Effort Estimation:
Software Effort Estimation by Analogy and "Regression Toward the Mean"

Magne Jørgensen[i], Ulf Indahl[ii], Dag Sjøberg[i]
i) Simula Research Laboratory, Oslo, Norway
ii) Teknometri as, Oslo, Norway,

Abstract
Estimation by analogy is, simplified, the process of finding one or more projects that are similar to the one to be estimated and then derive the estimate from the values of these projects. If the selected projects have an unusual high or low productivity, then we should adjust the estimates toward productivity values of more average projects. The size of the adjustments depends on the expected accuracy of the estimation model. This paper evaluates one adjustment approach, based on the findings made by Sir Francis Galton in the late 1800s regarding the statistical phenomenon "regression toward the mean" (RTM). We evaluate this approach on several data sets and find indications that it improves the estimation accuracy. Surprisingly, current analogy based effort estimation models do not, as far as we know, include adjustments related to extreme analogues and inaccurate estimation models. An analysis of several industrial software development and maintenance projects indicates that the effort estimates provided by software professionals, i.e., expert estimates, to some extent are RTM-adjusted. A student experiment confirms this finding, but also indicates a rather large variance in how well the need for RTM-adjustments is understood among software developers.

Keywords: Effort estimation, estimation by analogy, regression toward the mean.

1 Introduction

In the late 1800s Sir Francis Galton (Galton, 1997) measured the height of children and their parents. He found that, on average, short parents had children that were not as short as themselves, and that tall parents had children that were not as tall as themselves. Initially, he believed that there was some sort of biological force that made people's height move toward the mean, and he called that force "regression". There was, however, a problem if this force were biological; after a few generations all people would have had the same height, i.e., the regression toward the mean (RTM) could not be biological. Galton concluded that this regression toward the mean was a statistical phenomenon and an inherent feature of imperfect correlation, i.e., less than 100% correlation between two variables. The causes of an imperfect correlation could be random variation, measurement error, non-linearity or change of characteristics over time (Campbell & Kenny, 1999).

There are studies (Kahneman & Tversky, 1973, Nisbett & Ross, 1980) demonstrating that the RTM phenomenon in real life situations can be large and that people tend to overlook it. None of these studies were carried out in a software development context, which is the scope of this paper. More specifically, this paper focuses on the RTM phenomenon in estimation of software project effort using analogy based approaches. This focus includes informal expert estimation by analogy and the use of analogy based estimation models. The research questions are:

- Do RTM-adjustments improve analogy based software effort estimation models?
- Are software developers aware of the need for RTM-adjustments when the selected analogues are extreme and the estimation model is inaccurate?

This paper is organized as follows. Section 2 describes the estimation by analogy process, explains RTM, and exemplifies how it may impact the accuracy of the analogy based effort estimates. Section 3 evaluates the benefits of RTM-adjustments. Section 4 analyses the awareness of RTM when experts estimate software project effort. Section 5 summarizes and describes further work.

2 Estimation by analogy and RTM

From a study of nearly 600 organizations Heemstra (1992) reports that estimation by analogy is the most common estimation method in the software industry. Estimation by analogy is, simplified, the process of finding one or more projects that are similar to the one to be estimated and then derive the estimate from the values of these projects. Estimation by analogy can, for example, be performed as

- pure expert estimation (the "database" of previous projects is in the expert's head),
- expert estimation informally supported by a database containing information about finished projects, or
- estimation based on the use of a clustering algorithm to find similar projects.

The estimation models Optimized Set Reduction (Briand *et al.*, 1992), Angel (Shepperd & Shofield, 1997) and ACE (Walkerden & Jeffery, 1999) are examples of formal analogy based estimation models based on clustering algorithms.

To understand the relevance of RTM for effort estimation by analogy, it is important to acknowledge the randomness inherent in every software development project. This randomness is caused by uncertain events that cannot, or are very hard to, be predicted, e.g., different degrees of turnover or illness by key project members. A consequence of this project randomness is that even projects with quite similar characteristics must be expected to have different productivity values. Assume, for example, that 81 projects with similar characteristics have been completed and that the productivity (in work-hours per function point) is distributed as shown in Figure 1. The mean productivity of the projects is 17.5 work-hours per function point.

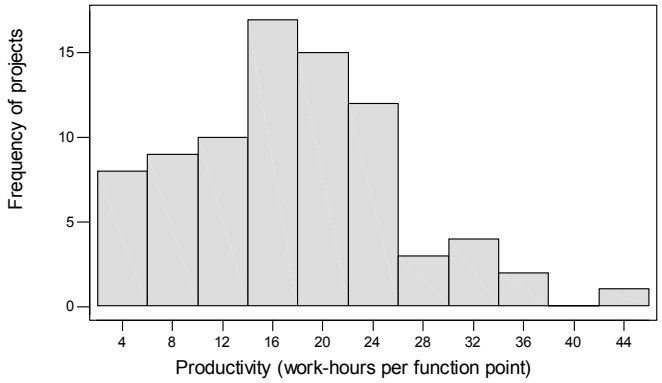

Figure 1 - Example productivity distribution

Some of the variation of the project effort is caused by the known differences between the projects. However, it is likely that the random factors and events have been biased toward a positive impact on productivity for the projects with higher productivity than the mean value and a negative impact for the projects with lower productivity than the mean value. Informally, the degree of "luck" is, on average, higher for the projects with high productivity than for the projects with low productivity values.

Assume that three previous projects (out of the 81 similar projects) are identified as the closest analogues to a new project. Those three projects have, on average, a productivity of 5.7 work-hours per function point. What should the new project use as its productivity estimate? 5.7 (the unadjusted value), 17.5 (the mean productivity) or some other value?

Which productivity value to use depends mainly on two variables: The *expected accuracy* of the estimation model based on the three closest analogues and the *extremeness* of the selected analogues. The importance of these two variables can be illustrated through an analysis of the optimal estimation strategy when the estimation model is very accurate (1) and when the estimation model is very inaccurate (2):

(1) Assume that the use of the three closest analogues so far has resulted in very accurate estimates. Then, the productivity of these analogues should be used unadjusted. In our example this means an estimated productivity of 5.7 work-hours per function point.

(2) Assume that the correlation between the actual and the estimated productivity, based on the use of the three closest analogues, is very low. Then, it is reasonable to use a productivity value close to the average productivity of all the projects as our best guess of the productivity. In our example this means a value close to 17.5 work-hours per function point. If the selected analogues have

productivity values close to the average of all the projects, then the change in value will be minor. If, however, the selected analogues have an extremely low or high productivity, as in our example, the need for adjustment of productivity estimates compared with the unadjusted value may be very large, i.e., the size of the adjustments depends on the extremeness of the selected analogues.

An important consequence of the need for adjustment is that the variance of the estimated productivity should be identical to the variance of the actual productivity only in the case of a very accurate estimation model. In all other cases the distribution of estimated productivity should be much narrower than the distribution of the actual productivity.

The adjustment method we investigate in this paper is based on the formula developed by Galton (1997) and elaborated in (Campbell & Kenny, 1999). Applying it in our context results in the following formula:

F1: RTM-adjusted productivity = PCA + (M – PCA)(1 – c), where*

PCA = Productivity of closest analogues, e.g., the two most similar projects
M = Mean productivity of similar projects
c = Correlation between productivity of closest analogues and actual productivity

This formula is based on the use of the:

- *distance* to the mean productivity of similar projects (M – PCA) as a measure of the extremeness of the selected analogues. This group of similar projects may for example consist of all projects using the same development environment and should typically contain a higher number of projects than the group of closest project analogues, and,
- *historical correlation* (c) between the non-adjusted analogy based productivity estimates and the actual productivity as a measure of the expected estimation accuracy.

In our example, assuming that the correlation (*c*) between the closest analogue based productivity estimates and the actual productivity is 0.5, the RTM-adjusted productivity equals:

RTM-adjusted productivity = 5.7 + (17.5–5.7)(1–0.5) = 5.7 + 5.9 = 11.6 work-hours per function point*

Consequently, in our example, the use of the mean value of the three closest analogues (5.7) would lead to a, on average, much too optimistic effort estimate, while the overall average (17.5) would be too pessimistic.

The use of the RTM formula in a software estimation context is not straightforward. For example, a meaningful use of it requires that the projects included in the calculation of the correlation *c* are representative for future estimation accuracy, that the group of similar projects regress toward the same average value *M*, and that this value is known.

In addition, a critical assumption for the goodness of the formula is that the correlation coefficient is a meaningful measure of the estimation accuracy. This may, for example, not be the case if the relationship between the predicted and the actual productivity is far from linear or a few very high values have a large impact on the correlation coefficient. Furthermore, if the productivity systematically is estimated too low or too high then a high correlation does not mean the same as high estimation accuracy. For these reasons, we evaluate the usefulness of the RTM formula using more traditional accuracy measures, such as MMRE and MdMRE on the resulting effort estimates (Conte *et al.*, 1986, Jørgensen, 1995).

In software development projects where a high degree of unexplained randomness is not unusual, we may frequently have situations where RTM-adjustments lead to significant different productivity and effort estimates compared with the closest analogues estimation method. However, a discussion in the literature on how to utilize RTM-adjustments in analogy based software estimation models seems missing. Selecting the closest analogue and using it to predict a new project is clearly not a recommended strategy if it has an extreme productivity value and the estimation model is not very accurate.

This need for adjustments corresponds to the well-known problem of over-fitting relatively small data sets in statistical modelling. Regularization (shrinkage), outlier detection (and elimination) and robustification (Rousseeuw & Leroy, 1986, Sen & Srivastava, 1990) to stabilize the estimated model-parameters and models often improve their predictions. The degree of improvement depends on how successfully we manage the trade-off between bias and stability of the actual models. This is conceptually also the case with RTM, where the contribution against unstable over-fitting is reflected through simple weighting toward the overall mean of the similar projects. More details on RTM can be found in (Campbell & Kenny, 1999).

3 Evaluation of RTM-adjustments

In this section we evaluate the benefits of adjusting for the RTM-effect in analogy based estimation. For this purpose we use the data set described in (Jeffery & Stathis, 1996) (data set A) and (Jørgensen, 1997) (data set B). These data sets were chosen because the use of analogy based estimation models performed well on these data sets, i.e., the use of an analogy based estimation model would be a natural choice. In addition, we briefly describe the evaluation carried out on three other data sets[1] (data set C, D and E).
Our evaluation approach includes the following steps:

1. Develop an analogy based effort estimation model for each data set.
2. Use the model to estimate the unadjusted and RTM-adjusted productivity and effort for each project. Calculate the mean and median unadjusted and RTM-adjusted estimation accuracy.
3. Compare the adjusted and the unadjusted accuracy values. Compare the *theoretical* and the *actual* accuracy improvement when adjusting for RTM. Typically, the actual mean improvement will be

[1] Data set C contains projects from the MSc thesis of J. M. Desharnais, data set D contains projects from Finnish companies, and data set E contains projects made available to the Mermaid Esprit project by a UK bank.

lower than the theoretical improvement due to violation of the underlying RTM formula assumptions, see (Campbell & Kenny, 1999). The size of the deviation between the theoretical and the actual improvement may indicate the potential for improvement of the RTM-adjustments.

Notice that this approach uses the same data sets for development and evaluation of the estimation models. A more correct approach would be, for example, to leave the project to be estimated out of the data set and then develop and use the estimation model based on the remaining projects, i.e., using a cross-validation approach (Bradley & Gong, 1983). Development and evaluation of the model on the same data set clearly has the risk of over-fitting the model to the data. We evaluated whether the "leave one out" approach would change the analogy based estimation models, but found only small changes. For this reason, we decided to use the simplified approach described above. Of course, the project to be estimated is never included in the set of its closest analogues.

3.1 Data set A

3.1.1 Estimation model

One of the projects in data set A (Jeffery & Stathis, 1996) is more than twice as large as the second largest project. In practice, this project would not be very useful as an analogue for other projects. Hence, we removed it from the data set.

The data set contains the variables: Programming LAnguage (C = 1, Cobol = 2, Mixed = 3, 4GL = 4), ARChitecture (Centralized = 1, Distributed = 2), SIZE (Unadjusted Function Points) and Actual Effort. Productivity is calculated as Effort/SIZE. The data set is given in Table 1.

ID	PLA	ARC	SIZE	Effort	Productivity
1	4	1	1164	3777	3.2
2	4	1	1834	4389	2.4
3	4	1	388	1647	4.2
4	2	1	336	1318	3.9
5	2	1	116	529	4.6
6	2	1	182	691	3.8
7	2	1	65	291	4.9
8	2	1	160	448	2.8
9	2	1	185	262	1.4
10	1	1	168	415	2.5
11	1	2	422	2070	4.9
12	1	2	296	1947	6.6
13	1	2	129	1500	11.6
14	2	2	143	1114	7.8
15	3	2	38	362	9.5
16	3	2	89	921	10.4

Table 1 - Data set A

The analogy based estimation model we developed uses the Euclidan distance to measure how close two projects are to each other. The Euclidian distance is measured as:

$\sqrt{(\Sigma_j(x_{ij} - x_{kj})^2)}$, *where* x_{ij} *is the value of project* i *for variable* j.

Consequently, the closest analogue to a project *p* is the project with the minimum Euclidian distance to *p*.

We used the average productivity of the closest analogues as the estimated productivity of the project to be estimated. The estimated effort is then derived from the productivity:

F2: $Estimated\ Effort = SIZE_{TARGET} * Productivity_{ANALOGUES}$

This is, in effect, the same approach as the linear adjustment due to differences in project size applied in (Walkerden & Jeffery, 1999). For example, assume that the productivity of the closest analogue is 10 work-hours per function point, and that the size of the "target" project, i.e., the project to be estimated, is 80 function points. Then, the estimated effort is 80 * 10 = 800 work-hours.

3.1.2 Estimation, evaluation and comparison

The best performance of the analogy based estimation model was achieved using the effort of the closest analogue. Inclusion of two or three projects in the set of closest analogues did not improve the estimation accuracy.

The closest analogue (CA), CA productivity, CA effort value and CA estimation accuracy are displayed in Table 2. The MMRE (mean MRE) and MdMRE (median MRE) of closest analogue effort estimates are 0.39 and 0.36, respectively.

When calculating the RTM-adjusted productivity and effort, we first calculated the correlation (c) between the productivity of the closest analogue and the actual productivity. That correlation was 0.3. A simple analysis of the data set showed that the mean productivity was much lower for projects with distributed architecture (7.8 work-hours per function point) than projects with centralized architecture (2.9 work-hours per function point). It was unlikely that projects of different architecture would "regress" toward the same mean productivity value. Therefore, we decided to use different mean productivity values for these two groups.

The estimated productivity was calculated using formula F1; the estimated effort was calculated using formula F2 (size measured in function points). The RTM-adjusted productivity and effort, and the estimation accuracy of the RTM-adjusted effort are also displayed in Table 2.

For example, the estimated productivity of project 8, which was a centralized architecture project (mean productivity 2.9), was calculated using F1:

*3.8 + (2.9 – 3.8) * (1 – 0.3) = 3.2 work-hours per function point*

The corresponding RTM-adjusted effort is: *3.2 * 160 = 512 work-hours.* The actual effort was 448 work-hours and the CA effort estimate was 607 work-hours.

The MMRE and MdMRE of the RTM-adjusted effort estimates are 0.31 and 0.26, respectively, i.e., a better accuracy compared with the unadjusted estimates 0.39 and 0.36. A paired, one-sided, t-test on the difference between the MMRE of closest analogue and RTM-adjusted MMRE gives a p-value of 0.07, i.e., the improvement seems statistically significant.

Id	Id of closest analogue (CA)	CA Productivity	CA Effort	CA MRE	RTM Productivity	RTM Effort	RTM MRE
1	2	2.4	2786	0.26	0.36	3209	0.15
2	1	3.2	5951	0.36	0.33	5561	0.27
3	4	3.9	1522	0.08	0.31	1236	0.25
4	9	1.4	476	0.64	0.45	748	0.43
5	8	2.8	325	0.39	0.34	337	0.36
6	9	1.4	258	0.63	0.45	405	0.41
7	5	4.6	296	0.02	0.30	214	0.26
8	6	3.8	607	0.36	0.32	506	0.13
9	6	3.8	702	1.68	0.32	585	1.23
10	8	2.8	470	0.13	0.34	488	0.18
11	12	6.9	2776	0.34	0.14	3111	0.50
12	11	4.9	1452	0.25	0.15	1958	0.01
13	12	6.6	849	0.43	0.14	951	0.37
14	13	11.6	1663	0.49	0.12	1234	0.11
15	16	10.4	393	0.09	0.12	319	0.12
16	15	9.5	848	0.08	0.12	732	0.20
			MMRE	**0.39**		**MMRE**	**0.31**
			MdMRE	**0.36**		**MdMRE**	**0.26**

Table 2 -The estimates of data set A

A measure of the theoretical improvement of the productivity when adjusting for RTM (TImpProd) is the RTM-adjusted distance toward the mean, i.e.:

F3: TImpProd = abs(M – PCA)(1–c)*

TImpProd is a function of extremeness of the closest analogues, i.e., (M – PCA) and the inaccuracy of the closest analogue based estimation model, i.e., (1-c). The more extremeness and inaccuracy, the higher expected improvement.

In our data set the mean TImpProd was 0.94. This means that if the all the RTM-adjustment assumptions were met, then the average project would be estimated 0.94 work-hours per function point more correctly than without the RTM-adjustments. The actual mean improvement in the productivity estimate was, however, not more than 0.29. This is not surprising. It is not probable that we have selected productivity values that we could expect the productivity of *all* projects to regress toward. More project data and more knowledge about the organization and the projects may improve this situation. Nevertheless, it seems unrealistic to achieve values close to the theoretical improvement level. For example, if the developers learn from experience, then the productivity average is moving. Theoretically, this can be included in the RTM-adjustments (Campbell & Kenny, 1999), but in practice it is difficult to formalize the level of learning.

3.2 Data set B

3.2.1 Estimation model

The productivity of the projects belonging to the data set described in (Jørgensen, 1997) differs very much in productivity depending on the use of development tools. To make the data set meaningful for estimation by analogy, we conducted the evaluation only on the projects developed either in a mainframe/Cobol environment or in PC/PowerBuilder environment. Only these two types of development environments had enough projects to establish meaningful analogy based models. The data set is displayed in Table 3.

Id	Development environment	Size (function points)	Effort	Productivity
1	Cobol	59.4	594	10.0
2	Cobol	71.5	512	7.2
3	Cobol	78	572	7.3
4	Cobol	120	279	2.3
5	Cobol	140	934	6.7
6	Cobol	140.6	548	3.9
7	Cobol	191.9	841	4.4
8	Cobol	446.6	1577	3.5
9	Cobol	523.9	1341	2.6
10	Cobol	907.5	10244	11.3
11	PowerBuilder	298.2	729	2.4
12	PowerBuilder	487.3	1696	3.5
13	PowerBuilder	645.7	1570	2.4
14	PowerBuilder	686.4	2785	4.1
15	PowerBuilder	737.1	816	1.1
16	PowerBuilder	780	816	1.1
17	PowerBuilder	933	3100	3.3
18	PowerBuilder	952	2531	2.7
19	PowerBuilder	1470.4	1946	1.3
20	PowerBuilder	2064.3	6084	23.0

Table 3 - Data set B

The analogy based estimation model applied the following simple rule to find the closest analogue:

Find the project using the same development environment with the most similar size.

This rule was motivated by the observation that there was a significant, although not very high, correlation between productivity and size (correlation = 0.35, p = 0.1).

The productivity of the closest analogue was then used as the estimated productivity. Multiplying the estimated productivity multiplied with the size of the project, similar to the estimation model on data set A, we got the estimated effort.

To calculate the RTM-adjusted productivity and effort we used the average productivity values for the Cobol environment (4.5) and the PowerBuilder projects (2.0) as the productivity values that the projects were supposed to regress toward. For example, the closest analogue of project 13, which has the size 645.7 function points, is project 14. Project 14 has the productivity of 4.1. The unadjusted estimated productivity and effort values are, accordingly: 4.1 work-hours per function point and approximately 2650 (4.1 * 645.7) work-hours.

3.2.2 Estimation, evaluation and comparison

Table 4 shows the resulting productivity and effort estimates, and the corresponding accuracy values for data set B. The correlation (c) between the unadjusted productivity (CA Productivity) and the actual productivity was 0.7.

Id	Closest analogue (CA)	CA Productivity	CA Effort	CA MRE	RTM Productivity	RTM Effort	RTM MRE
1	2	7.1	424	0.29	6.1	362	0.39
2	1	7.3	524	0.02	6.2	443	0.14
3	2	7.2	559	0.02	6.1	476	0.17
4	5	6.7	801	1.87	5.9	702	1.51
5	6	3.9	546	0.42	4.1	570	0.39
6	5	6.7	938	0.71	5.9	822	0.50
7	6	3.9	748	0.11	4.1	781	0.07
8	9	2.6	1143	0.28	2.9	1315	0.17
9	8	3.5	1850	0.38	3.8	1982	0.48
10	9	2.6	2327	0.77	3.0	2676	0.74
11	12	3.5	1028	0.41	2.8	846	0.16
12	11	2.4	1185	0.30	2.3	1114	0.34
13	14	4.1	2620	0.67	3.1	2005	0.28
14	13	2.4	1669	0.40	2.3	1569	0.44
15	16	1.1	771	0.06	1.2	900	0.10
16	15	1.1	863	0.06	1.3	998	0.22
17	18	2.7	2480	0.20	2.4	2260	0.27
18	17	3.3	3163	0.25	2.8	2643	0.04

19	18	2.7	3909	1.01	2.4	3562	0.83
20	19	1.3	2716	0.55	1.5	3029	0.50
			MMRE	**0.44**		**MMRE**	**0.39**
			MdMRE	**0.34**		**MdMRE**	**0.31**

Table 4 - The estimates of data set B

The RTM-adjustments improved the MMRE values from 0.44 to 0.39 and the MdMRE from 0.34 to 0.31. A paired, one-sided, t-test of the difference in MMRE gave p=0.09, i.e., adjusting for RTM seems to result in a significant improvement.

In this data set the mean TImpProd was 0.42 and the actual mean improvement 0.07. Similar to data set A, this may point at a large potential for improving the RTM-adjustments and/or violation of the underlying RTM assumptions. Exploring the productivity data we found that the assumption that there is a linear relationship between the actual and the estimated productivity is reasonably well met. The main problem may, therefore, be that the productivity values we assumed that the projects would regress to, i.e., the mean productivity of the groups of Cobol and PowerBuilder projects, are not optimal. More knowledge about the projects may lead to other productivity groups leading to more meaningful mean productivity values.

3.3 Summary of evaluation

The RTM-adjusted effort estimates were significantly more accurate than the unadjusted closest analogue effort estimates for data set A and B. To validate the RTM-adjustments further, we briefly evaluated the RTM-adjustments on the data sets C, D and E using an estimation model very similar to the one used on data set B. The correlation (c) between the estimated productivity, using the closest analogue approach, and the actual productivity was very low for these data sets. We found no clear "productivity clusters", in contrast to data sets A and B. For this reason, we used RTM-adjusted productivity values very close to the average productivity of *all* projects in each of these data sets. The improvement in the MMRE was for data set D from 0.42 to 0.35 (paired, one-sided, t-test of difference gave p=0.05), for data set E 1.54 to 1.06 (p=0.12) and for data set F 2.34 to 1.5 (p=0.01). As expected, the improvement was largest when the MMRE-value was low, i.e., when the estimation accuracy was low, as in the data sets E and F.

The consistent improvement of the estimates indicates that RTM-adjustments should be carried out when using analogy based estimation models. This is particularly important when projects with extreme productivity are selected as analogues and the estimation model is inaccurate. However, also in cases with less extreme analogues and more accurate estimation models there will be an expected improvement if the underlying assumptions of the RTM-adjustments are met.

4 Expert estimation and RTM

This section analyses whether expert estimators intuitively adjust for RTM. If they do, a necessary (but not sufficient) consequence is that the standard deviation of the productivity estimates should be less

than the standard deviation of the actual productivity, i.e., there is a regression toward the mean productivity. Note that we do not focus on the *causes* of a possible regression toward the mean of expert estimates, only the correspondence between the RTM-adjustments and the expert estimator behaviour. In other words, we study whether the expert estimators, regardless of causes, behave in accordance with the RTM phenomenon.

The size of the "shrinkage" of the effort estimation standard deviation should be closely related to the degree of correlation between the predicted and the actual values; the lower prediction accuracy, the larger shrinkage of standard deviation. We empirically tested the shrinkage of the standard deviation on the only two software project data sets we found that included information about expert estimates and actual effort (Jørgensen, 1997, Kitchenham *et al.*, 2001). Both data sets were collected in organizations developing and maintaining rather large administrative software applications. We analyzed only the subsets containing the projects that were reported as being estimated purely by experts, i.e., without the support of estimation models. The actual expert estimation strategies in these organizations were not reported.

We had no information about the estimated productivity of the projects in these data sets. For this reason, we were unable to measure whether the experts adjusted the productivity in accordance with the RTM formula. The estimated effort is, however, a proper substitute, since a narrower productivity distribution will lead to a narrower effort distribution. As can be seen from Table 5, the expert estimates of effort (work-hours) had less standard deviation (STD-EST) than the standard deviation of the actual effort values (STD-ACT). MMRE is the mean effort estimation accuracy of the expert estimates.

Data set	STD-ACT	STD-EST	Shrinkage	MMRE
21 medium large software development projects (Jørgensen, 1997)	2.500	2.100	17%	0.19
104 small maintenance task projects (Kitchenham *et al.*, 2001)	11.200	7.900	30%	0.24

Table 5 - Shrinkage of standard deviation

This shrinkage indicates an RTM-adjustment included in the expert estimates. Furthermore, a larger shrinkage in standard deviation in the data set corresponds with reduced prediction accuracy measured as MMRE, i.e., a difference in accordance with the RTM formula.

In other words, the experts' estimates regress toward the mean, without necessarily having explicit knowledge about the RTM formula. It should be no surprise that experts to some extent adjust for the RTM effect. For example, the project leader may know that the most similar project to the one to be estimated reduced its use of effort because of some very unusual situation. The project leader, then, would then not use the unadjusted effort of this project to estimate the effort of new projects. In addition, the project leader estimates may be influenced by knowledge about typical projects. The use of typical projects as input in the estimation process will lead to RTM-adjustments. Similar shrinkage of standard

deviation has been found in other domains. For example, Attneave (1953) reports a study where letter frequencies were estimated and compared with actual values. That study reports data that implies shrinkage of estimation standard deviation of approx. 50%. This large shrinkage compared with the shrinkage in Table 5 can be explained by the much higher estimation inaccuracy in that study. The mean MRE in that study was 2.1, which is in accordance with the RTM theory. In general, it seems to be a well-established phenomenon that low frequencies tend to be overestimated and high frequencies tend to be underestimated, which is in accordance with the RTM theory. See (Sedlmeier *et al.*, 1998) for an overview of frequency judgements under uncertainty.

This means that while most current analogy based software effort estimation models do not adjust for RTM, our results indicate that expert estimators do! This may be one of the reasons for the finding described in (Walkerden & Jeffery, 1999) where software project analogues selected by people were better than the analogues selected by their analogy based estimation tool (ACE).

Although expert estimators on average adjust for RTM, one should not overlook the possibility of a substantial difference in awareness of the need for this adjustment. To study how different adjustments for extremeness are made, we designed a small estimation experiment using 38 under-graduate students following a course in software engineering at the University of Oslo. The students were asked to estimate the productivity of a new project (P-NEW). They were given information about P-NEW and the results of a selection of the two most similar projects (P-11 and P34) from a project database, see Table 6 (kLOC = 1000 Lines of code, MM = Man Months). The project data and the project database characteristics were invented by us for the purpose of the experiment, i.e., they were not based on real project data. The use of invented projects is not an important limitation of our experiment since we study RTM adjustments, not estimation accuracy. We provided no formal definitions of the variables, and relied on the intuitive understanding of the values, e.g., high complexity. As far as we experienced, none of the participants felt this as a problem.

Project	Platform	Architecture	Size	Complexity	Productivity
P-11	Cobol/ MVS	Client/server	9 kLOC	HIGH	300 LOC/MM
P-34	Cobol/ MVS	Client/server	11 kLOC	HIGH	1100 LOC/MM
P-NEW	Cobol/ MVS	Client/server	10 kLOC (estimate)	HIGH (estimate)	?

Table 6 - Project information used in the experiment

The following information about the projects in the project database was presented:

- The project database consists of 100 software development projects.
- The projects vary in productivity from 200 LOC/man-months to 7000 LOC/man-months.
- The average productivity of all 100 projects is 2000 LOC/man-months.

- The average productivity of the ten projects with development platform and complexity identical to P-NEW is 1000 LOC/man-months (these are the two factors considered most important for the productivity).

Then, the students were asked: *What would you (based on the available information) suggest as productivity estimate for P-NEW? Explain how you derived this estimate.*

One of the intentions with this experiment was to study how the students used the information about the project with extremely low productivity (P-11). As there was obviously no clear correct solution on this question, we were mainly interested in the differences in how much the estimates regressed toward the mean productivity.

The estimation processes used by the students were classified into 5 strategies:
1) No regression toward the mean (use of the 2-closest analogues)
2) Minor regression toward the closest cluster mean (use of the 2-closest analogues and a minor adjustment toward the 10-closest analogues)
3) Major regression toward the closest cluster mean (use of the average of the 10-closest analogues and a minor adjustment for P-11)
4) Use of closest cluster mean (unadjusted use of the average of the 10-closest analogues)
5) Use of total average (use of the average of the 10-closest analogues and an adjustment for the total average)

To identify the strategy of each student, both the estimate itself and the explanation were taken into account. Table 7 shows the distribution of the strategies and the corresponding average estimated productivity.

Strategy	# Students	Average productivity estimate
1	8 (21%)	750 LOC/MM
2	1 (3%)	800 LOC/MM
3	7 (18%)	886 LOC/MM
4	17 (45%)	1029 LOC/MM
5	5 (13%)	1720 LOC/MM

Table 7 - Estimation strategies

The results are in accordance with the results in Table 5 and indicate that most effort estimates regressed toward the mean productivity. There was, however, a great variety in how much the estimates regressed toward the mean and which mean they regressed to.

As much as 21% (8 students) of the estimates had no regression toward the mean. To investigate the "robustness" of this strategy, we had added a second question in the experiment:

Assume that the estimation accuracy based on the 2-3 most similar projects has, on average, been very low. Does this information have an impact on the productivity estimate? Explain why/why not?

The purpose of this question was mainly to study how much this information changed the strategies of the students belonging to strategy 1 (no regression toward the mean). We found that only one of the eight students using strategy 1 in the first question changed to another estimation strategy due to the information in the second question. The other seven students answered that the uncertainty had no impact on the estimate. Their choice of strategy was, thus, robust even to the somewhat leading question formulation above.

While other studies, e.g., (Kahneman & Tversky, 1973), have found that people tend to overlook the RTM phenomenon, our experiment indicates that the students in their estimates intuitively account for the RTM phenomenon. There may be small differences between computer science students and software professionals in how they compensate for extreme analogues and uncertain predictions. Most software professionals do not, as far as we have experienced, get systematically trained in software estimation and many do not learn much from experience due to poor feedback *(Jørgensen et al., 2000)*. In addition, RTM impacts similar to those addressed in this experiment can be found in other tasks than software productivity or effort estimation. For example, assume that a football team played an outstanding match recently and that you want to predict the outcome of the next match. Clearly, you should use the information from the performance of the last match, but it would be optimistic to expect the football team to play just as outstanding in the following match.

More information about the experiment and the data set can be downloaded from http://www.ifi.uio.no/forskning/grupper/isu/forskerbasen/.

5 Summary and further work

The regression toward the mean (RTM) represents a statistical effect relevant for estimation by analogy. In particular, the RTM-effect is important when the selected analogues are extreme and the estimation model inaccurate. The studies described in this paper contribute to the answer of the following two research questions:

- Do RTM-adjustments improve analogy based software effort estimation models?
- To which degree are software estimators aware of the need for RTM-adjustments?

Through evaluation of five data sets we demonstrated that effort estimates can be significantly improved through RTM-adjustments. Current analogy based software estimation models do not include RTM-adjustments. Hence, RTM-adjustments may improve the estimation accuracy of these estimation models.

An analysis of industrial software projects indicated that expert estimates are RTM adjusted. An experiment we conducted on computer science students supported this finding. The individual

differences should, however, be expected as rather large. The experiment showed that there was a substantial variance in how the estimates were derived.

To support software professionals when estimating effort or productivity, we need to understand the strategies of the expert estimators better. This way we may be able to understand when the estimators need support from statistical tools and/or estimation models. We have started several studies in this direction. One of them investigates how the effort estimates based on group processes impact the degree of RTM-adjustments.

Acknowledgements:
The research project is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry). We are grateful to Barbara Kitchenham, University of Keele, who provided several useful data sets for this research.

References:

Attneave, F., 1953. Psychological probability as a function of experienced frequency, *Journal of Experimental Psychology,* 46(2), pp. 81-86.

Bradley, E. & Gong, G., 1983. A leisurely look at the bootstrap, the jackknife, and cross-validation, *The American Statistican,* 37(1), pp. 36-48.

Briand, L.C., Basili, V.R. & Thomas, W.M., 1992. A pattern recognition approach for software engineering data analysis, *IEEE Transactions on Software Engineering,* 18(11), pp. 931-942.

Campbell, D.T. & Kenny, D.A., 1999. *A primer on regression artifacts,* The Guilford Press.

Conte, S.D., Dunsmore, H.E. & Shen, V.Y., 1986. *Software engineering metrics and models,* Benjamin/Cummings Publishing Company Inc.

Galton, F., 1997. *Natural Inheritance,* New Mexico, Genetics Heritage Press (originally published in 1889 by Macmillan and Company).

Heemstra, F.J., 1992. Software cost estimation, *Information and Software Technology,* 34(10), pp. 627-639.

Jeffery, F.R. & Stathis, J., 1996. Function point sizing: structure, validity and applicability, *Empirical Software Engineering,* 1(1), pp. 11-30.

Jørgensen, M., 1995. An empirical study of software maintenance tasks, *Journal of Software Maintenance,* 7, pp. 27-48.

Jørgensen, M., 1997. An empirical evaluation of the MkII FPA estimation model, *Norwegian Informatics Conference,* Voss, Norway, pp. 7-18.

Jørgensen, M., Sjøberg, D. & Kirkebøen, G., 2000. The Prediction Ability of Experienced Software Maintainers, *4th European conference on software maintenance and reengineering,* Zürich, Switzerland, pp. 93-100.

Kahneman, D. & Tversky, A., 1973. On the psychology of prediction, *Psychological Review,* 80(4), pp. 237-251.

Kitchenham, B., Pfleeger, S.L., McColl, B. & Eagan, S., 2001. A case study of maintenance estimation accuracy, Accepted for publication in Journal of Systems & Software.

Nisbett, R.E. & Ross, L., 1980. *Human inference: Strategies and shortcomings of social judgment,* Englewood Cliffs, NJ: Prentice-Hall.

Rousseeuw, P.J. & Leroy, A.M., 1986. *Robust regression and outlier detection,* Wiley.

Sedlmeier, P., Hertwig, R. & Gigerenzer, G., 1998. Are judgements of the positional frequencies of letters systematically biased due to availability? *Journal of Experimental Psychology: Learning, Memory and Cognition,* 24(3), pp. 754-770.

Sen, A. & Srivastava, M., 1990. *Regression analysis – theory, methods and applications,* Springer-Verlag.

Shepperd, M. & Shofield, C., 1997. Estimating software project effort using analogies, *IEEE Transactions on Software Engineering,* 21(2), pp. 126-137.

Walkerden, F. & Jeffery, R., 1999. An empirical study of analogy-based software effort estimation, *Empirical Software Engineering,* 4, pp. 135-158.

Biography:

Magne Jørgensen received the Diplom Ingeneur degree in Wirtschaftswissenschaften from the University of Karlsruhe, Germany, in 1988 and the Dr. Scient. degree in informatics from the University of Oslo, Norway in 1994. He has 10 years industry experience as consultant and manager. He is now an associate professor in software engineering and member of the software engineering research group of Simula Research Laboratory in Oslo, Norway. His research interests include software process improvement, software maintenance, experience databases, software estimation and software measurement.

Ulf Indahl received an MSc degree in mathematics from University of Oslo in 1992 and the Dr. Scient. degree in applied mathematics from the University of Oslo, Norway in 1998. He has 3 years industry experience as consultant and Group Leader, and is for the moment leader of the analysis group in Teknometri - Software Innovation ASA. Among his research interests are pattern recognition, multivariate analysis, machine learning and fuzzy logic.

Dag Sjøberg received an MSc degree in computer science from University of Oslo in 1987 and a PhD degree in computing science from University of Glasgow in 1993. He has five years industry experience as consultant and group leader. He is now a professor in software engineering and is the leader of the research group Industrial System Development in the Department of Informatics, University of Oslo and director of the Software Engineering group of Simula Research Laboratory. Among his research interests are software evolution, software process improvement, programming environments, object-oriented methods and research methods in software engineering.