

Towards an Agile Approach to Empirical Software Engineering

Mike Holcombe, Tony Cowling and Francisco Macias

Department of Computer Science, University of Sheffield, Regent Court, Portobello Street,
Sheffield, S1 4DP, United Kingdom
{M.Holcombe, A.Cowling, F.Macias} @ dcs.shef.ac.uk

Abstract. For the last three years we have been conducting a comparative, longitudinal study of an agile software development methodology, Extreme Programming (XP). Every year the new experiment has been refined further, following standard recommendations for empirical software engineering research, but the analysis has so far proved inconclusive. Although the data suggests that the XP teams produce slightly better systems it is not statistically significant under standard analysis techniques (e.g. ANOVA or Mann-Whitney). Improving the experiments had no impact on the results of the analysis. An alternative analysis, based on Bayesian statistics has now been carried out. This provided a numerical measure of the slight superiority of XP over the design-led approach. This result has been unaffected by the tightening up of the experimental framework, causing us to question whether the heavy burden of doing this was worth the effort. As a result of these experiences we propose a more *agile* approach to studies of this nature.

1. Introduction

Software engineering is a rapidly evolving subject with new methodologies, design languages, tools, technologies and paradigms emerging every year. The market is competitive and demanding, and places great pressure on software companies to find successful ways to deliver cost effective solutions, rapidly. The dynamic nature of the business context is stretching traditional methodologies to the limit, and so there is a clear need to develop reliable and practical scientific ways to investigate the benefits of alternatives to the design methodologies that are available.

Empirical Software Engineering (ESE) aims to provide a rigorous experimental approach to investigating these issues, but it is becoming apparent that ESE itself is subject to the same pressures caused by rapid evolution. This paper explores how the discipline may respond to these pressures, based on recent work involving experiments to compare an agile methodology with more traditional software development approaches. These experiments have taken place in the “Sheffield Software Engineering Observatory” (“the Observatory” from now on).

Experiments that attempt to compare complete development methodologies are uncommon, and are extremely difficult to organize as compared to more limited studies that concentrate on a single small scale activity, such as pair programming or

code review. This is because there are so many critical parameters to be considered in “complete methodology” investigations that experiments are difficult to plan and data is often hard to validate. Even if an experiment meets the demands of the empirical guidelines of, for example [1], this does not guarantee that useful information can be derived from the experiment. We have found that traditional statistical analysis may not be very conclusive and that investing ever more resources into trying to perfect the experimental set up may not reap any dividends.

The reasons for this, as we describe later, are mainly due to human nature and the unpredictability of people’s behaviour and attitudes. This has led us to believe that guidelines for empirical research which are derived from experiences in very large clinical medical trials may be inappropriate for the subtleties of research into the benefits or problems of different software engineering methodologies, where the population of experimental subjects is much smaller and the attitudes and capabilities of these subjects critical. Yet it is information about the benefits and disadvantages of the various development approaches that is desperately needed for the software industries.

Rather in the spirit of eXtreme Programming [2] we suggest, instead, that a ‘quick and lightweight’ experiment collecting ‘sensible’ data and using a Bayesian approach could prove fruitful. This permits the analysis of the results as the experiment proceeds in order to adapt the data collection to provide a more suitable framework for the overall study. This is like the *test first* approach of XP and the *continuous delivery* of results. The ensuing reduction in bureaucracy may provide a basis for more useful and rapid results. Software engineering techniques, technologies and applications are changing so rapidly that very thorough, lengthy empirical research may produce results that are less relevant than they might be because the world has moved on.

2. The Observatory Project

The development of the Observatory has combined two aspects of the work at Sheffield. One is the role that software development projects play in the curriculum for our computing degrees [3], and in particular the projects that are run with second year undergraduates [4]. These, known as the “Software Hut” projects, require teams (typically of about four students) to each develop a small business system for an external organization. Partly to reduce the number of external clients who need to be found for such projects, and partly to increase the likelihood of the clients getting a useful product at the end of them, they are run on the basis that a number of teams (typically about six) compete to build the systems for each client. Thus, at the start of the project the client gives each of the competing teams the same basic brief, and if they come back subsequently to clarify aspects of this brief the client tries to steer each team in the same direction. At the end of the project the client then chooses whichever of the developed systems they judge to have come closest to actually meeting their requirements.

The other aspect is an interest in XP, because of the emphasis that it puts on the importance of testing throughout the development process, since testing has been the

main focus of our research group's work on the formal modeling of software systems [5, 6]. We realized that having competing teams working on the same scenarios in the software hut projects gave us an opportunity that was unique compared with industrial situations, where (apart from the special case of n-version programming) one would never set out to have a number of different teams each attempting the same project. Here, though, we could compare the performance of teams using XP against those using a more traditional approach, where each team would be working on the same scenario for the same client, and so the first of the Observatory experiments was set up. For these experiments the traditional approach was taken as one that emphasizes the production of documentation at each stage of the process, and that bases its stages essentially on the waterfall model, but typically allowing for two or perhaps even three increments. The number of increments is determined by each team on the basis of how well they will fit into the timescale of the projects, which covers 12 weeks of term time (during which the projects nominally account for one third of their time) and may spill over into the Easter vacation (which typically lasts three weeks).

Thus, the main focus of the Observatory is to conduct experimental evaluations of complete development processes. To support this we also need to measure the key parameters that characterize a complete development processes, such as the amounts of time spent by each team in the different activities that make up the process. This, of course, is extremely time-consuming if done thoroughly, and in the pilot phase of the Observatory's operation the amount of measurement work that could be done has been limited by the available staff resources. Thus, as it has become apparent that useful results could be obtained even from the limited amount of measurement work that has been possible, so proposals have been developed for a more comprehensive measurement programme, and funding has been sought to establish this.

3. The Observatory Experiments

We have conducted such experiments over the past three years, and the results have been reported in [7, 8, 9, 10]. Although the developers being observed during these experiments are students, the ways in which they have been expected to work during the development of their systems have been as close as we can get to "real-life" software development practices, since they are being used to solve "real-life" business problems for our external clients. As developers the students are comparatively inexperienced, of course, and so will make mistakes, but as this is a uniform effect across the population we do not believe that it is a significant one. Hence we consider that the results can be taken as typical of developers in general, and not just of students. This distinguishes our work from other studies of projects done by students (such as [11, 12]), where the main emphasis has been on evaluating them as elements of the educational process rather than as exemplars of software development practice.

What could be more significant are the ethical implications of observing students systematically for research purposes, while also assessing their performance for educational purposes. A detailed discussion of these ethical issues is beyond the scope of this paper, but we would observe that the key to solving them is to create a climate of trust and openness, in which the students are fully aware of what we are

trying to do and why, and are willing to contribute to this. The way in which they have volunteered information about weaknesses in the processes and in the data recording methods has convinced us that we have achieved this.

In the design of the experiments, the students form themselves into teams, in order to fit practical constraints such as their preferred patterns of work outside the formal timetable for classes, and the teams are then assigned randomly to the treatments: that is, to the development process to be used. The analysis of the experiments has then compared the quality of the final products (as measured by the clients), the quality of the processes used (as assessed by the academics managing the Observatory), the amount of time spent by the teams, and so on. The raw data from the first experiment gives the impression of slightly better results for the XP teams as compared with the traditional groups. In follow up interviews we became rather concerned about the quality of some of the data, since information from the students suggested that it might not be reliable, and we introduced substantial measures to improve the quality for the repeat of the experiment, with the next cohort of students, the following year. The results for this next year generated very similar results, a slight edge for the XP teams.

The results were of the following form, where the figures are taken from the second run of the experiment, in 2002. We wanted to know if the two populations produced systems with equal or different quality, so we ran several tests. For each, the null hypothesis was that both populations exhibit the same quality; the alternative one was that they have different quality.

We firstly ran ANOVA because the populations had broadly similar distributions (Table 1) but the result could not reject the null hypothesis. As this is considered a weak result, we then tried the Mann-Whitney test because of the (small) size of the sample, but this also could not reject the null hypothesis. Our concern over this result was that when we split the teams into blocks (clients) and compared them, XP teams reiteratively presented better quality than the traditional teams; the difference was not huge but was continuous. The previous year's experiment produced almost identical analysis, so improving the data collection and the structure of the experiment for the second run had had no discernible effect on the results.

Table 1. The two populations have similar distributions

	XP	Traditional
Size	10	10
Average	71.9	68.15
Standard Deviation	6.75	7.45
Data into 1 std. dev.	70%	70%
Data into 2 std. dev.	90%	100%
Data into 3 std. dev.	100%	100%

This experience led us to consider an alternative approach, since it seemed unlikely that the other option, of instigating ever more controls on the processes, the teams and the experiment as a whole, would produce data that would be any better for the traditional analysis than before. We were also concerned that there could be a problem of the experimental protocols interfering with the experiments, and indeed

the issue of balancing the extent to which an experiment of this sort can be fully controlled against the possible adverse effects of such controls underlies our argument that alternative approaches to ESE may be required. This issue also suggests that the conventional binary division between experiments and quasi-experiments needs to be generalized to a continuum of degrees of control, which is why the simple term “experiment” is used throughout this paper for the activities being discussed.

4. Problems with the Experiments

In seeking explanations for the difficulty in obtaining statistically significant results from these experiments, we have identified three different effects, all relating to the way in which individual people work when carrying out the processes that are being studied. These three effects are concerned firstly with the differing capabilities of individuals, secondly with the extent to which they have actually followed the processes that were supposed to be being compared, and thirdly with the effects of individual personalities on the effectiveness with which the teams have worked.

The first of these effects, of variations in the capabilities of individual developers, was identified over 30 years ago by Sackman, Erikson and Grant [13], who found that the productivity of the programmers whom they were studying varied by a factor of at least 10 to 1. Curtis [14] not only confirmed this finding, but found greater variations, of up to 20 to 1, which was also confirmed by DeMarco & Lister [15]. For the Observatory experiments, where the subjects are students, trying to measure their ability is an integral part of the educational process, and so these measurements have been incorporated into the experimental models. Also these students form a quite highly selected group, which should exhibit a smaller range of abilities than in a more random set of experimental subjects. The scale on which these measurements are made is in principle a ratio scale, but in practice the underlying empirical relation system is based on assembling small assessment tasks in a much simpler fashion than in the work required to develop software systems, and so we can not guarantee that there is a linear relationship between measurements on this scale and the productivity of the individuals being measured.

Hence, while the range of our measurements is at most about 3 to 1, this may well underestimate the range of actual productivities in software development. More to the point, this range far outweighs the effects due to the different development processes that we are trying to compare, since these are much less than 2 to 1. This, of course, is not just a problem for our experiments. The data above suggests that the subjects of typical experiments in ESE could well exhibit a range of ability of at least 10 to 1, which is significantly greater than that which we have measured. This would almost certainly be much larger than the variations in productivity that might result from the effects of adopting different processes, and so would swamp completely any effects that might be due to the actual processes being studied, as is happening in our case.

The second effect relates to a key assumption underlying any experiment involving the use of different processes to carry out particular tasks, which is that experimental subjects will in fact follow the particular processes that have been prescribed for them. For simple tasks this is a reasonable assumption, and it is also feasible within

the experiments to monitor closely the extent to which the subjects do actually follow the prescribed processes. As the tasks become larger, however, and the processes more complex, then ensuring that these processes are in fact followed correctly by each subject becomes more complex, as individuals become more likely to just do things their way, rather than in the way that is required by the experiment.

This has certainly proved to be the case in these experiments, where what was found in practice was that none of the teams who were supposedly following XP actually did so completely, in the sense of using all of the practices that characterize XP. Furthermore, some of the teams that were supposedly following a traditional methodology actually did so while putting such an emphasis on testing throughout the development that they were arguably using more of the XP practices than some of the teams who were supposedly using XP.

These departures from the intended processes may well have been extreme cases, and the monitoring of the actual work done by each team meant that they were detected, and so allowance made for them in drawing conclusions from the results of the experiments. On the other hand, monitoring teams' work in the kind of detail that is necessary to detect this sort of situation is extremely time-consuming, and is liable to become very intrusive. This raises the concern that this intrusion may start to distort the actual behavior that one is trying to observe, in what could be described as an ESE analog to the effects of Heisenberg's Uncertainty Principle.

The third effect derives from the observation that some combinations of individual personalities work better in teams than others [16], although as yet there seems to be little qualitative data, and virtually no quantitative data, that can characterize these effects on the activity of developing software systems within teams. As part of the processes used for running these experiments, extensive interviews have been held with some of the experimental subjects. This has provided considerable evidence, even if only anecdotal, to suggest that the interactions between the personalities of the different individuals has had a substantial effect on their performance, and consequently on the quality of the products that they have produced during the course of the experiments.

We are only in the early stages of studying these effects, and analysis of the first results is still in progress, but even from this it is apparent that, when teams experience personality clashes between the members, the result can be a substantial drop in the effectiveness with which they work. Indeed, it appears that, as with the effects of individual ability, these effects could be sufficient to swamp the kinds of effects due to the processes being compared that we would hope to see emerging from our experiments.

5. Conventional Solutions

These problems that we have encountered do not just affect our experiments: they also represent problems for the whole approach to ESE, or at least for the traditional approaches to it that we have been trying to scale up to our studies. To see this, we examine how these approaches would require each of these various problems to be tackled.

The traditional approach to solving the problem of the wide variation in developer ability would be through careful design of the experiments, and in principle there are two kinds of design that could be adopted. One of these designs would involve trying to measure the ability of the individual developers, so as to incorporate this explicitly into the model for the experiments, while the other design would try to structure the experiments so that each experimental subject is measured performing each of the various processes, with the aim of implicitly canceling out the differences in ability. Unfortunately, neither of these designs can reliably solve the problem.

Trying to measure developer ability runs into the problem that this is essentially an internal attribute of the individual developer, and moreover is a structured attribute rather than a single-valued one, whereas the quantities that can most easily be measured are single external attributes such as the productivity when performing a particular task. The relationships between these internal and external attributes are not at all clear, so that if the ability of the subjects is to be characterized appropriately for a particular experiment by using measurements of such external attributes, then it needs to be measured while carrying out tasks that are closely related to those to be carried out during the experiment itself. This immediately creates the danger that this measure of ability becomes both an independent and a dependent variable within the experimental model.

Furthermore, one of the internal attributes associated with ability is the speed at which new methods or techniques can be learnt, and this renders equally unreliable any experimental design where a subject undertakes some form of task more than once. This still applies even if the repetition involves using different techniques for the task, as the learning effect is likely to mean that the later attempts at the task will be carried out better than the earlier ones. Furthermore, more able subjects will show a greater improvement than the less able ones, so that simply requiring them to repeat the tasks with different processes will not necessarily achieve the compensation for different levels of ability that might be hoped for.

The problem that individuals might decide not to follow exactly the processes that they are meant to be using in their roles as experimental subjects is one that could lead to false results being produced by the experiment, which need to be designed to avoid this. On the other hand, every step in developing a piece of software involves individual judgment, and it is almost impossible to constrain this in such a way that individuals will be encouraged to make judgments about some aspects of the process while being prevented from making others. The traditional solution to this would be to monitor more carefully the actual processes being used, but this is not simple to do, particularly given the need to avoid excessive intrusion. Automation of the monitoring may help to solve this problem, but it will mainly provide low-level data on what the developers are doing, whereas the key to understanding how the processes are being applied is to obtain high-level data on why they are doing it.

The other possible solution is to design the experimental models so as to allow for such variations in the actual process, rather than just relying on simple comparisons of one process against another. For instance, in our experiments sufficient data had been collected that it was possible to identify how many of the XP practices each team had actually used (whether they were supposed to be following XP or not). Hence, some conclusions could be drawn from analyzing the results by taking this number as a dependent variable that characterized a range of treatments, rather than simply

comparing results for XP and traditional methods taken as the two treatments. The difficulty with this approach is that it requires the process models to be expressed in terms where such differences can be characterized quantitatively, rather than simply qualitatively. As hinted earlier, the development of such models is still in its early stages, and represents a significant strand of future work for the Observatory.

The third problem is that of the nature and magnitude of the effects of personality interactions within teams, but given the present state of knowledge (or lack of it) about this, it appears to us that these effects may well present one of the biggest challenges for ESE, and particularly for conventional approaches to it. As yet we can offer little in the way of guidelines for how to monitor teams to try to detect occurrences of such personality clashes, and even less in the way of hypotheses as to how such clashes might be predicted, or the magnitudes of their effects measured. Nevertheless, if ESE is to progress to measuring the actual behavior of real teams performing anything other than the simplest tasks, then the problems that are posed by the likelihood of such personality clashes have to be solved.

6. Solving the Problems

Given these limitations of traditional approaches to ESE, we have been investigating alternative solutions to these problems of noisy data and limited experimental populations. For the first of these problems we have been exploring the use of Bayesian statistics, and the second has led us to propose the adoption of more agile approaches to ESE.

6.1 Bayesian Statistics

Bayesian statistics represents a different point of view for analysing data [17], and provide an alternative to the classical (frequentist) approach [18]. It allows forecast (calculating probabilities) of future observations and incorporates evidence from previous experiments into overall conclusions. One possible problem is that this approach is subjective: different people may reach different conclusions from the same experimental results.

In our analysis of repeated experiments we have historical data from previous runs of the experiment, and so the information is available for a Bayesian approach. This approach involves [19] starting from a set of prior probabilities, and then combining this with the new information by applying Bayes Theorem in an inference process that will generate a set of posterior probabilities, from which a statistical conclusion can be drawn. This approach is particularly recommended for situations that involve uncertainty, which is certainly the case in these experiments, where the effects described above mean that much of the data is extremely noisy.

For the purpose of this analysis, we identify teams as either successful and failing, by defining a team to be “successful” if its quality measure is above the global average, or “failing” if its quality measure is below the global average. For the 2002 run of the experiment this global average of Quality (Lecturer + Client) was 70.025, and so a team succeeds if $Q > 70.025$, and fails if $Q \leq 70.025$. This gave the

distribution of teams shown in table 2, and the Bayesian analysis of this data can be summarized by the tree diagram in figure 1.

Table 2. The distribution of teams

SHP-02	Success	Fail
XP	6	4
Traditional	5	5

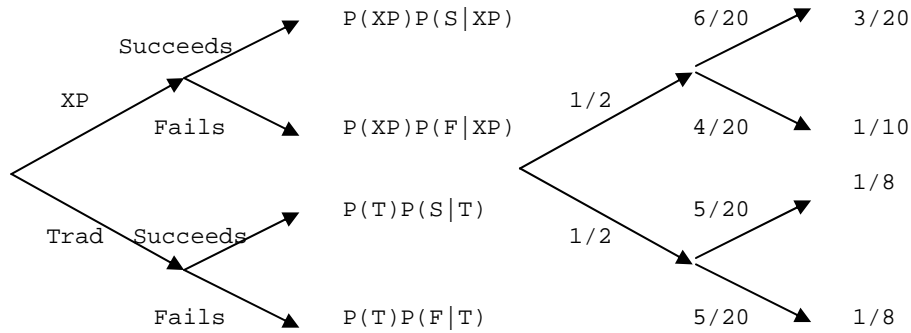


Fig. 1. The tree diagram for the Bayesian analysis

Hence, the result is that if we choose randomly a successful (or failing) team:

- the probability that a successful team was XP is: 0.5454;
- the probability that a successful team was Traditional is: 0.4545;
- the probability that a failing team was XP is: 0.444; and
- the probability that a failing team was Traditional is: 0.555.

Bayesian statistics thus help to take account of small differences that appeared continuously among our data, as though we could look at these differences through a magnifying glass. The Bayesian approach helps to predict the behavior of the data in the next replication and to correct the noisy factors, and so gives us a means to relate results from one experiment to the previous and even the next one.

6.2 Agile Approaches

Increasing the population size for an experiment usually involves repeating it several times with different groups of subjects, which can be a time-consuming process. By analogy with agile development methods, where a large process is broken down into small steps, with feedback as part of each step, we propose that ESE could also adopt an agile approach to such repetitions of experiments. This would involve concentrating on a limited set of data that can be measured easily and reliably, and carrying out Bayesian analysis after each repetition of the experiment.

Such an approach may then provide a more adaptable and cost-effective way to carry out empirical research into the highly complex and multi-parameter situations

that we are interested in. In particular, it should focus on obtaining results that are useful at the time, in the same way that agile development processes do, and so obviate the risk of an experiment producing no useful results until long after the research question that led to it was posed, as could happen with the Observatory experiments.

In developing this approach the key step that we need to achieve is to adapt the data collection to provide a more suitable framework for the overall analysis, rather like the *test first* approach of XP and the *continuous delivery* of results. In doing this we aim to reduce the bureaucracy involved in the data collection and monitoring as far as possible, so as to provide a basis for more useful and rapid results, and with less risk that these will have been influenced by intrusive experimental protocols.

When considering next year's experiment we will monitor a number of different characteristics that are essentially individual traits, to see what impact they may have on the projects. We can make judgments about, for example, internal quality, by measuring the incremental deliverables – documentation, plans, timesheets – on a weekly basis and use Bayesian analysis in order to decide what actions should be taken to monitor possibly new types of data. For example, if we find that some teams are progressing less well than others we will try to identify the traits that might be contributing to this in more detail, thus adapting the experiment as it proceeds.

7. Summary and Conclusions

Full life-cycle experiments are sufficiently complex that controlling all parameters is unlikely to be practical, and conventional approaches to analysis may be incapable of delivering useful results within an acceptable timescale. In particular, the impact of personal traits on the project process and outcome, whether derived from design team members or clients, can affect significantly the final quality of the systems developed and hence the success of the methodology used. Software engineering techniques, technologies and applications are changing so rapidly that very thorough, lengthy empirical research may produce results that are less relevant than they might be because the world has moved on.

Consequently, although we recognize that good experimental design is vital, as is careful modeling of the possible sources of noise in the data, we believe that some of the recommendations of some empirical software engineering researchers are neither necessary nor desirable. As well as making experiments prohibitively expensive, they also have the potential to interfere with the processes that the experiments are trying to observe, and hence destroy the very effects that they are trying to achieve.

As an alternative way of conducting such experiments, we propose a lightweight and adaptive approach, adopting Bayesian analysis methods in a series of light touch experiments, which adapt as the projects evolve and focus on interesting issues as they arise. We are already finding useful information about many important issues without the need for highly controlled and expensive experimental designs and protocols. In the long term we aim to try to validate this approach by continuing our experiments, to see whether adding modest additional controls in an agile fashion, and further increasing the population size, can eventually combine to lead to statistically

significant results. This will take much longer than lightweight approach, but if such statistical significance can be achieved it will provide empirical confirmation of the validity of using the lightweight approach to obtain results on a timescale that matches the requirements of the software development industry.

8. References

1. Kitchenham, B.A., Pfleeger, S. L., *et al.*: Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering* **28** (2002), 721-734
2. Beck, K.: *Extreme Programming Explained: Embrace Change*. Addison Wesley Longman (2000)
3. Cowling, A. J.: The First Decade of an Undergraduate Degree Programme in Software Engineering. *Annals of Software Engineering* **6** (1998) 61–90
4. Stratton, A., Holcombe, M., Croll, P.: Improving the Quality of Software Engineering Courses Through University Based Industrial Projects. In: Holcombe, M., Stratton, A., Fincher, S., Griffiths, G. (eds): *Projects in the Computing Curriculum*, Springer Verlag, Berlin and London (1998) 47–69
5. Holcombe, M., Ipaté, F.: *Correct Systems: Building a Business Process Solution*. Springer Verlag (Series on Applied Computing), Berlin & London (1988)
6. Ipaté, F., Holcombe, M.: Generating test sets from non-deterministic stream X-machines. *Formal Aspects of Computing* **12.6** (2000) 443–458
7. Holcombe, M., Gheorghe, M., Macias, F.: Teaching XP for real: some initial observations and plans. *Proc. XP2001* (2001) 14–17
8. Macias, F., Holcombe, M., Gheorghe, M.: Empirical experiments with XP. *Proc. XP2002* (2002) 225–228
9. Macias, F., Holcombe, M., Gheorghe, M.: Design-led and design-less: one experiment and two approaches. *Proc. XP2003*, Lecture Notes in Computer Science, Vol. 2675, Springer Verlag (2003) 394–401
10. Syed-Abdullah, S., Holcombe, M., Gheorghe, M.: Practice makes perfect. *Proc. XP2003*, Lecture Notes in Computer Science, Vol. 2675, Springer Verlag (2003) 354–356
11. Robillard, P. N.: Measuring Team Activities in a Process-Oriented Software Engineering Course. *Proc. 11th Conference on Software Engineering Education & Training*, IEEE Computer Society Press, Los Alamitos (1998) 90-101
12. Sobel, A. E. K., Clarkson, M. R.: Formal Methods Application: An Empirical Tale of Software Development. *IEEE Transactions on Software Engineering* **28** (2002), 308-320
13. Sackman, H. H., Erikson, W. J., Grant, E. E.: Exploratory experimental studies comparing online and offline programming performance. *Communications of the ACM* **11.1** (1968) 3–11
14. Curtis, B.: Substantiating Programmer Variability. *Proceedings of the IEEE* **69.7** (1981) 846
15. DeMarco, T., Lister, T.: Programmer performance and the Effects of the Workplace. *Proc. Eighth International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos (1985) 268–272
16. Bass, B. M., Dunteman, G.: Behaviour in Groups as a Function of Self-, Interaction, and Task Orientation. *J. of Abnormal and Social Psychology*, **66.5** (1963) 419 – 428
17. Berry, D. A.: *Statistics. A Bayesian Perspective*, Duxbury Press, USA (1996)
18. Aczel, A. D.: *Complete Business statistics. 3rd edn*. Irwin, USA (1996)
19. Anderson, D. R., Sweeney, D. J., Williams, T. A.: *Statistics for Business and Economics. 7th edn*. South-Western College Publishing, USA (1999)