for further analysis by selecting the smallest subset that fulfils certain statistical criteria. The subset model may actually estimate the regression coefficients and predict future responses with smaller variance than the full model using all predictors. Berenson and Levine [11] explain the approach and how to perform it in Minitab.

of the other projects that deliver both EDI, Interfaces and Conversions, and therefore this atypical project will be deemed efficient by DEA. In our study none of the projects had such characteristics.

*Distribution of efficiency measures*. The output from DEA, the efficiency measures, do not have a normal distribution that lends itself to simple statistical analysis since the distribution is truncated at 1. There are, however, more advanced techniques that may be used such as e.g. Tobit regression analysis [35].

*Weighting of the dimensions*. As with most other multidimensional measures, DEA does not solve the problem of weighting the dimensions. All the dimensions are normalised, i.e. they have equal weights. It should be observed that FP suffer from similar problems since the weights of the individual counts (e.g. external inputs, outputs, files) were fixed based on analysing one single data set. As opposed to this crude approach, a statistical technique like, say, regression analysis is more sophisticated in that it provides a technique to weight the dimensions (by providing the sample regression coefficients).

*On proving VRS.* DEA VRS handles variable returns to scale but one should be cautious in using it to *prove* the data are VRS. DEA handles CRS as well as VRS cases. However, if there are large random errors, one may wrongly conclude that the data are VRS when in fact they are CRS since DEA assumes there are no random errors. Therefore, it may be dangerous to use a VRS productivity model (like DEA VRS) to prove the existence of (dis)economies of scale such as Banker et al. do [8]. However, if one has a priori knowledge or a hypothesis stating that the data exhibit VRS (or (dis)economies of scale), DEA may be used as an indicator of the plausibility of this hypothesis.

## 8. CONCLUSIONS

The conclusions in this paper are of two kinds: i) conclusions on the results of the empirical study and ii) conclusions on the usefulness of DEA.

As for the results, we have identified six projects that appear to be important role models, thus deserving to be studied as part of a software process improvement initiative. These six projects remain stable on the front across different model specifications. The results further suggest that the average efficiency is approximately 50%. Consequently, there seems to be a substantial improvement potential compared with the “best in class” projects. Thus, it would seem beneficial to identify the common denominators of the process followed by these best practice projects and update the project methodology accordingly.

The results of the hypothesis testing suggest that there are significant differences in productivity between projects in different industries. If this is the case, one should exhibit caution when benchmarking across industries.

The results do not contradict the findings of Banker et al. [8] and Boehm [13] that software projects exhibit VRS. Indeed, ERP projects seem to exhibit similar characteristics. Thus, it seems appropriate to use a VRS model rather than a CRS model also for ERP projects.

Regarding the usefulness of DEA, we conclude that it is a pragmatic, useful method for productivity assessments of *individual* software projects because it enables us to compare apples with apples and pears with pears. Our recommendations regarding the use of DEA versus the commonly used univariate CRS productivity models are as follows.

*Use DEA VRS if the data set includes both small and large projects*. Unlike univariate CRS models (e.g. FP per PM), the DEA *VRS* model enables us to perform meaningful productivity comparisons between small and large projects by comparing small projects with small projects and large projects with large projects. The DEA VRS model therefore better assists us in identifying best practice projects in data sets where the projects span from small to large projects. Using a CRS model, we would not be able to discover all the projects deserving to be considered as potential role models. Worse, we might wrongly deem the smallest project as the most productive. (We believe it would not be very instructive to scrutinise a small one-person, one-month project in order to understand how to improve a 100-person, 5-year megaproject.)

*Use DEA (CRS or VRS) if the data set contains multivariate input or output indicators.* Unlike simple univariate CRS models (e.g. FP per PM), the DEA model provides a single productivity (or efficiency) measure in the multivariate case. It is therefore more appropriate than univariate models in the context of ERP projects since the output indicator from ERP projects is multivariate.

Also, DEA is appealing to a software practitioner because it uses the *best practice frontier* as a benchmark rather than some theoretical baseline. In software engineering, it seems sensible to compare the productivity with best practice rather than with some theoretical optimal (and probably non-attainable) productivity.

Finally, this study suggests that DEA used together with methods for hypothesis testing may be a useful technique for assessing the effect of alleged process improvements.

*There are no serious objections to be made against DEA*. The objections regarding its deterministic nature are general to productivity measurements and comparisons of *individual* observations and not particular to DEA. Therefore, whenever one performs productivity comparisons of individuals, one needs to exercise some caution and be somewhat sceptical to the results. Still, we recommend that productivity measurements and comparisons are made in software engineering, and we recommend that DEA VRS is adopted as the default productivity model, and that it replaces the commonly used univariate CRS models. To our knowledge, the univariate CRS model is currently the only model used in software engineering studies of productivity.

To summarise, productivity measurements are required to identify role models and best practice projects and to provide rough benchmarks and average efficiency scores. We recommend that productivity measurements are performed provided that certain guidelines are obeyed. First, the model must be fair, i.e. that the input and output indicators are carefully selected, and exogeneous factors are controlled for (to the extent possible). Second, we must have confidence that the measurement errors are small. Third, appropriate sensitivity analyses must be done. Fourth, productivity measurements should be used mainly to *assist* in identifying the best projects but not as the sole basis for compensation schemes or bombastic conclusions as to which project is deemed best.

## ACKNOWLEDGEMENTS

This work was supported in part by Accentures Research Fund in Norway. The authors are indebted to Barbara Kitchenham for her persistent inquiries into the fundamental assumptions underlying deterministic and non-deterministic models. We would also like to thank the anonymous referees for raising similar issues as well as for a number of useful suggestions to improve the paper.

## REFERENCES

[1] S.N. Afriat, “Efficiency Estimation of Production Functions”, *International Economic Review,* vol.13, pp. 568-598, 1972.

[2] A.J. Albrecht and J.R. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”, *IEEE Trans. Software Eng*., vol. 9, no.6, pp. 639-648, 1983.

[3] P. Andersen and N.C. Petersen, “A Procedure for Ranking Efficient Units in Data Envelopment Analysis”, *Man. Sci*., vol. 39, no. 10, pp. 1261-1264,1993.

[4] R.D. Banker, “Maximum Likelihood, Consistency and Data Envelopment Analysis: A Statistical Foundation”, *Man. Sci*. vol. 39, no. 10, pp.1265-1273, 1993.

[5] R.D. Banker, H. Chang, and C.F. Kemerer, “Evidence on Economies of Scale in Software Development”, *Inf. and Software Tech*., vol. 36, no. 5, pp. 275-82, 1994.

[6] R.D. Banker, A. Charnes, and W.W. Cooper, “ Some Models for Estimating Technical and Scale Inefficiencies”, *Man. Sci.,* vol. 39, pp. 1261-1264, 1984.

[7] R.D. Banker, S.M. Datar, and C.F. Kemerer, “A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects”, *Man. Sci*., vol. 37, no.1, pp. 1-18, 1991.

[8] R.D. Banker and C.F. Kemerer, “Scale Economies in New Software Development”, *IEEE Trans. Software Eng*., vol. 15, no. 10, pp. 1199-1205, 1989.

[9] V.R. Basili, L.C. Briand, and W.L. Melo, "How Reuse Influences Productivity in Object-Oriented Systems", *Comm. ACM*, Vol. 39, No. 10, pp. 104-116, 1996.

[10] C.A. Behrens, "Measuring the Productivity of Computer Systems Development Activities with Function Points", *IEEE Trans. Software Eng*., vol. SE-9, no. 6, pp. 648-652, 1983.

[11] M.L. Berenson and D.M. Levine, *Basic Business Statistics - Consepts and Applications*, Prentice-Hall International, pp.889-890, 1999.

[12] J.D. Blackburn, G.D. Scudder and L.N. Van Wassenhove, "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers", *IEEE Trans. Software Eng*., vol. 22, no.12, pp. 875-885, 1996.

[13] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall: Englewood Cliffs N.J., 1981.

[14] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "The Cocomo 2.0 Software Cost Estimation Model - A Status Report", *American Programmer*, pp. 2-17, 1996.

[15] L.C. Briand, K. El-Emam, F. Bomarius, "COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment", *Proc. 20th ICSE*, pp. 390-399, 1998.

[16] F.P. Brooks Jr., *The Mythical Man-Month - Essays on Software Engineering*, Anniversary Edition, Addison-Wesley: Reading Massachusetts, 1995.

[17] G.E. Bryan, "Not All Programmers are Created Equal", 1994 *IEEE Aerospace Applications Conference Proceedings*, IEEE New York, NY, pp. 55-62, 1994.

[18] A.Charnes, W.W. Cooper, and E. Rhodes, “Measuring the Efficiency of Decision Making Units”, *Eur. J. Oper. Res*., vol. 2, pp. 429-444, 1978.

[19] J. Doyle and R. Green, “Strategic Choice and Data Envelopment Analysis: Comparing Computers across Many Attributes”, *J. Inf. Tech*., vol. 9, no. 1, pp. 61-9, 1994.

[20] D.M. Fisher and D.B. Sun, “LAN-based E-mail: Software Evaluation”, *J. Computer Inf. Sys*., vol. 36, no. 1, pp. 21-5, 1995-96.

[21] F.R. Førsund and L. Hjalmarson, “Generalised Farrell Measures of Efficiency: An Application to Milk Processing in Swedish Dairy Plants”, *The Economic Journal*, vol. 89, pp. 294-315, 1979.

[22] R. Jeffery, M. Ruhe, and I. Wieczorek, "Using Public Domain Metrics to Estimate Software Development Effort", *Proc. METRICS'01*, IEEE Computer Society Press, Los Alamitos CA, pp. 16-27, 2001.

[23] C.F. Kemerer, "Reliability of Function Points Measurement - A Field Experiment", *Comm. ACM*, 36(2), Feb 1993, pp. 85-97, 1993.

[24] M.A. Mahmood, “Evaluating Organisational Efficiency Resulting from Information Technology Investment: an Application of Data Envelopment Analysis”, *Inf. Sys. J.*, vol. 4, no. 2, pp. 93-115, 1994.

[25] Minitab Statistical Software, release 13, htttp://www.minitab.com

[26] S. Moser, and O. Nierstrasz, "The Effect of Object-Oriented Frameworks on Developer Productivity", *IEEE Computer*, pp. 45-51, Sep. 1996.

[27] I. Myrtveit. and E. Stensrud, “Benchmarking COTS Projects Using Data Envelopment Analysis”, *Proc.*

*METRICS'99*, IEEE Computer Society Press, Los Alamitos CA, pp. 269-278, 1999.

[28] C. Parkan, K. Lam, and G. Hang, "Operational Competitiveness Analysis on Software Development", *J. Oper. Res. Society*, vol. 48, no. 9, pp. 892-905, 1997.

[29] H. Sackman, W.J. Erikson, and E.E. Grant, "Exploratory Experimental Studies Comparing Online and Offline Programming Performance", *Comm. ACM*, 11, 1, pp. 3-11, Jan 1968.

[30] M.J. Shepperd, and C. Schofield, "Estimating Software Project Effort Using Analogies", *IEEE Trans. Software Eng*. Vol. 23, no.12, pp. 736-743,1997.

[31] E. Stensrud, and I. Myrtveit, "The Added Value of Estimation by Analogy – An Industrial Experiment", *Proc. FESMA'98*, Technologisch Instituut vzw: Antwerp Belgium, pp. 549-556, 1998.

[32] E. Stensrud, and I. Myrtveit, "Human Performance Estimating with Analogy and Regression Models: An Empirical Validation", *Proc. METRICS'98*, IEEE Computer Society Press, Los Alamitos CA, pp. 205-213, 1998.

[33] C. Stolp, " Strengths and Weaknesses of Data Envelopment Analysis: An Urban and Regional Perspective", *Comput., Environ. And Urban Systems*, vol. 14, pp. 103-116, 1990.

[34] S. Thore, F. Phillips, T.W. Ruefli, and P. Yue, "DEA and the Management of the Product Cycle: the U.S. Computer Industry", *Computers & Oper. Res*., vol. 23, no. 4, pp. 341-56, 1996.

[35] J. Tobin, "Estimation of Relationships for Limited Dependent Variables", *Econometrica* 26, pp.24-36, 1958.

[36] A.M. Torgersen, F.R. Førsund, and S.A.C. Kittelsen, "Slack Adjusted Efficiency Measures and Ranking of Efficient Units", *J. of Productivity Analysis*, No. 7, p. 379-398, 1996.

[37] G. Weinberg, *The psychology of computer programming*, Van Nostrand Reinhold Company: New York, 1971.