

Using Linear Regression Models to Analyse the Effect of Software Process Improvement

Joost Schalken¹, Sjaak Brinkkemper², and Hans van Vliet¹

¹ Vrije Universiteit, Amsterdam, Department of Computer Science,
{jpp.schalken, jc.van.vliet}@few.vu.nl

² Utrecht University, Institute of Information and Computing Sciences,
{s.brinkkemper}@cs.uu.nl

Abstract. In this paper we publish the results of a thorough empirical evaluation of a CMM-based software process improvement program that took place at the IT department of a large Dutch financial institution. Data of 410 projects collected over a period of four years are analysed and a productivity improvement of about 20% is found. In addition to these results we explain how the use of linear regression models and hierarchical linear models greatly enhances the sensitivity of analysis of empirical data on software improvement programs.

1 Introduction

To improve the management and work processes in software development, the software process improvement field has proposed improvement models (such as the Capability Maturity Model [1], ISO-SPICE [2], and the Capability Maturity Model Integrated [3]) that are based on best practices and guidelines for software developing organisations. As each of these models focuses on a wide range of interacting practices, the benefits of an improvement model are not always intuitive. Therefore empirical data about the benefits of SPI is needed.

There is a lack of empirical studies on the effects of SPI in industry. Even for the most widely used improvement model, the Capability Maturity Model [1], little empirical data is available. In a recent meta analysis on the effects of the CMM, Galin and Avrahami [4] were only able to identify three studies that give details on productivity gains when an organisation progresses to CMM level 2 and only twelve studies that provide details on productivity gains when an organisation progresses to CMM level 3.

In this study we investigate the success of a software process improvement program at a large, Dutch financial institution. In this program the Capability Maturity Model has been used as the reference model for software process improvement and the Dynamic Systems Development Method (DSDM) [5] as the new project management methodology. We analysed the productivity of 410 projects during a period of four years and found a productivity increase of 20%.

If is often claimed that extensive measurement programs are ineffective in an immature organisation. We however believe that in immature organisations the

effects of SPI are invisible because the data is too noisy to be analysed with simple statistical techniques. In this paper we explain how regression models [6] and hierarchical linear models [7] help interpreting the productivity data. Regression models offer an improved sensitivity to changes in productivity and hierarchical linear models allow researchers to study SPI programs that take place in heterogeneous organisations. *Heterogeneous organisations* consist of departments that use different technologies and work on different products.

Using these statistical techniques results in an explained variance of over 60% for our data, as opposed to a mere 2% when a straight-forward productivity index is used.

The combined analysis of SPI programs that consist of similar SPI projects that are executed in these different departments requires special statistic techniques.

1.1 Research Questions

This paper provides an answer to the following two questions:

1. What is the impact of a project's CMM level on the productivity of that project?
2. To which extent do the outcomes of a CMM study depend on the choice of proper statistical techniques?

2 Related Work

Related work for this study consists of two bodies of literature: studies of the empirical results of software process improvement and literature of statistical techniques used in empirical software engineering.

In [8, 9, 10, 11] studies of software process improvement are described. The studies report the (solely positive) changes in productivity of software developing organisations, expressed in ratios of lines of code delivered per unit of effort (usually man months).

In their paper "*Do SQA Programs Work*" [4] Galin and Avrahami provide an overview of the above mentioned case studies and others that contain empirical evidence on the effects of software process improvement. They identified 22 studies relating to the effects of Capability Maturity Model based software process improvement. Of these 22 studies, only 19 contained sufficiently detailed quantitative data to allow a meta analysis of the SPI effects. The meta analysis examines the effects of CMM on error density, productivity, rework schedule time, conformance to schedule and the effectiveness of error detection.

The authors were able to locate three studies that examine the change in productivity when a software development organisation increases its maturity from CMM level 1 to CMM level 2 and ten studies that examine the change in productivity when an organisation increases its maturity from CMM level 2 to CMM level 3. On average an organisation increases its productivity by 42.3%

when it matures to CMM level 2. And an organisation that increases its maturity to CMM level 3 improves its productivity by an additional 44.4%.

Related work on the analysis of productivity data is hard to find in software process improvement papers, however research in cost-estimation provides instructions on how to apply regression models to relate effort to size [12, chap. 5] [13, chap. 12] and therefore how to analyse productivity. Unfortunately the statistical techniques used for cost-estimation models is not used to elucidate changes in productivity caused by software process improvement. Related work on hierarchical linear models applied to software process improvement has not been found³.

A subset of the data that we examined in this study, has also been analysed using different statistical techniques in [14], where a method is demonstrated to validate size measurements made by the organisation and time series are used to analyse the changes of productivity over time (irrespective of CMM level).

3 Research Methodology

3.1 Research Design

To answer our research questions, we use the experimental design of a cohort study [15, p.126]. We collected data on 410 software development projects and compared the productivity of the projects that have been executed in an environment that successfully passed a CMM assessment with projects that were executed in an environment that did not already fulfil the CMM requirements. For this comparison we used three different statistical techniques. We believe that through the repetition of the studies in multiple departments, we were able to overcome most validity threats that are commonly associated with cohort studies.

3.2 Research Context

This study has been performed within an internal Information Technology department of a large financial institution. In this department over 1500 people were employed during the course of study. The organisation primarily builds and maintains large, custom-built, mainframe transaction processing systems, most of which are built in COBOL and TELON (an application-generator for COBOL). Besides these mainframe systems, a large variety of other systems are implemented, constructed and maintained by the organisation. These systems are implemented in a large variety of different programming languages (such as

³ We did not find any relevant results in ACM's Portal, Springer-Verlag's Springer-Link, Elsevier's ScienceDirect or in IEEE Computer Society's Digital Library that contained both a relevant statistical term ("hierarchical linear model", "multi-level linear model", "mixed-effects model", "random-effects model", "random-coefficient regression model", or "covariance components model") and a relevant application domain term ("spi", "cmm", or "software process improvement").

Java and COOL:Gen), run under various operating systems (such as Microsoft Windows and UNIX) and are distributed over different platforms (batch, block-based, GUI-based and browser-based).

The organisation has undertaken a major software process improvement program (SPI) to improve the internal IT processes and cooperation with the business in the period of 1999 until 2004. The SPI program included the introduction of a software metrics program, the introduction of the Dynamic Systems Development Method (DSDM) [5] as the iterative development and project management method, the introduction of a tailor-made quality system consisting of two levels that comply with the requirements of CMM [1] level 2 and 3 respectively, and a culture change program to support the above mentioned changes.

To successfully implement the changes required for the software process improvement, a program with a dedicated senior vice-president was initiated. Through the SPI program the individual departments of the line organisation received assistance and coaching by dedicated internal and external consultants; IT staff received the required training and certification in CMM, DSDM and company-specific procedures; and improvement goals were set to maintain the commitment of upper management.

In the process the organisation designed an on-line knowledge base containing reference information about the DSDM method and the supporting management and quality processes, templates for documentation and review check-lists and background information about development tools. The on-line knowledge base was a replacement of an existing on-line knowledge base, that was based on the prior (linear) development method.

3.3 Analysis Methods

We examined three distinct ways of comparing the productivity of an organisation before and after a software process improvement program. The methods of comparing productivity increase in sophistication and complexity.

Classical approach In most studies, productivity is defined as size divided by effort. Conte, Dunsmore and Shen [12, chap.5] define productivity as “*the number of lines of source code produced per programmer-month (person-month) of effort*”. This leads to the following formula to estimate the productivity of an organisation (1):

$$\hat{L} = \frac{1}{n} \sum_{i=1}^n l_i = \frac{1}{n} \sum_{i=1}^n \frac{s_i}{e_i} \quad \text{where} \quad \begin{array}{l} \hat{L} \text{ is estimated organisational productivity,} \\ l_i \text{ is productivity of project } i, \text{ and} \\ s_i \text{ is size of the software } i, \text{ and} \\ e_i \text{ is effort of project } i, \text{ and} \\ n \text{ is the number of projects executed} \\ \text{in the organisation} \end{array} \quad (1)$$

Size measurements of a project can also be expressed in function points [16] or other size metrics and effort can also be expressed in other measures without

loss of generality. Although function points have certain desirable properties, such as technological independence [17], Galin and Avrahami [4] observe that “most of the reporting organisations applied the classic lines of code (LOC) measure for productivity”. In our study a productivity index consisting of function points (S_{fp}) per hour of effort of IT personnel (E_{hr}) is used. The effort of IT personnel includes not only programmer effort, but also the effort of requirements engineers, technical designers, architects and project management.

Productivity indices (L) can be used to determine whether the productivity has changed. To estimate the productivity index (\hat{L}) of an organisation the productivity indices of projects carried out in that organisation are averaged (1). To determine the effect of software process improvement, the productivity index of the organisation before the software process improvement initiative ($L^{\neg spi}$) is compared with that of the organisation after the software process improvement initiative (L^{spi}). A t-test [18] can be used to test whether the productivity of the organisation has changed significantly (2):

$$\begin{aligned} H_0 : L^{\neg spi} &= L^{spi} \\ H_1 : L^{\neg spi} &\neq L^{spi} \end{aligned} \tag{2}$$

Cost-model comparison approach In studies of productivity, changes in productivity are often measured in lines of code per man month [4] or in another ratio of size divided by effort. We believe however that better comparisons of productivity changes can be made based on the same data.

Instead of describing the productivity with a single productivity index L , the productivity also can be described by a regression equation that models effort as a function of size. The parameters of the regression model $\beta_{1\dots n}$ now function as a measure for productivity. The simplest regression model is a linear regression model (3):

$$e_{hr i} = \beta_0 + \beta_1 \cdot s_{fp i} + r_i \quad \text{where} \quad \begin{array}{l} e_{hr i} \text{ is effort of a project in hours,} \\ s_{fp i} \text{ is size of effort in function points, and} \\ r_i \text{ is unexplained, residual variance} \\ (R \sim N(0, \sigma^2)) \end{array} \tag{3}$$

In order to build a valid regression model, the relation must satisfy the assumptions of linear regression: (a) linearity of the relation, (b) independence of residuals, (c) residuals have constant variance and (d) residuals follow the normal distribution [6].

Unfortunately, in most organisations productivity data does not satisfy the assumptions of linear regression. In most organisations larger projects are less predictable and more prone to overrun their budget and schedules. This is an indication that the residuals (difference between observations and the regression model) are related to the size of a project, which is a violation of assumption (c) and can lead to invalid conclusions.

If the relation between effort and size violates one of the assumptions of the linear regression model, it is possible to scale the dependent (E_{hr}) and or independent (S_{fp}) variables before they are used in the regression equation. Although in principle every transformation is allowed, involution, extraction of roots and taking a logarithm are often used transformations. Based on diagnostic information, one can choose an appropriate transformation. This leads to the more general regression model (4):

$$e_{hr'_i} = \beta_0 + \beta_1 \cdot s_{fp'_i} + r_i \quad \text{where} \quad \begin{array}{l} e_{hr'_i} \text{ is scaled effort of a project in hours,} \\ s_{fp'_i} \text{ is scaled size of effort in function points,} \\ r_i \text{ is unexplained, residual variance} \end{array} \quad (4)$$

$(R \sim N(0, \sigma^2))$

Regression equations that model effort as a function of size are usually employed to build cost prediction models (such as COCOMO-II [19]). Their use however is not limited to cost estimation, as regression models are also useful to compare the productivity of organisations or to determine the productivity effects of a software process improvement initiative. Productivity of organisations can then be compared by comparing the parameters $\beta_{1...n}$ of the regression models.

To determine the effects of software process improvement, the process maturity (C) of the organisation can be factored into the equation (5) to determine the influence of software process improvement. If software process improvement has no effect, the regression parameters β_2 and β_3 should be equal to zero (6), which can be tested with ANOVA [18]. By examining the parameters β_2 and β_3 we obtain an estimate of the effect of software process improvement.

$$e_{hr'_i} = \beta_0 + \beta_1 \cdot s_{fp'_i} + \beta_2 \cdot c_i + \beta_3 \cdot s_{fp'_i} \cdot c_i + r_i \quad \text{where} \quad \begin{array}{l} e_{hr'_i} \text{ is scaled effort of a project in hours,} \\ s_{fp'_i} \text{ is scaled size of effort in function points,} \\ c_i \text{ is the maturity level of the organisation} \\ \text{in which the project is executed,} \\ r_i \text{ is unexplained, residual variance} \end{array} \quad (5)$$

$(R \sim N(0, \sigma^2))$

$$\begin{aligned} H_0 : \beta_2 = 0 \quad \text{and} \quad \beta_3 = 0 \\ H_1 : \beta_2 \neq 0 \quad \text{or} \quad \beta_3 \neq 0 \end{aligned} \quad (6)$$

The advantage of the regression model approach is that regression models take the effect of project size on the productivity of a project into account. After all, projects can have startup costs and projects can experience (dis-)economy of scale. Ignoring the effect of project size on project productivity increases the residual, unexplained variability in the data. This increased residual variance means that larger sample sizes are needed to obtain significant results. In certain

situations it is even possible that, by ignoring the effect of size on productivity, invalid conclusions are drawn (e.g. if the size of projects before and after the software process improvement changes significantly).

Hierarchical model approach Large organisations structure their work according to the principle of division of labour. These large organisations consist of different departments that perform different projects. These departments either specialise on the group of products they work on or specialise on the technology or skills that the department uses. We call these organisations with specialised departments heterogeneous organisations. In this section we explain why data from heterogeneous organisations cannot be adequately analysed using classical linear regression models and how hierarchical models [20, 7] can be used to analyse this data.

Linear regression models are based on the assumption that residuals in a model are independent, and therefore the observations need to be drawn from a single homogeneous pool of subjects. The characteristics of such a homogeneous pool from which the observations are drawn should have the same statistical distribution. Because the characteristics of departments in a heterogeneous organisation differ, the assumption of independent residuals does not hold.

We illustrate the problems sketched above with a hypothetical example. If we want to understand the effects of a type of fertiliser on the growth-rate of fruit, an experiment could be set up with eight plots of apples and eight plots of pears, four plots of bananas and two plots of pineapples. Half the plots are assigned with the new fertiliser (the treatment group) and half the plots are assigned with no fertiliser (the control group). At the moment the plants bear fruit, the yield of each of the plots is weighed. To determine the effect of the fertiliser (a) the yields of all plots with fertiliser could be compared with the yields of all plots without the fertiliser or (b) for each type of fruit the average yields of plots with fertiliser could be compared with the plots grown without fertiliser. In the first approach, we literally compare apples with pears, which severely increases error variance and therefore reduces the chance we detect the effect of the fertiliser. If we on the other hand take the second approach to gauge the effect of the fertiliser, we have to make four comparisons, which increases the chance of making an error by fourfold.

When determining the effect of software process improvement on productivity, the productivity of projects that took place before and after the software process improvement initiative are compared. Unfortunately the number of projects that take place within a single department of a company is usually too small to find significant results of software process improvement. If the productivity of projects that took place in different departments is compared, effectively ‘apples’ are compared with ‘pears’. Hierarchical linear models can be used to make a single comparison of the overall effects of changes in software process (or fertiliser) and at the same time take into account that we are comparing ‘apples’ with ‘pears’. Multi-level linear models, mixed-effects models, random-effects models, random-coefficient regression modes, and covariance components

models are other names for hierarchical level models. To measure the overall effects of software process improvement, hierarchical linear models should be used to take the differences of the departments into account.

Hierarchical linear models are an extension of ordinary linear regression models. In a hierarchical linear model, the regression model is split up in two components: a level 1 model and a level 2 model. Hierarchical linear models extend linear regression models by fitting a new set of regression parameters for each group of data, the department in which the project took place. For each group a different set of regression parameters $\beta_{0j} \dots \beta_{nj}$ is found. The level 2 model brings structure in the regression parameters; an overall γ_{n0} value for a group of parameters β_{nj} is determined, from which each group j is allowed to deviate by u_{nj} .

The level 1 model (7) for effort looks similar to the regression model from the previous section. Note however that the observations i are grouped according to the department j in which they were made. Also note that the parameters in the level 1 regression model also have a subscript for their group j .

$$\begin{aligned}
 e_{hr'ij} = & \beta_{0j} + \beta_{1j} \cdot s_{fp'ij} + & e_{hr'ij} & \text{is scaled effort of a project in hours,} \\
 & \beta_{2j} \cdot c_{ij} + & s_{fp'ij} & \text{is scaled size of effort in function points,} \\
 & \beta_{3j} \cdot s_{fp'ij} \cdot c_{ij} + & c_{ij} & \text{is the maturity level of the organisation} \\
 & r_{ij} & & \text{in which the project is executed,} \\
 & & & r_{ij} & \text{is unexplained, residual variance} \\
 & & & & (R \sim N(0, \sigma^2))
 \end{aligned}
 \tag{7}$$

We also have a level 2 model (8) for productivity, which breaks up each β_{nj} in the level 1 model into an organisation wide parameter γ_{nj} and a deviation u_{nj} from that organisation average for each department j .

$$\begin{aligned}
 \beta_{0j} &= \gamma_{00} + u_{0j} & u_{0j} &\sim N(0, \tau_{00}) \\
 \beta_{1j} &= \gamma_{10} + u_{1j} & u_{1j} &\sim N(0, \tau_{11}) \\
 \beta_{2j} &= \gamma_{20} + u_{2j} & u_{2j} &\sim N(0, \tau_{22}) \\
 \beta_{3j} &= \gamma_{30} + u_{3j} & u_{3j} &\sim N(0, \tau_{33})
 \end{aligned}
 \tag{8}$$

Bryk and Raudenbush [7, chap. 3] provide a conceptual explanation of how these models can be fitted and a more thorough mathematical description of how values for these parameters can be found is provided by Pinheiro and Bates [21, chap. 2].

To determine whether software process improvement has an effect, the parameters γ_{20} and γ_{30} should be tested for equality with zero (9). The parameters γ_{20} and γ_{30} are related with the maturity of the organisation (c_i) and if the parameters are equal to zero that would indicate that SPI has no effect.

$$\begin{aligned}
 H_0 : \gamma_{20} = 0 \quad \text{and} \quad \gamma_{30} = 0 \\
 H_1 : \gamma_{20} \neq 0 \quad \text{or} \quad \gamma_{30} \neq 0
 \end{aligned}
 \tag{9}$$

The advantage of using hierarchical linear models in determining the effects of SPI in heterogeneous organisation is that by taking the department into account, the residual variance of the data is reduced, which reduces the amount of data required to make an analysis. Furthermore the usage of hierarchical linear models can guard against making erroneous conclusions. Such erroneous conclusions could be made if workload of departments that perform easy assignments increases at the expense of the workload of departments that perform difficult assignments. In such cases the productivity of the organisation would seem to have increased, whereas in reality the work has changed and no real performance increase has occurred.

4 Results

4.1 Data transformations

To perform the evaluation of the software process improvement program, two sources of data were used: the project database and a log of the assessment results. The project database contains generic information on all projects executed in the organisation and the assessment log contains information on which domains have been assessed and what the outcomes of the CMM assessment were. From the database, data about 410 projects has been extracted.

From the project database the size in function points, the effort, the end date and the department in which the project was executed has been extracted for each project. To obtain the maturity of the organisation in which the project has been executed, the end date of a project was compared with the assessment date of its domain. If the project has been executed before the assessment, the project has been considered to have been executed in an organisation with CMM maturity level 1. The decision rule to determine the maturity of the department in which the project has been executed leads to a conservative estimate of the effects of SPI.

Please note that before we analysed the data, we have multiplied the effort data with a random constant ($0.75 < \alpha < 1.5$) for the sake of confidentiality of the actual productivity figures. This linear scaling of the data does not in any way affect the improvement ratios that are provided in this paper.

4.2 Classical approach

In this section we use the classical approach to determine the effects of software process improvement. Table 1 shows the average productivity of projects that are executed in a CMM Level 1, 2 and 3 organisation.

From Table 1 we can conclude that projects executed in a CMM level 2 or level 3 organisation are on average 20.19% more productive than projects that are executed in a CMM level 1 organisation. When we use a t-test to test hypothesis (2), which states that $L_{hr/fp}^{spi} \neq L_{hr/fp}^{-spi}$, we find that have to reject H_0 with $p = 0.002$ ($t = 3.13$, $df = 267$). We can therefore conclude that there is a significant productivity increase after the implementation of SPI.

Table 1. Productivity Indices per Maturity Level.

<i>Maturity</i>	<i>CMM Level</i>	Productivity $L_{hr/fp}$
low	1	14.46
medium	2	12.08
	3	8.50
	2 & 3	11.54

Although we do find statistically significant results with the classical approach, the results are not satisfactory if we look at the amount of explained variance. When we fit productivity as a function of process maturity (level 1 vs. level 2 and 3), we obtain $R^2 = 0.02$. This means that only 2% of the differences in productivity can be explained by process maturity.

4.3 Cost-model comparison approach

We use linear regression models to determine the effect of software process improvement on productivity. As explained in Sect. 3.3, we first need to find a suitable linear regression model between effort and size.

In Table 2 we tabulated diagnostic information on six different combinations of transformations on both the dependent (effort) and the independent variable (size). The Shapiro-Wilk test [22] is used to test for normality and the Breusch-Pagan test [6, p. 115] is to test the constance of variance of the residuals. The residuals of the linear model and the log-transformed model ($\log(E) = \beta_0 +$

Table 2. Diagnostic Information on Effort Regression Models.

<i>Formula</i>	<i>Residuals</i>				
	<i>Goodness of fit</i>	<i>Constance of Variance</i>		<i>Normality</i>	
		R^2	X_{bp}^2 ^a	p	W^b
$E = \beta_0 + \beta_1 S + R$	0.493	86.07	< 0.001	0.798	< 0.001
$E = \beta_0 + \beta_1 S^2 + R$	0.315	87.29	< 0.001	0.748	< 0.001
$E = \beta_0 + \beta_1 \log(S) + R$	0.407	33.14	< 0.001	0.830	< 0.001
$\log(E) = \beta_0 + \beta_1 S + R$	0.436	27.26	< 0.001	0.981	< 0.001
$\log(E) = \beta_0 + \beta_1 S^2 + R$	0.202	54.24	< 0.001	0.988	0.002
$\log(E) = \beta_0 + \beta_1 \log(S) + R$	0.576	4.43	0.035	0.995	0.283

^a Breusch-Pagan X^2 to test for dependence between predictors and residuals.

^b Shapiro-Wilk W to test deviation of residuals from the normal distribution.

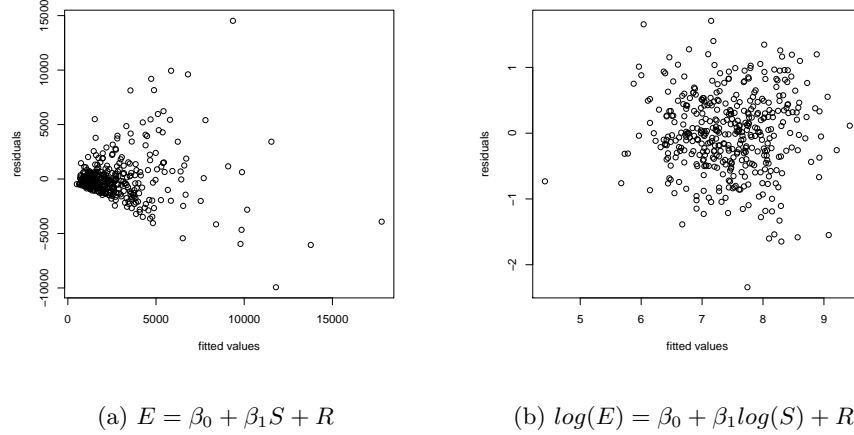


Fig. 1. Residuals of Linear Models plotted against the Fitted Values.

$\beta_1 \log(S) + R$ are shown in Fig. 1. From Fig. 1 we can see that the residuals from the linear model are correlated with the fitted values (show heteroscedascity) and that the residuals from the logistic model are uncorrelated with the fitted values (homoscedastic). Homoscedasticity, or constance of variance, is assumed in linear regression models and therefore the logistic model is superior to the simple linear model. For goodness of fit we used the unadjusted multiple coefficient of determination R^2 , which tells us how much of the variation in effort can be explained by size.

If we examine the diagnostic information in Table 2, we see that the log-transformed model ($\log(E) = \beta_0 + \beta_1 \log(S) + R$) has the best characteristics; 58% of the variance is explained and its residuals are normally distributed and the variance of the residuals only has negligible relation with size.

We can transform the model for log-transformed effort to a model for effort by raising both sides of the equation to the power e , which leads to equation (10). It turns out that we effectively arrive at an exponential effort estimation model [12, p. 281] of the form $effort_i = \beta_0 * size_i^{\beta_1} * e^{r_i}$. To determine if the maturity of an organisation has an effect on the productivity, we effectively are comparing cost-estimation models.

$$e_{hr_i} = \exp(\beta_0) \cdot s_{fp_i}^{\beta_1} \cdot \exp(r_i) \quad \text{where} \quad \begin{array}{l} e_{hr_i} \text{ is effort of a project in hours,} \\ s_{fp_i} \text{ is size of effort in function points} \\ r_i \text{ is unexplained, residual variance} \end{array} \quad (10)$$

$(R \sim N(0, \sigma^2))$

Table 3. ANOVA on Linear Regression Model ($\log(E_{hr}) = \beta_0 + \beta_1 \log(S_{fp}) + \beta_2 C + \beta_3 \log(S_{fp}) C + R$).

	<i>df</i>	<i>SSE</i>	<i>MSE</i>	<i>F</i>	<i>p</i>
$\log(S_{fp})$	1	199.725	199.725	580.082	< 0.001
C	2	6.937	3.469	10.075	< 0.001
$\log(S_{fp}):C$	2	0.966	0.483	1.403	0.2471
residuals	404	139.099	0.344		

Having selected an appropriate regression model, we continue by testing the hypothesis (6) that process maturity influences on productivity with ANOVA, the results of which can be seen in Table 3.

From the ANOVA we can see that both size ($\log(S_{fp})$) and process maturity (C) have a significant effect on the effort ($\log(E_{hr})$). Furthermore we can observe that there is no interaction between size and process maturity, which means that software process improvement has a similar (positive) effect on both large and small projects. If we examine the regression coefficients, we arrive at the following relations between effort and size, which overall is a 20.86% improvement of productivity for CMM level 2 & 3 projects over CMM level 1 projects.

$$\begin{aligned}
 \text{CMM level 1: } E_{hr} &= 31.68 \cdot S_{fp}^{0.80} \\
 \text{CMM level 2: } E_{hr} &= 26.35 \cdot S_{fp}^{0.80} \\
 \text{CMM level 3: } E_{hr} &= 18.93 \cdot S_{fp}^{0.80}
 \end{aligned} \tag{11}$$

If we look at the explained variance, we see an $R^2 = 0.60$. This means that 60% of the variation in effort can be explained by process maturity and size. This leave less chance that the results can be explained in an alternative way.

4.4 Hierarchical model approach

In this section we examine the effects of software process improvement with hierarchical linear models. In the previous section we established that that the log-scaled model best fits the data ($\log(E) = \beta_0 + \beta_1 \log(S) + \beta_2 C + R$). In a similar manner we examine the influence of domain on the regression coefficients ($\beta_{1...3}$) in Table 4. This table contains the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) [21] and the log-likelihood of each model.

From Table 4 we see that model $\log(E) = \beta_{0j} + \beta_1 \log(S) + \beta_2 C + R$ has the lowest AIC and therefore is the best balance between goodness-of-fit and number of parameters. If we compare the likelihoods with the optimal regression model from the previous section, we obtain that the hierarchical linear model is significantly better (log-likelihood ratio=22.27645, $p < .0001$). Although some

Table 4. Diagnostic Information on Hierarchical Linear Models.

<i>Formula</i>	AIC	BIC	<i>Log-likelihood</i>
$\log(E) = \beta_0 + \beta_1 \log(S) + \beta_2 C + R$	733.17	753.25	-361.58
$\log(E) = \beta_{0j} + \beta_1 \log(S) + \beta_2 C + R$	712.89	736.90	-350.45
$\log(E) = \beta_{0j} + \beta_{1j} \log(S) + \beta_2 C + R$	715.94	748.07	-349.97
$\log(E) = \beta_{0j} + \beta_1 \log(S) + \beta_{2j} C + R$	718.60	762.77	-348.30
$\log(E) = \beta_{0j} + \beta_{1j} \log(S) + \beta_{2j} C + R$	725.92	786.16	-347.96

other hierarchical linear models have an even lower log-likelihood, this difference is not significant.

When we obtain an ANOVA on model $\log(E) = \beta_{0j} + \beta_1 \log(S) + \beta_2 C + R$ to test whether process maturity has an influence on productivity, hypothesis (9), we obtain the results as shown in Table 5:

Table 5. ANOVA on Hierarchical Linear Model ($\log(E) = \beta_{0j} + \beta_1 \log(S) + \beta_2 C + R$).

	<i>num df</i>	<i>den df</i>	<i>F</i>	<i>p</i>
$\log(S_{fp})$	1	369	630.06	< 0.001
C	2	369	8.06	< 0.001

So, significant effects of process maturity on productivity are not only found if we analyse the data with linear regression models, but also if we analyse the data using hierarchical linear models. As hierarchical linear models take the impact of both size and organisation into account, we rejected that these obvious alternative explanations explain the change in productivity instead of software process improvement. Taking organisation into account when analysing the data increases the explained variance from 60% to 67% ($R^2 = 0.67$). This increases the confidence in our results.

If we examine the regression coefficients, we obtain an 23.42% overall productivity increase for CMM level 2 & 3 organisation when compared with a CMM level 1 organisation. Examining the regression coefficients leads to the following, organisation-wide models for productivity:

$$\begin{aligned}
 \text{CMM level 1: } E_{hr} &= 33.09 \cdot S_{fp}^{0.82} \\
 \text{CMM level 2: } E_{hr} &= 26.68 \cdot S_{fp}^{0.82} \\
 \text{CMM level 3: } E_{hr} &= 20.40 \cdot S_{fp}^{0.82}
 \end{aligned} \tag{12}$$

5 Conclusions

From the study we have found clear evidence that CMM does increase the productivity of an organisation. We found a productivity increase of 20%. More planning and more attention to management and work processes do seem to have a positive effect on the productivity of the organisation. The improvements made in this study are smaller than found in certain similar studies, but we believe that this can be explained because in some studies small convenience samples are analysed instead of the productivity data on all projects in that organisation.

In addition we found that the classical method of comparing productivity indices has a lot of disadvantages, as only a tiny part of the variance can be explained by the maturity level. By using more sophisticated statistical techniques, linear regression models and hierarchical linear models, we gain confidence in the results of the analysis as the underlying assumptions of the analytical techniques are met and the influence of alternative explanations for the change in productivity are excluded. Linear regression models allow us to exclude the impact of project size on changing productivity and hierarchical linear models allow us to exclude the impact of the department or organisational unit in which the project takes place. It gives confidence in the results that the increases found in productivity using both statistical methods are approximately equal.

Acknowledgements

We would like to thank Jean Kleijnen, Marcel Uleman, Ton Groen for their insights into the SPI program and their help in obtaining the required data. Frank Harmsen's guidance and supervision at the beginning of the project is greatly appreciated, as is Geurt Jongbloed's helpful statistical advice.

References

1. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-24, DTIC ADA263403, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA (1993) Available from: <http://www.sei.cmu.edu/>.
2. ISO/IEC: Information technology - software process assessment. Technical Report ISO/IEC TR 15504:1998, International Organization for Standardization/ International Electrotechnical Commission (1998)
3. CMMI Product Development Team: CMMI for systems engineering/software engineering/integrated product and process development/supplier sourcing, version 1.1 continuous representation. Technical Report CMU/SEI-2002-TR-011, ESC-TR-2002-011, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA (2000) Available from: <http://www.sei.cmu.edu/>.
4. Galin, D., Avrahami, M.: Do SQA programs work – CMM works. A meta analysis. In Amir Tomer, R., Schach, S.R., eds.: Proceedings of the International Conference on Software - Science, Technology & Engineering (SwSTE'05), Washington, DC, USA, IEEE Computer Society Press (2005) 95–100

5. Stapleton, J.: Framework for Business Centred Development: DSDM Manual version 4.1. DSDM Consortium, Ltd., Kent, United Kingdom. (2002)
6. Peter, J., Kutner, M.H., Nachtsheim, C.J., Wasserman, W.: Applied Linear Statistical Models. 4 edn. WCB/McGraw-Hill, Boston, MA, USA (1996)
7. Bryk, A.S., Raudenbush, S.W.: Hierarchical Linear Models: Applications and Data Analysis methods. 1st edn. Volume 1 of Advanced Quantitative Techniques in the Social Sciences. Sage Publications, Newbury Park, CA, USA (1992)
8. Diaz, M., King, J.: How CMM impacts quality, productivity, rework, and the bottom line. *CrossTalk: The Journal of Defense Software Engineering* **15**(3) (2002) 9–14
9. Diaz, M., Sligo, J.: How software process improvement helped Motorola. *IEEE Software* **14**(5) (1997) 75–81
10. Wohlwend, H., Rosenbaum, S.: Software improvements in an international company. In: Proceedings of the 15th International Conference on Software Engineering (ICSE-93), Washington, DC, USA, IEEE Computer Society Press (1993) 212–220
11. Oldham, L.G., Putman, D.B., Peterson, M., Rudd, B., Tjoland, K.: Benefits realized from climbing the CMM ladder. *CrossTalk: The Journal of Defense Software Engineering* **12**(9) (1999) 7–10
12. Conte, S.D., Dunsmore, H.E., Shen, V.Y.: Software Engineering Metrics and Models. Benjamin/Cummings Publishing, Menlo Park, CA, USA (1986)
13. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach. 2nd edn. International Thomson Computer Press, London, UK (1998)
14. Verhoef, C.: Quantifying software process improvement. Technical report, Vrije Universiteit Amsterdam, Amsterdam, NL (2005) Available from: <http://www.cs.vu.nl/~x> [Accessed: 05/12/2005].
15. Cook, T.D., Campbell, D.T.: Quasi-Experimentation: Design & Analysis Issues for Field Settings. Rand McNally College Publishing Company, Chicago, IL, USA (1979)
16. Albrecht, A.J.: Measuring application development productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Applications Development Symposium. (1979) 83–92
17. Furey, S.: Point: Why we should use function points. *IEEE Software* **14**(2) (1997) 28–30
18. Bhattacharyya, G.K., Johnson, R.A.: Statistical Concepts and Methods. Wiley Series in Probability and Statistics. Wiley-Interscience, New York, NY, USA (1977)
19. Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D.J., Steec, B.: Software Cost Estimation with Cocomo II. Prentice-Hall PTR, Upper Daddle River, NJ, USA (2000)
20. Lindley, D.V., Smith, A.F.M.: Bayes estimates for the linear model. *Journal of the Royal Statistical Society* **34**(1) (1972) 1–41
21. Pinheiro, J.C., Bates, D.M.: Mixed Effects Models in S and S-Plus. 1st edn. Statistics and Computing. Springer-Verlag, Berlin, D (2000)
22. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). *Biometrika* **52**(3 & 4) (1965) 591–611