Empirical Software Engineering, 2, 291–312 (1997)

An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents

LIONEL C. BRIAND
Fraunhofer Institute (IESE), Kaiserslautern, Germany

CHRISTIAN BUNSE
Fraunhofer Institute (IESE), Kaiserslautern, Germany

JOHN W. DALY
Fraunhofer Institute (IESE), Kaiserslautern, Germany

CHRISTIANE DIFFERDING
Dept. Computer Science, University of Kaiserslautern, Kaiserslautern, Germany

Abstract. Several important questions still need to be answered regarding the maintainability of object-oriented design documents. This paper focuses on the following issues: are object-oriented design documents easier to understand and modify than structured design documents? Do they need to comply with quality guidelines such as the ones provided by Coad and Yourdon? What is the impact of such quality standards on the understandability and modifiability of design documents? Answers can be based on informed opinion or empirical evidence. Since software technology investments are substantial and contradictory opinions exist regarding design strategies, performing empirical studies on these topics is a relevant research activity.

This paper presents a controlled experiment performed with computer science students as subjects. Results strongly suggest that quality guidelines based on Coad and Yourdon principles have a beneficial effect on the maintainability of object-oriented design documents. However, there is no strong evidence regarding the alleged higher maintainability of object-oriented design documents over structured design documents. Furthermore, results suggest that object-oriented design documents are more sensitive to poor design practices, in part because their cognitive complexity becomes increasingly unmanageable. However, because our ability to generalise these results is limited, they should be considered as preliminary, i.e., it is very likely that they can only be generalised to programmers with little object-oriented training and programming experience. Such programmers can, however, be commonly found on maintenance projects. As well as additional research, external replications of this study are required to confirm the results and achieve confidence in these findings.

Keywords: experiment, maintainability, object-oriented design, structured design.

1. Introduction

Object-oriented techniques have become increasingly popular as a methodology for developing new software systems. Unfortunately, this occurred mainly as a result of opinion and anecdotal evidence, and not as a result of empirical evidence demonstrating that these techniques offer significant advantages over other different techniques. Jones (1994), for example, identified several areas where a distinct lack of empirical evidence exists to support the assertions of gains in productivity and quality, reduction in defect potential and improvement in defect removal, and reuse of software components.

More empirical research has been performed since Jones' position, but the evidence does not support the claim that object-oriented development techniques always provide the many benefits accredited them, as has been suggested by advocates in the past. For example, positive results have been provided by Basili *et al*. (1996) who found that for their study object-oriented techniques provided significant benefits from reuse in terms of reduced defect density and rework as well as increased productivity. Similarly, the NASA SEL showed that, after having introduced and tailored an object-oriented design method into their development environment and providing substantial training to their developers, benefits could be obtained from reuse in terms of reduced defect density and increased productivity (Basili et al., 1992). Not so favourable empirical evidence was provided by van Hillergersberg *et al*. (1995) who investigated the performance and strategies of programmers new to object-oriented techniques and concluded that object-oriented concepts were not easy to learn and use quickly. Daly *et al*. (1996) provide evidence which suggests that inheritance depth and conceptual entropy of class hierarchies can cause programmers difficulty maintaining object-oriented software.

Clearly more empirical research is needed to investigate when object-oriented techniques provide significant advantages over other techniques and when they do not. One particular area which warrants immediate investigation is maintainability of object-oriented software—time and again, object-oriented development techniques have been promised to increase maintainability. If true, an organisation switching to object-oriented techniques would be likely to save large amounts of money throughout the lifetime of an object-oriented system.

This paper presents an empirical study which investigates two important components of design maintainability, namely its understandability and modifiability, by comparing the effect of (a) different design techniques and (b) design principles perceived to be 'good' and 'bad' practice, on these attributes. We view this study as exploratory—we intend to identify and refine important hypotheses and investigate them further. The paper is partitioned as follows. Section 2 presents the experimental details of the study. Section 3 summarizes the results of the data analysis and presents important details which help to explain them. Section 4 discusses the various threats to the validity of the study. The collected data shows that subjects with little training do not greatly benefit, in terms of understanding and modifying design documents, from object-oriented development techniques. In contrast, the data shows with statistical significance that adherence to good object-oriented design principles is required if promised object-oriented benefits are to be realised—if not, there is further statistical evidence to suggest that object-oriented design documents are more difficult to understand and modify than appropriate structured design documents.

2. Description of the Experiment

The study was conducted to investigate two separate effects and their interaction. First, do object-oriented techniques increase the understandability and ease of modification of the resulting design documents over the use of structured techniques? Second, does the use of perceived ‘good’ and ‘bad’ design principles have any influence on the understandability and ease of modification of these design documents? Coad and Yourdon identify a set of design principles which they advocate, if adhered to, will result in a better object-oriented design (Coad and Yourdon, 1991a; 1991b). The applicable design principles they identify include guidelines on

Coupling. First, interaction coupling between classes should be kept low, something which can be achieved by reducing the complexity of message connection and simplifying the number of messages that can be sent and received by an individual object. Second, inheritance coupling between classes should be high, achievable by ensuring that each specialisation class is indeed a specialisation of its generalisation class.

Cohesion. First, a service should carry out one, and only one, function. Second, the attributes and services should be highly cohesive, i.e., all attributes and services should be referenced and used and they should all be descriptive of the responsibility of the class. Third, a specialisation should actually portray a sensible specialisation—it should not be some arbitrary choice which is out of place within the hierarchy creating a less cohesive class.

Clarity of design. First, use of a consistent vocabulary is important—the names in the model should closely correspond to the names of the concepts being modeled. Second, the responsibilities of a class should be clearly defined and adhered to. Furthermore, the responsibilities of any class should be limited.

Generalisation-Specialisation depth. It is important not to create specialisation classes just for the sake of it. Rather an inheritance hierarchy should attempt to model part of the problem. The rule is to specialise only if X (the specialisation class) *is-a* Y (the generalisation class).

Keeping objects and classes simple. First, avoid excessive numbers of attributes in a class—an average of one or two attributes for each service in a class is usually all that is required. Second, “fuzzy” class definitions should be avoided. All definitions should be clear, concise, and comprehensive.

Of course, these design principles are not operationally defined and their application requires a certain degree of subjective interpretation.

2.1. Hypotheses

To be able to test the hypotheses below, four different design documents were required, two object-oriented and two structured. The two object-oriented design documents were prepared according to the above design principles, the ‘good’ system being designed to adhere as best as possible to the design principles and the ‘bad’ system being changed to prevent the principles being adhered to—the OMT methodology of Rumbaugh *et al*. (1991) was used to represent the designs. As a result, the ‘bad’ design, which did not obey the design principles of Coad and Yourdon, had additional coupling between classes, specialisation levels which were not fully appropriate, less cohesive classes, e.g., by concatenating two classes into a single one, classes with unneeded, although sensible, methods and attributes, and classes which had an inconsistent vocabulary and inconsistent use of method names and messages. Similarly, the two structured design documents were designed in the same manner, where the relevant Coad and Yourdon object-oriented design principles were adapted to apply to structured designs—MIL/MDL based on DeRemer and Kron (1976) was used to represent the designs. Differences between the 'good' and 'bad' structured design documents were similar to the differences between the object-oriented design documents, the exception being additional specialisation—here the procedure calling hierarchy was made deeper by one or two levels. (See section 2.3 for details of the different application domains used).

Of interest are the concepts of understandability and modifiability (see Section 2.6). As with many other concepts in software engineering, they are difficult to measure fully—in this study, understanding is captured via means of asking questions about the components of the system designs. Modifiability is captured by means of subjects performing impact analyses on the design documents. Impact analysis, which is only one important dimension of modifiability, is defined as the activity of identifying what to modify to accomplish a change; see for example (Arnold, 1993). We discuss our choice of measuring the success of impact analyses (but not making the actual changes required) to capture modifiability in Section 4.

Standard significance testing was used to clearly specify the two effects identified—for the sake of brevity, we have supplied only one null hypothesis and have included both understandability and modifiability in each alternative hypothesis instead of creating a single one for each. The null hypothesis is stated as

H_0 —There is no difference between design documents, in terms of ease of understandability and modifiability, developed by the use of object-oriented or structured techniques regardless of any 'good' or 'bad' design principles applied.

The alternative hypotheses, i.e., what was expected to occur, were then stated as

H_1 —'Good' object-oriented design is easier to understand and modify than 'good' structured design.

H_2 —'Good' object-oriented design is easier to understand and modify than 'bad' object-oriented design.

H_3 —'Good' structured design is easier to understand and modify than 'bad' object-oriented design.

H_4 —'Good' structured design is easier to understand and modify than 'bad' structured design.

H_5 —'Bad' structured design is easier to understand and modify than 'bad' object-oriented design.

We now explain our reasoning behind these alternative hypotheses. H_1 is stated because it is a commonly held belief by many in the software engineering community. H_2, H_3, and H_4 are stated on the basis that when sensible design principles are applied they will aid the understandability and modifiability of the resulting design documents. H_5 is stated in this direction because of research conducted by Daly *et al*. (1995). In their survey it was discovered that many software practitioners were of the opinion that if object-oriented software was badly designed it would be more difficult to maintain than a poorly structured designed equivalent. The reasoning is that object-oriented concepts when abused, cause many more difficulties to maintainers than structured concepts do.

2.2. Subjects

The participants of the study were computer science students at the University of Kaiserslautern, Germany, who were enrolled in the basic software engineering class lasting a semester. During the lectures the students were taught the basic software engineering principles as well as being introduced to object-oriented and structured development techniques. The lectures were supplemented by practical sessions where the students had the opportunity to make use of what they had learned through completion of various software development exercises.

During the course, subjects were asked if they would be interested in participating in further practical exercises. The subjects knew data would be collected during these exercises and that analysis would be performed on this data, but were unaware of the experimental hypotheses that were being tested. Twenty students expressed their interest in participation. These subjects were then given extra practical sessions where they received intensive training on how to read design documents and how to perform impact analysis on the documents, prior to participation in the experiment (see section 2.4 for full details of the training).

As the German system allows students to take different classes at different times during their studies, the students were of varying degrees of experience, although the majority of the students who volunteered had their Vordiplom.[1] In general, the subjects had little knowledge of structured development techniques and very little or no knowledge of object-oriented development techniques.[2]

2.3. Experimental Materials

To test the hypotheses stated in section 2.1 four separate software designs were required: a ‘good’ and ‘bad’ object-oriented design and a ‘good’ and ‘bad’ structured design. The documentation accompanying each design was approximately thirty pages and included the system description, the customer requirements, the developer requirements documents, and the design documents. The documentation used for the study was intended to be as similar as possible in terms of the information content they contained, i.e., a serious attempt was made to keep the differences between the four design documents to those caused by (a) the design techniques used, (b) the design principles applied, and (c) and different application domains. The application domains used for the four designs were (i) a temperature controlling system (‘good’ object-oriented), (ii) an automatic teller machine (‘bad’ object-oriented), (iii) a software measurement tool (‘good’ structured), and (iv) a scheduling software system ('bad' structured). Of course, some domains are better suited to an object-oriented solution than to a structured one and vice versa. We discuss the impact of this as a threat to internal validity in Section 4.2.

For each of these designs there were two sets of tasks to be performed. First, subjects had to read the documents and then had to complete a questionnaire which asked various questions about (i) their overall understanding of the design, (ii) the structure of the design, and (iii) more specific questions which were answerable from the design documentation provided. The second task required two separate impact analyses to be performed. First, impact analysis had to be performed on both the system description document and the design documents as a result of a change in customer requirements. Second, impact analysis had to be performed on both the system description document and the design documents, this time as a result of an enhancement of system functionality.

The tasks were created for each design independently but in such a way that comparison between subject performances could be made. However, it is almost an impossible task to design potential modifications which required exactly the same number of places to be changed in a design. As a result the number of places to be identified in each design ranged between 22 and 33 places. Of course, the difference may also be due in part to the nature of object-oriented systems which tend to have their functionality distributed more widely (Wilde and Huitt, 1992; Kung et al., 1994). To be sure the tasks were comparable, an expert in both object-oriented and structured techniques was timed while performing the tasks. It was found that for each task the time required was approximately the same. Similarly, to answer all questions in each questionnaire a comparable amount of time was required.

After completion of the tasks subjects were given a debriefing questionnaire. This questionnaire captured opinions with respect to (i) their performance, e.g., how much of the understanding questionnaire did they estimate they had answered correctly, how accurate and complete did they think their impact analyses were, (ii) their motivation for participation, and (iii) the experiment itself, e.g., realism of tasks, was there enough time given? (N.B. The debriefing questionnaire used in this study had several weaknesses. We point the reader to second study (Briand, Bunse and Daly, 1997) where substantial effort was spent defining a more thorough debriefing questionnaire—this questionnaire is of more use to those considering performing a replication).

2.4. Experimental Procedure

Before the experimental study took place, in addition to the software engineering course, subjects received some intensive training. The training began with additional teaching where the students were taught how to efficiently perform impact analyses. A practical session then followed which was essentially a dry run of the experiment proper—students had to answer information questionnaires and perform impact analyses on design documents similar to the real experimental tasks. The practical session was conducted interactively so subjects were able to ask about anything they did not understand.

The experiment was then performed over two separate days with each subject receiving different design documents each day (see section 2.5 for details of subject allocation). Each experimental run took place in a class room where the subjects had plenty of space to examine all the design documents. Each subject sat next to a subject who was examining different design documents—this was performed to reduce plagiarism, although this was by no means a significant worry. Subjects were told verbally that there were different designs being worked upon, but were not told anything about the nature of the study, e.g., what hypotheses were being tested, what type of design document they were working with. The subjects were then given a maximum of two hours to complete all the tasks. During this time subjects were told not to talk between themselves, but to direct any questions they had to the three monitors. Questions directed towards the monitors were not answered if thought to assist subjects' performance. After completing their tasks, each subject completed a debriefing questionnaire before leaving.

2.5. Experimental Design

A 2×2 factorial design in two blocks of size two was employed (Moen, Nolan and Provost, 1991). The two independent variables being the design technique used (object-oriented or structured) and the design principles applied ('good' or 'bad'). This design assumes that these variables are fully independent. If this were not the case the design would be nested, not factorial, and comparison of subjects' performance could not be made across factors, only within them. In particular, one argument might be that the design principles applied are not the same across the design techniques, i.e., design principles are confounded with design technique, and comparison can be only made between good and bad object-oriented and good and bad structured. This argument does not consider the following facts. First, although design principles are implemented in a manner which *is* determined by the design technique used, the design principles of Coad and Yourdon apply to the same internal attributes of a design regardless of the design technique used, e.g., coupling, cohesion, decomposition structure. Second, although the design principles can be violated in very different ways according to which design technique is used, this is a reflection of reality and should, therefore, not threaten the validity of our design. For example, considering the decomposition structure of the software, in structured designs lower level functions are encapsulated in higher level functions whereas in object-oriented designs classes are specialized into more specific classes via inheritance. As a result, a decomposition error in the former case can result in inappropriate calls in the higher levels of the call graph whereas, in the latter case, it can result in inappropriate specializations/generalizations of classes. We believe this does not cause a problem for our experimental study because it represents, in a realistic manner, violations of what is perceived as good design practice for a given design technique. If these violations turn out to be more costly for a particular design technique then that is something we should be interested in. Third, if we use a nested design then it makes it impossible to compare maintainability of object-oriented and structured designs while controlling for their differences in terms of adherence to Coad and Yourdon principles. It then becomes difficult to provide precise answers to our questions. Consequently, we feel justified in using a factorial design rather than a nested one.

For educational purposes there was a requirement that each subject had to have exposure to an object-oriented design and a structured design as well as exposure to a design which had adhered to 'good' and 'bad' design principles.[3] This meant that repeated measures

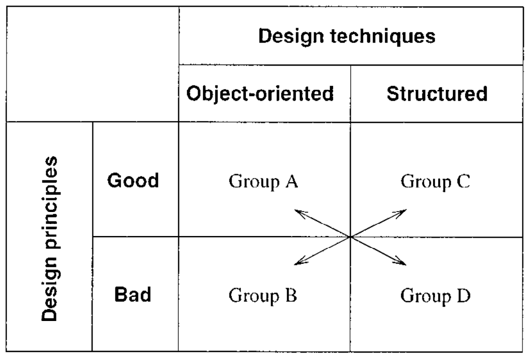

Figure 1. Group allocation to the different types of design documents.

analysis could only be performed for one of our hypotheses (H_3). Figure 1 illustrates this constraint through the allocation of groups to the different designs, where Groups A and D form one block and Groups B and C form the second block. For example, Group A performed the experimental tasks for the 'good' object-oriented design first and then the experimental tasks for the 'bad' structured design. Group D did the opposite of Group A. Note that this procedure is known as counter-balancing, one method which should eliminate any ordering effects caused by the tasks as well as any learning and fatigue effects.

Subjects were then randomly assigned to one of these four groups. This was achieved by asking each subject to draw a number from a hat. Before the numbers were drawn the numbers had been allocated to groups sequentially, numbers 1 to 5 for group A, numbers 6 to 10 group B, and so on. Once a subject drew their number, allocation to a group became clear.

2.6. Data Collection Procedures and Dependent Variables

As stated previously, subjects' understanding of the designs was measured based on their accuracy of completing the task questionnaire. Data for each impact analysis was collected in two ways: (i) subjects had to mark on the system description and design documents exactly where they thought modifications would have to be made and (ii) subjects then had to complete a data collection form to summarise the places identified. This allowed the accuracy of the form to be cross checked by the researchers. The time to complete the tasks was also recorded. From this data three sensible dependent variables are derived. Que_%, which represents the percentage of questions that were answered correctly—as the questionnaire was used to gauge the subjects' understanding of the design, it is reasonable to use the percentage of correct answers as a measure of this understanding. Mod_%, which represents the percentage of places to be changed during the impact analysis that were correctly found—it is reasonable to measure the effectiveness of a modification by the relative amount of places to be changed found. Mod_Rate, which represents the modification rate, calculated by dividing the number of correct places found by the total time taken—it is reasonable to measure the efficiency of a modification by the number correct places found per time unit. We are confident that our dependent variables are valid measures of the understandability and modifiability of the system documents. It is important to note that modifiability is expressed only in terms of impact analysis—where changes were required was identified, but the changes themselves were not implemented.

2.7. Data Analysis Procedure

Data was collected for thirteen subjects over the two experimental runs. Therefore, twenty six data points were available for analysis—six data points for the 'good' object-oriented design, seven data points for 'bad' object-oriented, seven data points for 'good' structured,[4] and six data points for 'bad' structured (see section 4.2 for details of subject loss as a threat to internal validity).

As discussed in section 2.5, repeated measures analysis could not be applied because of the constraint placed upon the design; therefore, the appropriate test to use was a single factor, one way ANOVA test (Devore, 1991). The exception to this though was for H_3—the data collected for this hypothesis *is* within-subjects and consequently a repeated measures test is applicable; we use the paired t-test in this instance. (Note that for each parametric test applied and reported in Section 3, an alternative non-parametric test was also applied and obtained similar results). To proceed with the analysis, we have to preset a level of significance, i.e., the α level, at which we will be working for this study. Several factors have to be considered when setting α. First, the implications of committing a Type I error, i.e., incorrectly rejecting the true null hypothesis, have to be determined. In our application context, that would mean the cost of using a new design technique without achieving any beneficial effect, using a less than optimal design technique, or applying useless design principles. Second, the goals of the study have to be taken into account. This can be discussed from two perspectives:

A scientific perspective: identify cause-effect relationships between design techniques, quality standards, and maintainability, with a high level of confidence.

A practical perspective: which design technique is more likely to perform better with respect to maintainability? Are we more likely to significantly benefit from introducing standards regarding structural properties of design than by not introducing them?

We regard this empirical study as exploratory research whose goal is twofold: first, we want to identify potentially interesting and practically significant trends to focus future studies. Second, we wish to gain initial insights into what might be the consequences of using object-oriented design, 'good' design principles, and their interaction. Therefore, we should not adopt a too stringent α level—this might result in overlooking potential areas of further investigation. In addition, from a practical perspective, we are in a situation where a decision has to be made regarding the selection of a design technique or 'good' design principles. In that context, we are more interested in what is the most likely optimal decision than in absolute scientific statements. An α as high as 0.2, or more, might be considered good enough to make a decision, even though the empirical evidence is not strong enough to make a scientific statement with a high degree of confidence. In our study, we use $\alpha = 0.1$, which can be seen as an acceptable compromise between the different perspectives above and considering the exploratory nature of our work. In addition, we will provide p-values up to 0.2 (i.e., exact probabilities of committing an error of Type I) resulting from ANOVA—this allows the reader to make their own decisions regarding the trends observed.

Another factor affecting the analysis procedure is that while there is a large enough number of data points to apply the statistical tests, the small sample sizes are likely to have an adverse effect on the power of these methods, i.e., the chance that if an effect exists it will be found; for details see (Kraemer and Thiemann, 1987; Miller et al., 1997). For example, a power value of 0.4 means that if an experiment is run ten times, an existing effect will be discovered only four times out of the ten experimental runs. Power of a statistical test is dependent on three different components: α, the size of the effect being investigated, and the number of subjects.[5] Given the effect size and number of subjects are constant, increasing α is the only option for increasing the power of the test applied (Miller et al., 1997).[6] This provides further justification for our decision to set α to 0.1 instead of the 0.05 level which is more commonly used in software engineering. Low power will have to be considered when interpreting non significant results. It is for this reason that practical (or clinical) significance also needs to be considered (Slakter, Wu and Suzuki-Slakter, 1991; Rosnow and Rosenthal, 1989). Practical significance is concerned with whether the effect being investigated impacts upon the dependent variable(s) in a manner that can be considered practically meaningful, i.e., the effect is large enough to be of interest. To determine if this is the case we will calculate the observed effect size (γ) detected for each dependent variable for each hypothesis. This measure is expressed as the difference between the means of the two samples divided by the root mean square of the variances of the two samples (Miller et al., 1997). We intend to discuss all practically significant results and not constrain ourselves to discussing only statistically significant results. For this exploratory study we consider effects where $\gamma \geq 0.6$ to be of practical significance (the unit is one standard deviation). We make this decision on the basis of effect size indices proposed by Cohen (1969).

3. Experimental Results

Tables I and II presents a descriptive summary of the data collected for each of the four software designs. The columns represent the mean ($\bar{x}$), median ($\tilde{m}$), and standard deviation (s) for each different software system. The rows provide this data for each of the dependent variables, Que_%, Mod_%, and Mod_Rate. The sections below detail the results of the analysis for each stated hypothesis. Before discussing these results, we examine anomalies discovered in the data set.

Table 1. Summary descriptive statistics for the object-oriented systems.

	Object-Oriented					
	$\bar{x}_{Good}$	$\tilde{m}_{Good}$	s_{Good}	$\bar{x}_{Bad}$	$\tilde{m}_{Bad}$	s_{Bad}
Que_%	98.1	100	4.5	82.9	80.0	13.8
Mod_%	69.8	68.8	23.9	53.9	54.5	28.4
Mod_Rate	0.70	0.78	0.22	0.36	0.30	0.24

Table 2. Summary descriptive statistics for the structured systems.

	Structured					
	$\bar{x}_{Good}$	$\tilde{m}_{Good}$	s_{Good}	$\bar{x}_{Bad}$	$\tilde{m}_{Bad}$	s_{Bad}
Que_%	85.7	100	24.7	96.7	100	8.2
Mod_%	67.4	76.1	34.0	52.2	52.2	26.4
Mod_Rate	0.49	0.54	0.28	0.41	0.39	0.21

3.1. Anomalies in the Data Set

Thorough examination of Tables I and II shows two anomalies in the data set—in software engineering experiments, because of the varying degrees in subjects' ability (Brooks, 1980; Curtis, 1980), it is to be expected that anomalies in the data set occur; when working with small samples sizes the importance of debriefing questionnaires to help explain such occurrences must be stressed. First, notice that for Que_%, structured $\bar{x}_{Bad}$ is greater than structured $\bar{x}_{Good}$. This occurs as a result of a relatively low average performance by the 'good' structured block, i.e., groups B and C, as well as a relatively high average performance by the 'bad' structured block, i.e., groups A and D. Examination of the raw data found that two subjects from in the 'good' block did not do particularly well thereby reducing the mean score—as can be seen in Table II $\tilde{m}_{Good}$ is actually 100% whereas $\bar{x}_{Good}$ is only 85.7%. In contrast, all subjects in the 'bad' block did particularly well. Subjects' debriefing questionnaires were examined to facilitate an explanation. Little was found to explain the high $\bar{x}_{Bad}$, but some explanatory comments were provided by the two subjects who had a poor performance with the 'good' system helping to explain the low $\bar{x}_{Good}$. The first subject commented that their English was not of a high standard. Subsequently, they took longer to study the document than the other subjects (their time taken was the longest of all structured performances). The subject also mentioned that they had difficulties as a result of not enough time being available. In contrast, the second subject provided the quickest of all structured performances. This subject stated afterwards they had not read the document fully and this is supported by their very quick time. This is the likely cause for their poor score. Therefore, the performance difference between the 'good' and 'bad' structured groups for Que_% is somewhat explainable, although we are unable to provide explanations for the excellent performance of the 'bad' group (see Section 4.2 for further discussion).

Second, notice that for Mod_%, object-oriented $\bar{x}_{Bad}$ is greater than structured $\bar{x}_{Bad}$,

Table 3. Results for 'good' OO versus 'good' structured.

Variable	γ	df	F	Crit. $F_{0.90}$	p-value
Que_%	0.70	12	1.46	3.23	
Mod_%	0.54	11	0.02	3.29	
Mod_Rate	0.84	11	2.10	3.29	$p = 0.18$

again going against the direction predicted in the hypothesis. The distributions for these data sets are very similar with almost equal quartiles, medians, and maximums. No data was uncovered to suggest any alternative interpretations so we conclude that the observed effect is small and of no practical significance.

3.2. H_1—'Good' Object-Oriented Design Versus 'Good' Structured Design

Table III presents a summary of the results of the statistical tests for the three dependent variables with respect to H_1. Column one represents the dependent variable, column two the size of the effect detected, column three the degrees of freedom, column four the F value of the ANOVA test, column five the critical value for $\alpha = 0.10$ which F has to exceed to be significant, and column six provides the p value if it is below 0.20. By examining columns four and five it is obvious that only Mod_Rate is close to being significant—H_1 cannot be accepted. It is worth noting though that the values for each of the three dependent variables support the direction of this hypothesis, although only Que_% and Mod_Rate show an effect size of practical significance (remember practical significance is deemed to have been achieved when $\gamma \geq 0.6$). For replication purposes which involve using the same experimental design, we have calculated the minimum number of subjects necessary to have a reasonable chance of achieving statistical significance, i.e., one where the power of the test is approximately 0.8. For Que_%, even with α set at 0.1, 56 subjects will be required to to provide the test with a power of 0.8. In Section 3.1 the performance of the 'good' structured block was found to be unduly influenced by two outliers. On this basis, we cannot be sure if the number of subjects required may be even larger. For Mod_Rate, again with α at 0.1, 37 subjects are required.

3.3. H_2—'Good' Object-Oriented Design Versus 'Bad' Object-Oriented Design

Table IV presents a summary in the same format as Table III of the results of the statistical tests for the three dependent variables with respect to H_2. Even with the small sample sizes used in this study significant effects have been detected—significant results are achieved for Que_% and Mod_Rate. We regard this as sufficient evidence to accept H_2. The effect on Mod_%, while not significant, was also in the direction supporting the hypothesis and has an effect size deemed to be of practical significance. For replication purposes, we performed the power calculation and found, with $\alpha = 0.10$, at least 70 subjects are required for a power of 0.8.

Table 4. Results for ‘good’ versus ‘bad’ OO.

Variable	γ	df	F	Crit. $F_{0.90}$	p-value
Que_%	1.48	12	6.67	3.23	$p = 0.03$
Mod_%	0.61	12	1.16	3.23	
Mod_Rate	1.48	12	7.34	3.23	$p = 0.02$

Table 5. Results for ‘good’ structured versus ‘bad’ OO.

Variable	γ	df	t-ratio	Crit. $t_{0.90}$	p-value
Que_%	0.59	5	0.82	1.48	
Mod_%	0.99	4	2.21	1.53	$p = 0.05$
Mod_Rate	0.90	4	2.87	1.53	$p = 0.02$

3.4. H_3—‘Good’ Structured Design Versus ‘Bad’ Object-Oriented Design

Table V presents a summary of the results of the statistical tests for the three dependent variables with respect to H_3 in the usual format. The results of this repeated analysis are surprising with respect to the variable Que_%—it was hypothesized that object-oriented concepts would cause understanding difficulties when badly designed yet the test does not indicate statistical significance, although it can be argued the effect size shows practical meaningfulness. The findings of Section 3.1 help explain this result—it is clear that the mean structured ‘good’ value of Que_% was lower than might otherwise be expected because of outliers. Consequently, this has affected our ability to detect a significant effect. For the two dependent variables concerned with the impact analysis, both achieve statistical and practical significance. Hence we accept the part of H_3 documenting that ‘good’ structured designs are easier to modify than ‘bad’ object-oriented designs. Note that the paired t-test eliminated some data points because of missing values—consequently, γ has been calculated from the data points which the test used.

3.5. H_4—‘Good’ Structured Design Versus ‘Bad’ Structured Design

Table VI presents a summary of the results of the statistical tests for the three dependent variables with respect to H_4. The results of this analysis are rather perplexing. Most striking, is that for Que_% the mean score for the ‘bad’ structured block is higher than that of the ‘good’ structured block—indicated by an * because it is in the opposite direction of the stated hypothesis. We have partially explained the reasons for this in Section 3.1. For the other two variables, there are no obvious explanations for the fact that the performance on the ‘good’ structured system was not significantly better than for the ‘bad’ structured system other than that, for this study, the effect was quite small.

Table 6. Results for ‘good’ versus ‘bad’ structured.

Variable	γ	df	F	Crit. $F_{0.90}$	p-value
Que_%	*	12	1.06	3.23	
Mod_%	0.50	11	0.75	3.29	
Mod_Rate	0.32	11	0.37	3.29	

Table 7. Results for ‘bad’ structured versus ‘bad’ OO.

Variable	γ	df	F	Crit. $F_{0.90}$	p-value
Que_%	1.22	12	4.59	3.23	$p = 0.06$
Mod_%	*	12	0.01	3.23	
Mod_Rate	0.22	12	0.15	3.23	

3.6. H_5—‘Bad’ Structured Design Versus ‘Bad’ Object-Oriented Design

Table VII presents a summary of the results of the statistical tests for the three dependent variables with respect to H_5. A significant result is achieved for Que_%, indicating that subjects had a better understanding of the ‘bad’ structured design documents than of the ‘bad’ object-oriented design documents. The first point to be raised is the apparent inconsistency between this result and the result of Section 3.4 with respect to the variable Que_%. By deduction, if poorly designed structured systems are easier to understand than badly designed object-oriented systems, it should hold that well designed structured systems are too. We have explained that this inconsistency occurred partly as a result of two low score outliers which unduly influenced the mean score for the ‘good’ structured block—subsequently, statistical significance was not achieved for Que_% for H_3. Here, on the other hand, statistical significance has been achieved. Hence, it is reasonable to assume that a practically significant effect also exists for this part of H_3.

Mod_% is the second anomaly noted in Section 3.1 as it is in the opposite direction of the hypothesis stated. However, the difference between the two means is almost negligible. There is also little of interest for variable Mod_Rate. Consequently, it seems there is little or no effect visible for modifiability.

3.7. Analysis Summary

We briefly summarise and review the results of the analyses in terms of evidence to support our hypotheses. We categorise this support into the following: strong support, i.e., the data shows statistical significance (at α level 0.10) and practical significance ($\gamma \geq 0.6$), weak support, i.e., the data shows practical significance but no statistical significance, and no support, i.e., the data has neither statistical nor practical significance.

Strong support. Statistical and practical significance was obtained for two of the dependent variables in support of H_2—‘Good’ object-oriented design is easier to understand

and modify than 'bad' object-oriented design. This result is consistent with the results of a correlational study by Basili *et al*. (1996). We also found significant and practical significance for ease of modification documented in H_3—'Good' structured design is easier to modify than 'bad' object-oriented design. In addition, significant and practical significance was discovered for ease of understanding documented in H_5—'Bad' structured design is easier to understand than 'bad' object-oriented design. An anomaly was discovered and explained for ease of understanding documented in H_3. By deduction, support is also provided for this part of H_3. These results are consistent with the opinions expressed by practitioners in (Daly et al., 1995).

Weak support. Practical significance was discovered for H_1—'Good' object-oriented design is easier to understand and modify than 'good' structured design. Care must be taken when interpreting this result in terms of ease of understanding because the anomaly discovered in the data suggests the effect size may actually be smaller than has been observed, i.e., the data is biased in the direction of object-oriented understanding.

No support. We found no practical significance to support either H_4—'Good' structured design is easier to understand and modify than 'bad' structured design, or ease of modification documented in H_5—'Bad' structured design is easier to modify than 'bad' object-oriented design.

4. Threats to Validity

This section discusses the various threats to validity of the study.

4.1. Construct Validity

Construct validity is the degree to which the independent and dependent variables accurately measure the concepts they purport to measure. The following possible threats have been identified:

1. Understandability and modifiability are difficult concepts to measure. We argue that the dependent variables used here are intuitively reasonable measures. Of course, there are several other dimensions of each concept, e.g., performing impact analysis is not the only important dimension of modifiability—making the actual changes is just as important. In a single controlled experiment, however, it is unlikely that all the different dimensions of a concept can be captured; the researcher must focus on what can be realistically achieved. Additional studies are required to investigate the other dimensions of modifiability. In future research, we also intend to supplement the dependent variables used with additional ones, e.g., accuracy of impact analysis.

2. There is no general consensus on what constitutes a 'good' and 'bad' object-oriented design and, therefore, the system designs used in this study may not be representative of these. On the other hand, recent empirical work tends to support the design principles

which were used, e.g., (Basili et al., 1996; Daly et al., 1996). Therefore, our choice of design principles seems to be more than reasonable.

4.2. Internal Validity

Internal validity is the degree to which conclusions can be drawn about the causal effect of independent variables on the dependent variable. The following possible threats have been identified: selection effects, non-random subject loss, interaction effect between problem domain and design technique, instrumentation effect, and maturation effect.

1. A selection effect occurs as a result of differences of ability between the groups of subjects. As random assignment was employed a selection effect would not normally be considered a threat to validity, but when the number of subjects is relatively small random assignment can become less effective. A selection effect might therefore explain the anomaly that occurred for understanding performance differences between 'good' and 'bad' structured. On the other hand, to create this difference of understanding, any selection effect would have to be quite large and therefore unlikely. If a large selection effect did exist it would be expected to influence other results in our study, i.e., H_1 and H_5. Our results do not suggest that this occurred.

2. Subjects dropping out from a study non-randomly can create differences in groups designed or intended to be equivalent. Of the twenty subjects who expressed an interest in the study only thirteen of them actually turned up to participate. The threat to this study arises from the fact that the randomization plans included all twenty subjects; because subject loss was non-random this left the groups A, B, C, and D with 2, 5, 2, and 4 subjects respectively. This of course could have meant the groups were no longer equivalent in terms of ability to perform the tasks, although having checked the debriefing questionnaires we found no evidence to suggest differences between the groups in terms of motivation and qualifications. However, we are uncertain of the effect this subject loss had on the outcome of the study.

3. An interaction effect between design method and problem domain would mean that subjects' performance was affected by both these variables. This might occur because the problem domains used are more or less suited to an object-oriented solution than a structured one, e.g., object-oriented design is not thought to be very useful for solving problems which mainly involve complex mathematical calculations because it is difficult to identify entities in the problem domain which represent real world objects. To counter this threat, the problem domains we used for OO were taken from examples in books detailing OO design methodology; as such, these domains were deemed to be amenable to an object-oriented design. If an interaction effect did exist, it does not explain why only small differences were found between the object-oriented and structured designs.

   To fully address this threat requires four different system designs for each domain—it could then be determined if such an interaction threat existed. However, this solution is hampered by two new problems: (i) the high cost involved developing 16 different designs and (ii) the number of subjects required to provide sufficient data points in each cell of the design, e.g., as with our design, if each subject were to participate twice, to obtain six data points per cell would require 48 subjects.

4. An instrumentation effect may result from differences in the experimental materials employed. The threat to this study was that possible differences between the four software systems other than those controlled (i.e., technique used to design them and design principles employed) were causing performance differences. As previously stated, a serious attempt was made to control for such a threat by ensuring that the same information was contained within each system documentation.

5. A maturation effect is caused by subjects learning as an experiment proceeds. The threat to this study was that subjects learned enough from the first experimental run to bias their performance in the second experimental run. The design controlled this confounding variable across the subjects, but in software engineering experiments it is usually stated as a potential threat. We have no evidence to suggest that this occurred.

The non-random subject loss threat is the most critical threat to this study. The lesson learned is that when designing randomization plans, ensure the subjects included in the plans are going to participate—in this study this would have meant drawing up the randomization plans once the subjects had arrived for the first experimental run. If we had performed this then there would not have been a threat to validity. Threats four and five are common to almost every software engineering controlled experiment and can rarely be completely controlled for—we made a serious attempt to eliminate them as best as possible.

4.3. External Validity

External validity is the degree to which the results of the research can be generalised to the population under study and other research settings. The following possible threats have been identified: subject representativeness and the materials used.

1. The subjects who participated in this study are unlikely to be representative of software professionals and therefore it is impossible to generalise the results to that population. However, it is argued that student based experiments can provide useful results for several reasons. First, they can be used to focus weak hypotheses on phenomena which appear to be important. These hypotheses can then be tested in more realistic settings with a better chance of important and interesting findings. Second, they can be used as a basis for deciding whether a hypothesis is worth investigating further in, e.g., an industrial case study. And third, they provide confirmatory power for any findings that are replicated in such a case study.

2. The materials used in this study, i.e., the software systems and tasks subjects were asked to complete, may not be representative in terms of their size and complexity.

We would emphasize the point that this research is regarded as exploratory and we are in the process of building upon it. While these two threats limit generalisation of this research it does not limit the results being used as the basis of future studies. It is also important to point out that weaknesses imposed by these two threats can be addressed if similar results can be obtained by using different empirical techniques—the idea is that the weaknesses of one study can be addressed by the strengths of another; see, e.g., (Daly, El Emam, and Miller, 1997; Kaplan and Duchon, 1988). For example, when the NASA SEL investigated the benefits of introducing OOD in their development process, they found that benefits were not immediate (Basili et al., 1992). Indeed, it took investments in training programmes, tailoring of OOD, and reuse before tangible benefits were gained over previous structured development practices. This NASA SEL field study has a strong external validity given the research setting and developers investigated; it tends to support the findings of this controlled experiment with respect to inexperienced developers not being provided with immediate benefits from using object-oriented technology. On the other hand, such a field study has weak internal validity in the sense that it is difficult to determine what factor(s) actually provided most of the benefits received, e.g., would similar benefits have been obtained without investment in object-oriented reuse?

5. Conclusion

This study has investigated two different effects with respect to understandability and modifiability of system design documents, two essential components of maintainability. First, it has compared designs developed by means of object-oriented and structured techniques. And second, it has investigated the use of perceived ‘good’ and ‘bad’ design principles from Coad and Yourdon and their influence on the resulting system design documents.

An interpretation of the results based solely in terms of the stated hypotheses, however, is not possible for the following reasons. One, the power of the statistical tests applied seem to be too low to detect all existing effects at the set α level, even though our data did show several interesting trends. Two, the various threats to external validity limit our ability to generalise the results—in this instance, we plan to use the results of this student based experiment to facilitate further investigation. However, our data do support some plausible interpretations. First, we found little evidence (i.e., some practical significance was identified but statistical significance was not achieved) to suggest that maintainers with little experience gain great benefit maintaining object-oriented designs over structured designs. This implies that when an organisation introduces object-oriented design techniques into their development process, proper training would appear to be a crucial activity if significant maintenance benefits are to be achieved; and the learning curve must be completed, i.e., the maintainers are no longer inexperienced, before any real benefits can be achieved. Second, our results suggest (with statistical significance) that adherence to ‘good’ object-oriented design principles will provide ease of understanding and modification for the resulting design when compared to an object-oriented design to which to the principles have not been adhered to. And third, we found significant evidence to suggest that an object-oriented design which did not adhere to quality design principles is likely to cause more understanding and modification difficulties than an appropriate structured design, i.e., abuse of object-oriented concepts apparently adds significantly to cognitive complexity. Also, the difference between the structured design obeying quality principles and the one which did not was not found to be significant. Although little should be read into this particular result, it does lend some support to the argument that it may be more important to follow stringent quality standards when using object-oriented development techniques. Finally, these results suggest that switching developers proficient in structured techniques to object-oriented techniques may, in relative terms, actually have negative effects on the designs they produce until they become as proficient with the object-oriented techniques.

In software engineering, to answer the type of questions we are addressing here, we usually expect to have work with small sample sizes—it is common to work with a sample of convenience, e.g., students in a programming class or with professional programmers during a training session. It is quite difficult and expensive to obtain large subject samples, something which can usually only be achieved through sufficient motivation to participate as well as sufficient funds. Consequently, we conclude that the power of statistical tests is an important factor when interpreting non significant results. Performing power analysis as well as external replications are necessary to achieve significant, reliable and generalisable results. In addition, it is likely that consistent data will have to be collected from different studies and integrated to allow meta-analyses to be performed (Judd, Smith and Kidder, 1991; Rosnow and Rosenthal, 1989)—for smaller effect sizes it may be extremely difficult for an individual experiment to obtain the required number of data points to achieve significance. To be plausible, collaboration between different research groups is necessary, an objective of research networks such as ISERN (International Software Engineering Research Network). Finally, we would stress the importance of debriefing subjects—the information gained can help explain anomalies in the data as well as support the quantitative results or aid alternative interpretations.

Further research planned as a result of this empirical research includes investigation into what constitutes a ‘good’ and ‘bad’ object-oriented design and identification of other variables which are valid measures of the concepts of understandability and modifiability.

Acknowledgments

We wish to thank the students within the Computer Science Department at the University of Kaiserslautern who took the time to participate in this empirical study and Dieter Rombach for making the study possible. Finally, we acknowledge the efforts of Barbara Kitchenham, Filippo Lanubile, Oliver Laitenberger, Shari Lawrence Pfleeger, and the anonymous reviewers for providing constructive comments which significantly improved the contents of this paper.

Notes

1. The Vordiplom is the initial set of exams which students have to pass after (at least) two years at University. The qualification requires passes in theoretical, technical, and practical computer science, mathematics, and a fifth elective class.
2. This information was captured by asking each subject to complete a questionnaire which characterized their background in terms of experience, qualifications, knowledge of structured and object-oriented techniques, etc.

3. This requirement was necessary because the researchers promised to provide subjects experience with different types of design techniques as well as experience with designs constructed by practices perceived to be 'good' and 'bad'. The effect this would have on the experimental design was overlooked at the time.
4. Note, however, that one subject did not attempt the impact analysis tasks for the 'good' structured design—consequently, there are only six data points for the variables Mod_% and Mod_Rate.
5. Power calculations are also performed to help researchers estimate how many subjects are required to have a reasonable chance (usually 0.8) of achieving a statistically significant result for a given effect.
6. Whether the test is repeated-measures or not also affects the power of the test. This option is directly dependent on the experimental design being within-subjects; so it is not a component which can be manipulated in the same way as α.

References

Arnold, R. 1993. Impact analysis—towards a framework for comparison. *Proceedings of the IEEE Conference on Software Maintenance*, 234–243.

Basili, V., Briand, L., and Melo, W. 1996. How reuse influences productivity in object-oriented systems. *Communications of the ACM* 39(10): 104–116.

Basili, V., Briand, L., and Melo, W. 1996. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering* 22(10): 751–761.

Basili, V., Caldiera, G., McGarry, F., Pajerski, R., Page, G. and Waligora, S. 1992. The software engineering laboratory—an operational software experience factory. *Proceeding of the IEEE International Conference on Software Engineering*, 370–381.

Briand, L., Bunse, C., and Daly, J. 1997. An experimental evaluation of quality guidelines on the maintainability of object-oriented design guidelines. Technical Report ISERN-97-02, Fraunhofer Institute (IESE), Kaiserslautern, Germany.

Brooks, R. 1980. Studying programmer behavior experimentally: The problems of proper methodology. *Communications of the ACM* 23(4): 207–213.

Coad, P., and Yourdon, E. 1991. *Object-Oriented Analysis*. Second edition, Prentice-Hall.

Coad, P., and Yourdon, E. 1991. *Object-Oriented Design*. First edition, Prentice-Hall.

Cohen, J. 1969. *Statistical Power Analysis for the Behavioral Sciences*. First edition, Academic Press.

Curtis, B. 1980. Measurement and experimentation in software engineering. *Proceedings of the IEEE* 68(9): 1144–1157.

Daly, J., Brooks, A., Miller, J., Roper, M., and Wood, M. 1996. Evaluating inheritance depth on the maintainability of object-oriented software. *Empirical Software Engineering, An International Journal* 1(2): 109–132.

Daly, J., El Emam, K., and Miller, J. 1997. An empirical research methodology for software process improvement. Technical Report ISERN-97-04, Fraunhofer Institute (IESE), Kaiserslautern, Germany.

Daly, J., Miller, J., Brooks, A., Roper, M., and Wood, M. 1995. A survey of experiences amongst object-oriented practitioners. *Proceedings of the IEEE Asia-Pacific Software Engineering Conference*, 137–146.

DeRemer, F., and Kron, H. 1976. Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering* SE-2(2): 80–86.

Devore, J. 1991. *Probability and Statistics for Engineering and the Sciences*. Third edition, Brooks/Cole.

Jones, C. 1994. Gaps in the object-oriented paradigm. *IEEE Computer* 27(6): 90–91.

Judd, C., Smith, E., and Kidder, L. 1991. *Research Methods in Social Relations*. Sixth edition, Harcourt Brace Jovanovich, Inc.

Kaplan, B., and Duchon, D. 1988. Combining qualitative and quantitative methods in information systems research: A case study. *MIS Quarterly*: 571–586.

Kraemer, H., and Thiemann, S. 1987. *How many subjects?* First edition, Sage Publications.

Kung, D., Gao, J., Hsia, P., Wen, F., Toyoshima, Y., and Chen, C. 1994. Change impact identification in object-oriented software maintenance. *Proceedings of the IEEE International Conference on Software Maintenance*, 202–211.

Miller, J., Daly, J., Wood, M., Brooks, A., and Roper, M. 1997. Statistical power and its subcomponents—Missing and misunderstood concepts in empirical software engineering research. *Information and Software Technology* 39(4): 285–295.

Moen, R., Nolan, T., and Provost, L. 1991. *Improving Quality Through Planned Experimentation*. First edition, McGraw-Hill, Inc.

Rosnow, R., and Rosenthal, R. 1989. Statistical procedures and the justification of knowledge in psychological science. *American Psychologist* 44(10): 1276–1284.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. 1991. *Object-Oriented Modeling and Design*. Prentice Hall.

Slakter, M., Wu, Y., and Suzuki-Slakter, N. 1991. *, **, and ***; statistical nonsense at the .00000 level. *Nursing Research* 40(4): 248–249.

van Hillegersberg, J., Kumar, K., and Welke, R. 1995. An empirical analysis of the performance and strategies of programmers new to object-oriented techniques. *Psychology of Programming Interest Group: 7th Workshop*.

Wilde, N., and Huitt, R. 1992. Maintenance support for object-oriented programs. *IEEE Transactions on Software Engineering* SE-18(12): 1038–1044.

Lionel C. Briand received the B.S. degree in geophysics and the M.S. degree in Computer science from the university of Paris VI, France. He received the Ph.D. degree, with high honors, in Computer Science from the university of Paris XI, France.

Lionel is currently the head of the Quality and Process Engineering Department at the Fraunhofer Institute for Experimental Software Engineering (FhG IESE), an industry-oriented research center located in Rheinland-Pfalz, Germany. His current research interests and industrial activities include measurement and modeling of software development products and processes, software quality assurance, domain specific architectures, reuse, and reengineering. He has published numerous articles in international conferences and journals and has been a PC member or chair in several conferences such as ICSE, ICSM, METRICS and SEKE. He is also serving on the steering committee of ICSM.

Before that, Lionel started his career as a software engineer at CISI Ingénierie, France. He then joined, as a research scientist, the NASA Software Engineering Laboratory, a research consortium: NASA Goddard Space Flight Center, University of Maryland, and Computer Science Corporation. Before going to FhG IESE, he held the position of lead researcher of the software engineering group at CRIM, the Computer Research Institute of Montreal, Canada.

Christian Bunse received the B.S. degree (Vordiplom) and the M.S. degree (Diplom) in computer science from the University of Dortmund, Germany. From 1993 to 1995 he was a faculty research assistant of the Software-

Technology-Transfer-Initiative at the University of Kaiserslautern, Germany. He is currently with the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern. His research interests are in the area of software engineering, especially 'Cleanroom Software Engineering', object-orientation, verification techniques, and their transition into industrial practice.

John W. Daly received the B.Sc. and Ph.D. degrees in Computer Science from the University of Strathclyde, Glasgow, Scotland in 1992 and 1996 respectively. He is currently a software engineering researcher in the Fraunhofer Institute (FhG IESE), Kaiserslautern, Germany where he has been employed since April, 1996.

John's current research interests and industrial activities include software measurement and software process improvement, conducting empirical research by means of quantitative and qualitative methods, and object-oriented development techniques.

Christiane Differding received the M.S. degree (Diplom) in computer science from the University of Kaiserslautern, Germany in 1993. From 1993 to 1995 she conducted an industrial quality improvement program of the Software-Technology-Transfer-Initiative at the University of Kaiserslautern. Currently she is a teaching and research assistant of the Software Engineering Group, Department of Computer Science, University of Kaiserslautern. Her research interests are in the area of software quality improvement and measurement, especially goal-oriented measurement approaches.