Fraunhofer Einrichtung Experimentelles Software Engineering

A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs

Authors:
Lionel C. Briand
Christian Bunse
John W. Daly

Submitted for Publication in
IEEE Transactions on Software Engineering

IESE-Report No. 002.99/E
Version 1.1
May 1999

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

This paper presents a controlled experiment focusing on the impact of applying quality design principles such as the ones provided by Coad and Yourdon on the maintainability of object-oriented designs.

Results, which repeat the findings of a previous study, strongly suggest that such design principles have a beneficial effect on the maintainability of object-oriented designs. It is argued that object-oriented designs are sensitive to poor design practices because the cognitive complexity introduced becomes increasingly unmanageable. However, as our ability to generalize these results is limited, they should be considered as preliminary, i.e., it is very likely that they can only be generalized 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: design documents, experiment, maintainability, object-oriented, replication.

Table of Contents

1 Introduction

Interest in object-oriented techniques has been rising as more and more companies have switched to an object-oriented methodology for developing their software systems. Today object-oriented analysis and design methods, languages, and development environments are visible in both small and large software organizations alike. This situation occurred mainly as a result of opinion and anecdotal evidence provided by vendors and experts; not really as a result of empirical evidence demonstrating that these techniques offer significant advantages when compared to different techniques. Jones [18], for example, identified several areas where there was a distinct lack of empirical evidence to support the assertions of gains in productivity and quality, reduction in defect potential and improvement in defect removal, and reuse of software components.

Interesting 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 to them, as it has been suggested by its advocates in the past. For example, positive results have been provided by Basili et al. [2] 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. Deubler and Koestler's [16] experience with object-oriented technology lead them to believe the soft-ware was more structured, had substantially less errors, and had less interdependencies among components. Not so favorable empirical evidence was provided by van Hillergersberg et al. [24] 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. [14] provided evidence which suggests that inheritance depth and conceptual entropy of class hierarchies can cause programmers difficulty when trying to maintain 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 organization switching to object-oriented techniques would be likely to save large amounts of money throughout the lifetime of an object-oriented system. It is clear, however, that object-oriented techniques can only yield benefits if they are applied properly. An important issue, therefore, is to find out what makes a “good” object-oriented design. Design principles have been proposed

[11] [12] but they are usually defined at an intuitive level and require further interpretation to be applicable. In addition, it is unclear whether such design principles actually increase the quality of the resulting software with respect to some interesting external attribute, i.e., they have not been empirically investigated to a sufficient extent to be relied upon; they should therefore be considered as hypotheses rather than guiding principles.

This paper presents an empirical study which investigates two important components of design maintainability, namely its understandability and modifiability, by comparing the effect of design principles perceived to be `good' and `bad' practice, on these attributes. The paper is partitioned as follows. Section 2 presents the details of the experiment. Section 3 summarizes the data collected and presents the data analysis results. Section 4 identifies and discusses possible threats to the validity of the study. The data suggest with statistical significance that adherence to good object-oriented design principles provides practically significant benefits to object-oriented design documents in terms of ease of understanding and modifiability.

2 Description of the experiment

2.1 Introduction

The research reported in this paper builds upon the results of a previous experiment [5] which investigated the benefits of the quality design principles by Coad and Yourdon [11][12] with respect to the maintainability of object-oriented and structured design documents. The objective was two-fold: (1) assess the effect, in terms of maintainability, of applying Coad and Yourdon principles to object-oriented and structured designs, (2) compare the maintainability of structured and object-oriented designs. This paper concentrates solely on the investigation of the use of quality design principles and their influence on a developer's ability to understand and modify object-oriented design documents — a detailed description of the quality design principles of interest can be found in Section 2.2. There are several reasons for our decision to concentrate only on object-oriented design quality. First and foremost, the original study suffered from a lack of data points in each cell of the design, making it difficult to achieve statistical significance for all but the largest observed effects. This time, it was decided to increase the power of the study by focusing on one main effect and using all data points to investigate this effect (rather than halving the number of data points over two main effects as occurred last time). Second, by concentrating our efforts in this manner, we shall have more data to analyze for interesting trends, e.g., particular aspects of the designs that cause difficulty or facilitate ease of understanding or modification. Third, object-oriented development techniques are being used in industry — so regardless of the validity of the question "are object-oriented development techniques 'better' than structured ones?" there is a need to identify quality object-oriented design principles that developers should follow. Moreover, there is a need to evaluate these design principles quantitatively in order to understand the difference they make to the quality, e.g., maintainability in this study, of object-oriented designs. That is, not are we only concerned with achieving statistical significance in this study, but also achieving something of practical significance.

The investigation utilizes and improves upon the object-oriented materials from the original study (see Section 2.10 for details of the improvements made). In addition, the hypotheses of this study are a subset of the hypotheses from the original study (see Section 2.3). Consequently, using the framework of Brooks et al. as a reference [7] the study can be classified as an internal replication — that is, a replication conducted by the same set of researchers that performed the original study. The replication framework of Brooks *et al.* [7] provides a classification scheme for replications along three different dimensions of an experiment **(method, tasks, subjects)**. Accordingly, we would classify this inter-

nal replication as **(similar, improved, improved)**. **Method** is classified similar because it is the same method used in the original study. **Tasks** are classified improved because they were modified to test the hypotheses more thoroughly, i.e., they were further focused on understanding and modifying the parts of the system affected by Coad and Yourdon's design principles. **Subjects** are also classified as improved because, although the same subject pool was used, the number of subjects was far greater and the debriefing questionnaire elicited more detailed information.

2.2 Quality design principles of Coad and Yourdon

Coad and Yourdon [11], [12] identify a set of quality design principles which they advocate, if adhered to, will result in a better object-oriented design. These design principles 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 decreasing 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 specialization class is indeed a genuine specialization of its generalization class.

Cohesion. First, a service in a class should carry out one, and only one, function. Second, the attributes and services should be highly cohesive, i.e., no unused attributes and services and they should all be descriptive of the responsibility of the class. Third, a specialization should actually portray a sensible specialization — it should not be some arbitrary choice which is out of place within the hierarchy creating a less cohesive class due to unrelated inherited features.

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 in scope.

Generalization-Specialization depth. It is important not to create specialization classes which are conceptually not a real specialization, e.g., created for the sake of reuse. Rather an inheritance hierarchy should capture a conceptual taxonomy used to model the problem at hand.

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. A class should map to a type of entity in the problem description. 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. It is also important to note that some of these principles are interdependent. It is very difficult to change cohesion without affecting coupling and both coupling and cohesion have an impact on the perceived simplicity of objects and classes. However, as further discussed below, our main goal here is to determine whether companies applying these principles as a whole, for example through appropriate quality assurance and control procedures, would significantly benefit in terms of improved maintainability.

2.3 Hypotheses

To be able to test the hypotheses below, two different object-oriented system designs were required. These system documents were designed according to the design principles presented in 2.2 above, the good system designed to adhere as best as possible to the design principles and the bad system being changed to prevent the principles to be fully adhered to — the OMT notation of Rumbaugh *et al.* [22] was used to represent both designs. As a result, the design which did not obey the design principles of Coad and Yourdon had additional coupling between classes, specialization 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.

Section 2.9 details the application domains of the systems and illustrates the difference between the systems through various applicable object-oriented design measures. In addition, it will be shown how the design that transgresses Coad and Yourdon principles has been obtained through a realistic degradation of an originally "good" design. Due to space constraints the designs cannot be provided here, but they are described in the replication package of this experiment [9]. Appendix B, however, describes in more detail the types of changes applied through the degradation process. Section 2.5 lists a number of arguments that support the claim that the two designs being compared are actually comparable for the purpose of our experiment.

Of interest are the concepts of understandability and modifiability (see Section 2.9). 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 design. 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 de-

fined as the activity of identifying what to modify to accomplish a change; see for example [1]. 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 these effects, the null hypothesis being stated as

H_0 — There is no significant difference between 'good' and 'bad' object-oriented design in terms of ease of understanding and impact analysis.

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

H_1 — 'Good' object-oriented design is significantly easier to understand than 'bad' object-oriented design.

H_2 — It is easier to perform impact analysis (locate changes) on a 'good' object-oriented design than on a 'bad' object-oriented design.

H_1 and H_2 are stated on the basis that when sensible design principles are applied they will aid the maintainability of the resulting system documents through increased understandability and ease of impact analysis. Note that these hypotheses represent a subset of the hypotheses presented in [5].

Rather than studying each principle separately, we examine the Coad and Yourdon principles as a whole for a number of reasons:

- As discussed above, there exists some strong dependencies between principles and it would be extremely difficult, if at all possible, to look at coupling, cohesion, and class simplicity separately.

- As a first step, we want to see whether the application of these principles as a whole have any practical significance before designing more complicated experiments where the various principle effects would have to be isolated from each other.

- In practice, it is very likely that quality assurance (QA) procedures include inspections or formal reviews of design documents that will check for the conformance to all of Coad and Yourdon's principles. When such QA procedures do not take place, all types of violations to these principles are usually observed during OO developments. Therefore, it still makes practical sense that a cost-benefit analysis of such QA procedures should encompass all Coad and Yourdon principles, although their individual impact is also of scientific and practical interest.

2.4 Subjects

The experimental subjects used in the study were all students from the Department of Computer Science at the University of Kaiserslautern who were enrolled in a semester long class on Software Engineering. During the lectures, subjects were taught basic software engineering principles and were introduced to structured and object-oriented 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 informed that a series of experiments were planned and were asked to participate. Students were motivated by making it clear that they would gain valuable experience from participating during the subsequent training and experimental sessions. As a result, 33 subjects agreed to take part. These subjects were then given an intensive training session before the experiment took place (see Section 2.7 for details).

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 of those who volunteered only three did not have their Vordiplom.[1] In general, *before* attending Software Engineering I the subjects had little experience [2] with (a) software practice — median response 2 (min 1, max 3), (b) design documents — median response 2 (min 1, max 3), (c) object-oriented design documents — median response 2 (min 1, max 3), and (d) performing impact analysis — median response 2 (min 1, max 3). However, the subjects were fairly motivated to participate (median 3 from an ordinal scale of 1 — not motivated to 5 — highly motivated).

2.5 Experimental materials

Two object-oriented designs were used to test the hypotheses. Each design documentation was approximately thirty pages and included, in addition to the design itself, a general system description and a requirements document. To make them comparable, each set of documentation was intended to be as similar as possible in terms of the layout of the information and information content (see [1] for more details). As a first indication regarding the information content of designs, Table 1 provides simple counts related to artifacts and relationships. As noted in Section 2.3, the bad object-oriented system was obtained by modifying an originally good design (i.e., complying with Coad and Yourdon

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 means of the debriefing questionnaire based on the ordinal scale of 1 - no experience to 5 - professional experience; see Appendix A.

principles) by including additional coupling between classes, inappropriate inheritance, less cohesive classes, and so on. To quantify the degradation applied to this design, we used counts of design artifacts and relationships as well as standard design measures (see below discussion on Table 2). Table 1 provides artifact and relationship counts for the original design from which the bad design was obtained ("Bad Original"), the bad design itself ("Bad Used"), and the good design ("Good Used"). From these counts we can see the bad used design shows significantly more associations, attributes, inheritance, and classes. The bad original design and the good used design are comparable in terms of several counts but show differences in terms of aggregation and inheritance relationships and number of attributes. Although we tried our best to use functional designs of comparable size and complexity, there are still differences that may be considered as threats to the validity of the experiment. This will be further discussed in Section 4. In other words, despite some differences, we hope that the variations between the two compared designs (Bad used and Good used) are mainly due to the violations of Coad and Yourdon's design principles, that is, the phenomenon we are studying.

Counts	*Bad Original*	*Bad Used*	*Good Used*
classes	14	18	13
inheritance relationships	4	8	2
Associations	12	16	9
Attributes	24	30	16
Operations	20	25	18
max. inheritance Depth	1	3	1
Inconsistencies	0	8	0
Aggregations	2	2	8

Table 1 Artifact and relationship counts for each system design

Turning now our attention to the design measurement results in Table 2, we used the DIT (depth of inheritance tree), NOC (number of children), and CBO (coupling between objects) measures [10], which are well-researched measures known to capture class inheritance and coupling. These measures are often considered to be related to the complexity of designs [10]. Table 2 provides these measurements for the three designs mentioned above. Note that there are no measures used to capture class cohesion. This is because the information required by existing measures is not usually available at high level design, e.g., method-attribute interactions, method-method interactions, and pairs of methods which reference common attributes [6]. None of this information was present in the system designs, so cohesion cannot be compared across systems in a quantifiable manner. However, it should be pointed out that since cohesion and coupling are expected to be interdependent, this is not considered a serious issue in the context of this study.

Measure	*Bad Original*	*Bad Used*	*Good Used*
DIT	0.29	0.67	0.22
NOC	0.29	0.45	0.22
CBO	2.90	3.10	2.6

Table 2 The mean values for DIT, NOC, and CBO for each system design

For the same reasons as those for Table 1, we performed design measurement to illustrate two things. First, we wanted to quantify the effect of the degradation of the original bad design in terms of standard coupling and inheritance measurement. Second, we wanted to demonstrate that there were significant differences in terms of coupling and inheritance between the compared good and bad designs. Table 2 presents the mean values, over all classes, of the three measures[3] and for the three designs mentioned above. As can be seen from the measurement means the larger measures for the used bad design than for the original bad design show that the degradation introduced, on average, more coupling and more (inappropriate) inheritance. Second, the smaller values for the good used design than for the bad used design show it contains less coupling and less (inappropriate) inheritance. Therefore, similarly to the analysis of artifacts and relationship counts above, this difference in terms of structural properties between the bad used and good used designs makes them suitable for our experiment. Last, it can be seen that the bad original design is comparable, in terms of inheritance and coupling measurement, to the good used design. This further supports the idea that the originally selected designs are comparable with respect to their information content and that the differences between the bad used design and the good used design are mainly due to the degradations applied to the former, that is the phenomenon we wish to study here.

From the discussion above, it can be argued that a serious attempt was made to keep the differences between the good and bad design system documents to those caused by (a) the design principles applied and (b) the different application domains — there was no additional information given in either system documentation. The application domains used for the designs were (i) good system — a temperature controlling system and (ii) bad system — an automatic

3 There exists a problem with calculating the mean CBO value for system classes — the mean is sensitive to changes in an inheritance hierarchy which do not involve additional class coupling in the system, i.e., if a class is added to a hierarchy, without additional coupling being added, then the mean CBO value for the system classes is reduced. This prevents accurate comparison of coupling across the systems since such a coupling measure is somewhat confounded with the inheritance structure depth. A measure is needed to compare coupling across different systems independently of the inheritance depth, e.g., the original bad system and the used bad system. The solution adopted was to calculate the mean CBO value for inheritance hierarchies rather than for individual classes. Therefore, we keep the same measurement procedure, but change the unit of analysis. Consequently, a CBO value was calculated for each hierarchy in the system, the values were summed and then divided by the number of hierarchies to compute the mean hierarchy CBO for the system. Classes which were not part of an inheritance hierarchy were just treated as a base class with no descendents, i.e., CBO was calculated as normal for these classes. This solution, in effect, allows comparisons whose outcome is not affected by differences or changes in inheritance structure.

teller machine. It is important to note that, despite the difference in application domains and based on the typical students curriculum at the university of Kaiserslautern, we have no reason to believe that one application domain is more familiar than the other to the students involved in the experiment. This was also supported by the results of the debriefing questionnaire as discussed in Section 4.2. Additional justifications regarding the experiment design choices made and the comparability of designs, are also provided in Section 4.2 when addressing the internal validity of our experiment. Appendix B provides additional information regarding the degradations applied to the original bad system.

2.6 Experimental tasks

For each system document there were two sets of tasks to be performed. First, subjects had to complete a questionnaire (see Appendix C for generic descriptions of the type of questions asked and examples) 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 system documentation provided. Each questionnaire contained exactly the same number of questions (7) and the questions were conceptually similar and in identical order.

The second task required subjects to undertake two separate impact analyses, i.e., mark all places with the system documentation that have to be changed, but not make the actual changes themselves (see Appendix D for examples). First, impact analysis had to be performed on the system design as a result of a change in customer requirements. Second, impact analysis had to be performed on the system documentation this time as a result of an enhancement of system functionality. The number of change places to be found to complete the impact analyses were 21 and 22 for the good and bad systems, respectively. 14 of them were due to a requirement change and the remainder to new requirements. In the bad system, all changes affected classes that were affected by the degradations of the original bad system.

After completion of the tasks subjects were asked to complete a debriefing questionnaire. This questionnaire captured (i) personal details and experience, (ii) opinions with respect to a subject's motivation and performance, e.g., what approach did they adopt to complete the tasks, how accurate and complete did they think their impact analyses were, what had caused them the most difficulty, and (iii) opinions on the experiment itself, e.g., size of the systems, difficulty and realism of tasks. (A copy of the questionnaire is provided in Appendix A).

2.7 Procedures

The week before the experiment proper took place, subjects were given an additional practical session as training. The training session was essentially an interactive dry run of the experiment proper. Subjects were given the system documentation for a university scheduling system. Subjects then had to perform some understanding tasks and one impact analysis task. The idea behind the training session was to familiarize subjects with the experimental setting and procedure, how to perform impact analysis in the required manner, and, more generally, answer any questions they had. At the end of the training session the subjects were told to review what they had learned before participating in the experiment proper.

The experiment proper was then performed over two consecutive days with each subject receiving a different system documentation each day (see section 2.8 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 system documentation. Each subject sat next to a subject who was examining the other system documentation — 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, i.e., what hypotheses were being tested. The subjects were then given a maximum of one and a half 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, subjects completed their debriefing questionnaire and then returned this and all experimental materials to a monitor before leaving.

2.8 Design

As depicted in Table 3 a standard within-subject 2 x 2 factorial design was employed; see, e.g., [23]. The two independent variables being the experimental run (*X* — run 1 and run 2) and the design principles applied to the object-oriented system (*Y* — good OO and bad OO). An important advantage of using a within-subjects design such as this is that the error variance due to differences among subjects is reduced. From our experience in performing software engineering experiments [14][5][7] and the literature [13], we believe that, when dealing with small samples, variations in participant skills is a major concern that is difficult to fully address by randomization or blocking. At the same time, it can result in the independent variables becoming confounded with the order of presentation which can lead to learning and fatigue effects. To control for this, counterbalancing was introduced, i.e., half the subjects were presented with the tasks for the good OO system in experimental run 1 and then the tasks

for the bad OO system in experimental run 2 (group A). The other half of the subjects did the opposite (group B). This is illustrated in Table 3.

Variable X	Variable Y --- *System*	
Run	Good OO	Bad OO
1	A	B
2	B	A

Table 3 Experimental design employed

To further control for learning and fatigue effects and differences between subjects, random assignment to these two groups was performed. This was achieved by drawing a letter for each subject from a hat. Once this had been performed subjects were then shown to a desk with the appropriate system documentation. As the number of subjects was known before running the experiment it was a simple procedure to create two groups of equivalent size, which is important to prevent the independent variables from becoming non-orthogonal. Each subject remained in the same group for the second experimental run.

2.9 Dependent variables and their collection procedures

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 summarize the places identified. This allowed the validity of the form to be cross checked. The time to complete these tasks was also recorded. From this data six sensible dependent variables are derived.

Und_Time which represents the time spent on understanding the system in order to complete the questionnaire.

Und_Corr which represents the correctness of the understanding questionnaire, i.e., the number of questions correctly answered. As the questionnaire was used to gauge the subjects' understanding of the design (and both understanding questionnaires had the same number of questions), it is reasonable to use the number of correct answers as a measure of this understanding.

Mod_Time which represents the time spent identifying places for modification.

Mod_Comp which represents the completeness of the impact analysis, i.e., the number of places to be changed that were found. Completeness, normalized on the scale of 0 to 1, is calculated as

$$Mod\_Comp = \frac{number\ of\ correct\ places\ found}{total\ number\ of\ places\ to\ be\ found} \quad (1)$$

It is reasonable to measure the effectiveness of a modification by the amount of places to be changed found relative to the total number to be found.

Mod_Corr which represents the correctness of the impact analysis, i.e., the number of places indicated to be changed which were correct. Correctness, normalized on the scale of 0 to 1, is calculated as

$$Mod\_Corr = \frac{number\ of\ correct\ places\ found}{number\ of\ places\ indicated\ as\ found} \quad (2)$$

It is reasonable to measure the correctness of a modification by the correct amount of places to be changed found relative to the total number of places identified to be changed.

Mod_Rate which represents the modification rate. The modification rate is calculated as

$$Mod\_Rate = \frac{number\ of\ correct\ places\ found}{Mod\_Time} \quad (3)$$

It is reasonable to measure the efficiency of a modification by the number correct places found per time unit, i.e., in this case per minute.

In addition, qualitative data from the debriefing questionnaire (Appendix A) was collected.

2.10 Improvements made for the replication

Several changes were made to the materials from the original study to further isolate the effects being investigated. In addition, several steps were taken to ensure that several threats to validity which occurred in the original experiment did not occur again. We now describe the improvements made.

Dependent variables. In this study, correctness of the impact analysis is measured, something which was not considered in the original study.

This measure further illustrates the effectiveness of a given impact analysis.

More thorough subject debriefing. The importance of debriefing questionnaires in subject-based experiments is often overlooked in software engineering. For this experiment, additional time was spent producing a more thorough and comprehensive debriefing questionnaire. The new debriefing questionnaire also incorporates questions which were derived from issues uncovered in the original experiment, e.g., there was no information about the procedure subjects adopted when tackling the tasks nor any information about what had caused them the most difficulties in understanding and modifying the designs.

Number of subjects. In this study 33 subjects participated compared to 13 subjects in the original study. Furthermore, because the design employed was completely within-subjects, each subject was required to contribute an observation for each system (although due to various human factors this did not occur completely — Section 3 explains this in detail). Consequently, given the identical effect being investigated by both studies [4], this study will have greater statistical power.

Randomization plans. The original randomization plans, because they were conducted prior to the experiment being run and subjects not turning up on the day, led to an unbalanced number of subjects in each group. This made it difficult for certain analyses to determine if the independent variables were confounded with the order of presentation. The randomization plans used here, because they were employed just before the first experimental run began, ensured that an equal number of subjects was allocated to each group.

Systems. In the original experiment there were several inconsistencies between the object-oriented systems, which could be classed as noise — although no evidence was found to suggest that this noise subsequently affected the results. The inconsistencies which needed to be addressed were (a) there was no standard layout between the systems, e.g., one system had documents clearly marked as system description, requirements, and design whereas in the other system this was not so clearly marked (because of photocopying difficulties), (b) customer and developer requirements were present in one system, but were not present in the other, (c) the notation used across the systems was not entirely standardized, and (d) there were some differences between the systems in terms of the information contained other than that resulting by either the domain of the system or the design principles applied, e.g., more detailed

[4] We must point out that this assumption might not necessarily be true given that improvements to the experimental systems and tasks have been made.

description of classes was presented in the good system. For this study all these inconsistencies were removed. For this experiment we are more confident that any differences between the two system documents could only be caused by (a) the design principles applied and (b) and different application domains used. The latter source of variation would obviously be a serious threat to our internal validity (see Section 4.2). However, as discussed in Section 2.5, we have provided evidence that the two designs were comparable in terms of information content and complexity, when both conformant to Coad and Yourdon principles. In addition, we have made sure to select application domains that were similar in terms with respect to the level of familiarity of the students who follow a typical computer science curriculum at the University of Kaiserslautern. Last, as discussed in Section 4.2 in more detail, the alternative strategy consisting of using the same application domain and design to perform this experiment would have resulted in other, more serious validity problems.

Tasks. In this study a serious attempt was made to make the tasks more comparable in terms of complexity and in terms of the number of understanding questions to answer and the number of places to be identified for the impact analysis tasks. More importantly, the tasks were modified to test the hypotheses more thoroughly, i.e., they were further focused on understanding and modifying the parts of the system affected by Coad and Yourdon's design principles.

As noted in Section 2.1, we classify the internal replication of the object-oriented part of [5] as (similar, improved, improved) because of these improvements [7].

2.11 Data analysis procedure

Data was collected for all subjects over the two experimental runs (one subject was unable to return for their second run due to prior commitments). Therefore, thirty three data points were available for analysis for the good system and thirty one data points were available for the bad design; because the design was completely within-subjects, repeated measures analysis can be performed. The first step of the analysis procedure is to check the normality of the data — if the data is substantially non-normal then the appropriate tests to use are non-parametric ones; if not then parametric tests can be applied.

To proceed with the analysis, we first 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 H_0, have to be determined. In our application context, that would mean the cost of applying useless design principles,

e.g., cost of training, quality assurance procedures. 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 quality design standards and the maintainability of object-oriented design with a high level of confidence.

A practical perspective: Are quality design standards more likely to be significantly beneficial to object-oriented designs than detrimental?

From a scientific perspective, it is necessary to work at a low α level (usually quoted as $\alpha = 0.05$ or $\alpha = 0.01$). From a practical perspective, such a stringent level of significance may not be required depending on the cost of applying this type of quality standards. It is basically a matter of minimizing risks and the expected loss. An α as high as 0.2, or more, might therefore be considered good enough to make an informed decision, even though the empirical evidence is not strong enough to make a scientific statement with a high degree of confidence.

Another factor to be considered is the potential situation of the results not being statistically significant — there are two possible interpretations: (i) the effect does not exist and H_0 is retained or (ii) the experiment does not have sufficient statistical power (expressed as $1-\beta$[5]) to detect the effect investigated (usually because the sample size is too small). To reduce the chance of a Type II error being committed, i.e., incorrectly retaining the false H_0, it is necessary to perform power analysis[6] *before* executing the study [20], [21].

Power of a statistical test is dependent on three different components: the probability of error of type I (α), the size of the effect[7] being investigated (γ), and the number of subjects. Using the observed effect sizes reported in [5], we calculated power estimates for this study first with $\alpha = 0.10$. However, the number of subjects required was sufficiently small that it was decided we could afford to reduce α to 0.05. Based on the effect sizes observed in [5], the number of subjects required to obtain a power level of 0.8 or higher, based on repeated measures t-test power curves, was 11 for Und_Corr ($\gamma = 1.48$), 22 for Mod_Comp ($\gamma = 0.61$), and 11 for Mod_Rate ($\gamma = 1.48$). Because Mod_Corr

[5] β being the probability of failing to reject the null hypothesis when the alternative hypothesis is true

[6] Power calculations are 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. Thus, if a significance result is not obtained, a researcher can have sufficient confidence in retaining H_0 (assuming all alternative explanations, which might explain why no result was found, are dismissed).

[7] Effect size may be defined in several ways but a common definition we use is: the difference in the dependent variable means across the two design samples, divided (i.e., standardized) by the geometric mean of the standard deviations of the two samples.

was not measured in the original study we have no basis for an estimate. We therefore make the assumption that the effect size for Mod_Corr is not smaller than 0.61. Therefore, with the anticipation of 33 subjects participating our power estimates seem sufficiently high, even on the basis that they might be inaccurate (which is quite possible; see [21] for details). To provide confidence in our estimated power values, we also calculated the smallest effect size we would have a good chance of detecting significantly, i.e., $1-\beta = 0.8$, with $N = 33$ at $\alpha = 0.05$. In this case, γ was found to be 0.48 which means that our power estimates can only afford to be approximately 21% inaccurate before the power value drops below 0.8 (assuming we can perform a paired t-test with 33 observations). Therefore, a more stringent α level, e.g., 0.01, would not be suitable if we are to retain sufficient statistical power.

In our study, we can afford to preset $\alpha = 0.05$, which favours the scientific perspective above, because the power analysis estimates are sufficient enough to work with a lower α level. To complement the scientific perspective with the practical perspective we shall also provide p-values up to 0.2 (i.e., exact probabilities of committing an error of Type I) — this allows the reader to make their own decisions regarding the trends observed without having to adhere to the scientific approach.

Finally, the qualitative data collected via the debriefing questionnaire will be analyzed to search for qualitative evidence which helps explain (and hopefully further supports) our quantitative results, e.g., can specific difficulties reported by subjects help explain differences in performance across the two designs?

3 Experimental results

Table 4 presents a descriptive summary of the dependent variables for the two different object-oriented designs. The columns represent the number of valid observations (*N*), the mean ($\bar{x}$), the median ($\tilde{m}$), the minimum and maximum values, and the standard deviation (*s*). A quick review of the table shows that all variables directly concerned with the hypotheses are in the predicted direction.

As discussed in Section 2.11, before deciding which tests were appropriate to apply to the data collected, the data had to be checked for deviations from a normal distribution. Appropriate normality tests include the Kolmogorov-Smirnov test and the Shapiro-Wilks' W test. These tests were applied and both indicated that for all dependent variables except Mod Time, the data was substantially non-normal. As a result, non-parametric tests were the appropriate tests to apply - as the data is within-subjects, the most appropriate of these is the Wilcoxon matched pairs test. However, to make sure a parametric test would not have lead to different conclusions we also applied the t-test for dependent samples (which is thought to be robust against minor deviations from normality [25]). In every case, the parametric test supported the findings of the non-parametric one. Before discussing the results of this statistical analysis we examine and discuss anomalies in the data set.

	Good OO Design						***Bad OO Design***					
Variable	*N*	$\bar{x}$	$\tilde{m}$	Min	Max	*s*	*N*	$\bar{x}$	$\tilde{m}$	Min	Max	*s*
Und_Time	22	48.18	50.50	13	69	14.12	19	46.74	45.00	25	75	9.61
Und_Comp	33	5.64	6.00	3	7	1.03	32	4.31	4.00	2	7	1.38
Mod_Time	21	29.71	30.00	9	65	13.41	19	30.21	30.00	10	50	9.74
Mod_Comp	33	0.66	0.77	0	1.00	0.27	32	0.45	0.48	0.10	0.76	0.22
Mod_Corr	31	0.95	1.00	0	1.00	0.18	32	0.93	1.00	0.50	1.00	0.12
Mod_Rate	21	0.65	0.62	0	1.69	0.40	19	0.37	0.27	0.08	1.40	0.31

Table 4 Descriptive statistics for each dependent variable

3.1 Anomalies in the data

Examination of Table 4 shows one or two strange results in the data set — in software engineering experiments, because of the varying degrees in subjects' ability [8], [13], it is to be expected that anomalies in the data set occur. The first anomaly has little to do with ability though — it is the low number of subjects for which there are valid times for Und_Time and Mod_Time, i.e., *N* is far below 33 and 32 for the good and bad designs, respectively. Subjects were told

to record on their collection sheet what time they finished the understanding tasks and moved onto the modification tasks. Many subjects, e.g., 13 subjects for the bad design, unfortunately did not obey this instruction so, although we knew the total time they took to complete the study, we had no idea of the time split between the time spent on understanding and the time spent on the impact analysis. We examine missing data as a threat to validity in Section 4.

The second anomaly is the fact one subject achieved minimum values of zero for Mod_Comp, Mod_Corr, and Mod_Rate for the good design, lower than that for the bad design. The reason for this is the subject only spent a small amount of time (9 minutes) on the impact analysis tasks and during that time did not identify any correct places for modification. Thus, the subject's scores for these variables were all zero. These scores were not removed from the data set to prevent biasing the results in favor of our hypothesis regarding modification. However, we expect this outlier to decrease the visible effect size of good design on maintainability in our experiment. More formally, outlier analysis over all the dependent variables across the two designs show that this particular subject is definitely a strong outlier (see Figure 1). The Mahalanobis distance [17] from the centroid (multivariate mean) of the sample space was computed and this particular subject (Subject P26) was much further away from it than any other observation in the sample; see Figure 1 which displays the jackknife distance which is calculated for each observation with the estimates of mean, standard deviation, and correlation matrix without including the observation itself. We consider this outlier in more detail later in the analysis of the experiment.

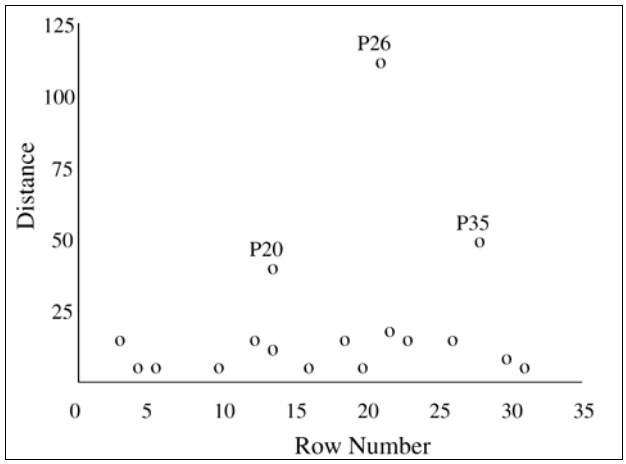

Figure 1 The Mahalanobis Jackknife Distance for outliers

3.2 H_1 — Ease of understanding

Table 5 presents a summary of the results of the statistical tests for the two dependent variables concerned with 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 *Z* value of the Wilcoxon test, column five the critical value for α = 0:05, one-tailed, which *Z* has to exceed to be significant, and column six provides the p value if it is below 0.20.

Variable	γ	Valid *N*	*Z*	Crit. $Z_{0.95}$	*p*-value
Und_Time	0.12	18	1.02	1.64	
Und_Comp	1.09	32	3.46	1.64	0.00

Table 5 Wilcoxon matched pairs test for dependent variables concerning understanding

Table 5 shows that there was no significant difference in the amount of time spent completing the understanding tasks, but there was a significant difference for the variable Und_Corr which measured the extent of a subject's understanding.

We also performed analysis of the qualitative data collected by means of the debriefing questionnaire. First, regarding question 7 (if you could not complete the tasks, indicate why) it was found that 67% (10 subjects from 15) and 74% (17 subjects from 23) of the participants felt they ran out of time when performing the required tasks on the good and bad design, respectively. This is an indication that the time allocated for the experiment was at best sufficient for them to complete their entire task. This "ceiling effect" explains why performance in terms of understanding time was not significantly better for the good design and that the effect of a better design was only reflected in terms of understanding correctness.

Second, regarding question 8 in the debriefing questionnaire (what caused the most understanding difficulty), it was found that when comparing subjects' responses for the good and bad designs 39% (11 subjects from 28) compared to 48% (14 subjects from 29) stated they had problems with either the inheritance, coupling or cohesion the design. In isolation this cannot be regarded as significant evidence, but these numbers do support the direction of our hypotheses and can be considered further support for the results of the quantitative analysis. We therefore accept H_1.

Let us look more closely at the data to identify more precisely what questions are responsible for differences between the good and poor design. Figure 2 and Figure 3 show the distribution of correct answers for each question within each of the two design questionnaires (the outlier P26 is excluded in Figure 3, hence the 31 maximum) . Note that questions are ordered in such a way that similar questions have similar indices in both questionnaires (see Appendix C).

From these histograms, we can see that questions 3, 4, and 5 are mainly responsible for the difference between designs' understanding correctness. On the other hand, questions 1 and 2 are relatively easy questions that are not much affected by the violation of design principles. Questions 3, 4, and 5 concern the relationship between domain requirements and classes, and the services and relationships of classes, respectively. All the classes involved in the questions were affected by the degradations applied to the original bad design (see Section 2.5). Therefore, we see that the differences between the good and bad designs can be qualitatively explained and are mainly due to three questions (out of seven). To a lesser extent, questions 6 and 7 are also affected. Question 6 is more general as it concerns subsystems (or components in OO terminology) and does not require a detailed understanding of classes. Question 7 requires to look at an interaction diagram to understand what message invocations are triggered by a given message invocation and then look at method textual descriptions in the concerned classes. Considering the information processed by the participants, this is not expected to be strongly affected by a lack of conformance to Coad and Yourdon's principles.

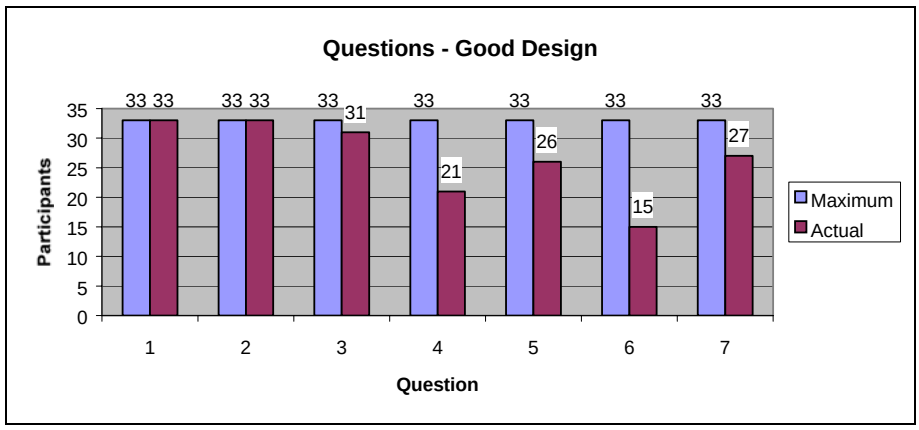

Figure 2 Number of correct answers for the questionnaire of the good design

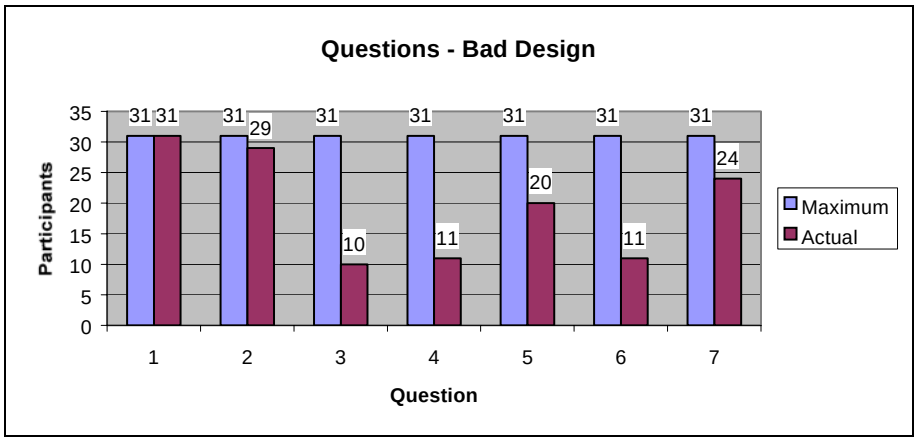

Figure 3 Number of correct answers for the questionnaire of the bad design

If we now look more closely at questions 3, 4, and 5, we see that the classes that are concerned by these questions have undergone degradations with respect to the *coupling* and *Generalization-Specialization* principles, the *cohesion* principle, and the *clarity* principle, respectively (see Appendix B). So we see, based on this qualitative evidence, that all principles (*Simplicity* being related to several of the principles cited) play a significant role in decreasing or increasing the understandability of designs, although *Clarity* appears to be weaker. This, however, would have to be confirmed through replication as several experiments are necessary to cover in enough details all these principles while isolating their effects on maintainability.

3.3 H_2 — Ease of impact analysis

Table 6 presents a summary of the results of the statistical tests for the four dependent variables concerned with H_2.

Variable	γ	Valid *N*	*Z*	Crit. $Z_{0.95}$	*p*-value
Mod_Time	-0.04	17	0.70	1.64	
Mod_Comp	0.84	32	3.42	1.64	0.00
Mod_Corr	0.13	30	1.49	1.64	0.07
Mod Rate	0.61	17	2.11	1.64	0.02

Table 6 Wilcoxon matched pairs test for dependent variables concerning modification

The table shows that there was no significant difference in the amount of time spent completing the modification tasks and their correctness, but there was a significant difference for the variables Mod_Comp and Mod_Rate that measured the completeness of the impact analysis and the modification rate, respectively.

The lack of significance regarding modification time can also be explained, as for hypothesis H1, by tight time constraints during the experiment. In addition, when removing the outlier discussed above from our analysis, the difference in modification correctness becomes significant (p-value = 0.03). Even though we use a non-parametric test, which is not as sensitive to outliers as the t-test, the Wilcoxon matched pair test is sensitive to the magnitude of differences in ranks, for each individual, between the bad and good designs. This outlier represents the poorest performance for the good design and therefore shows, for the reasons explained above, a large difference in rank in a direction opposite to the one expected. This explains the impact of this outlier on our results and since it can be explained, it is reasonable not to consider this observation in our results.

Analysis of the data collection for question 9 (what caused the most difficulty performing the impact analyses) found similar differences exposed by the analysis above for the data from question 8. 44% (12 subjects from 27[8]) compared to 56% (15 subjects from 27) indicated difficulties caused by either inheritance, coupling, and cohesion on the good and bad designs, respectively. Again, on its own, this is not significant evidence, but it provides additional support for both the direction of the hypotheses and the results provided by the statistical tests. Therefore, we accept H_2 .

However, like for hypothesis H1, we want to have a closer look at the data and see if we get additional support for the results presented above. Figure 4 and Figure 5 show, for the good and bad designs, the proportion and number of participants who found each change place during impact analysis (the outlier P26 is excluded in Figure 5, hence the 31 maximum). The change places indexed from 1 to 14 are due to a change in customer requirements. Change places indexed from 15 to 21/22 are due to the enhancement of the system's functionality. The results show clearly that the difference in found change places between the good and bad designs are not due to any particular small subset of places. To the contrary, a general tendency is clearly visible. A Kruskal-Wallis test (non-parametric, rank-based, independent samples) comparing the proportions of participants who found change places in the two designs yields a p-value < 0.000 (two-tailed), thus confirming what is graphically visible. The difference in proportions is therefore statistically significant between the two designs. This further supports the plausibility of H_2 since we can see that the difference between good and bad designs is not due to any small number of change places but represents a strong, general trend.

[8] Although every participant was asked to fill out the debriefing questionnaire only 27 from 33 did so.

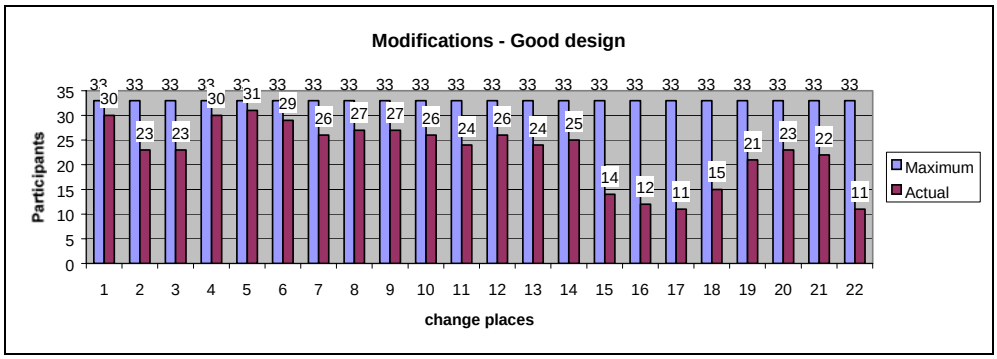

Figure 4 Number of participants who found change places for the good design

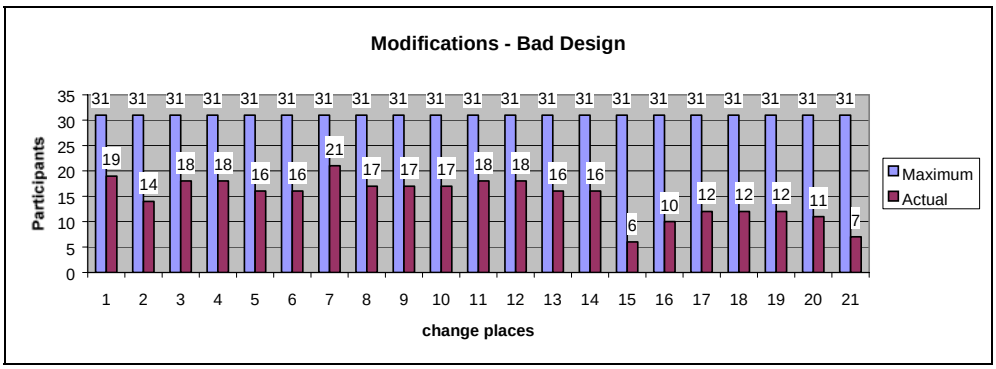

Figure 5 Number of participants who found change places for the bad design

Like for questions in the previous section, it is also interesting to look at the class degradations that are associated with the change places in the bad system. Table 7 provides a list of the principles that have been transgressed for the classes corresponding to each change place. It can be seen that transgressions to each of Coad and Yourdon's principles seem to affect some changes, either in isolation or in combination with other principles. More precisely, for each principle, the numbers of transgression instances are: Clarity (10), Coupling (10), Cohesion (5), Simplicity (6), Generalization-Specialization (4). For example, change 1 shows to be affected by Clarity only and shows a low participant proportion of 60% (19/31). On average, similar proportions are obtained for other changes (5, 7, 11, 18) where Clarity shows in isolation. Changes 3 and 4 also show similar participant proportions for coupling and cohesion, respectively. Because of interdependencies, other design principles do not show up in isolation but are usually associated with low participant proportions.

Therefore, it seems that, based on qualitative evidence, all Coad & Yourdon's principles seem to play a significant role during the impact analysis of designs.

Ch. Place	Description	Ch. Place	Description
1	Clarity	11	Clarity
2	Coupling, Gen-Spec	12	Clarity, Cohesion
3	Coupling	13	Clarity, Gen-Spec
4	Cohesion	14	Clarity, Simplicity, Coupling
5	Clarity	15	Coupling, Gen-Spec, Clarity
6	Simplicity, Coupling	16	Coupling, Gen-Spec
7	Clarity	17	Cohesion, Simplicity
8	Simplicity, Clarity, Coupling	18	Clarity
9	Simplicity, Coupling	19	Coupling, Simplicity
10	Simplicity, Cohesion	20	Coupling, Simplicity
		21	Cohesion

Table 7 Principles transgressed for each change place

3.4 Analysis summary

Overall the results show that the good design, i.e., complying with Coad and Yourdon's design principles, is significantly easier to understand and modify. This is further supported by results obtained from analyzing the qualitative data from the debriefing questionnaire. This is particularly true with respect to the correctness and completeness of the responses provided by the experiment participants. Because of tight time constraints during the experiment (i.e., ceiling effect), differences in understanding and modification times could not have been observed to a significant level. For future replications, it might be advisable that more time be allocated or, alternatively, that the tasks performed be shorter (although this would not be as good as the first option). In such conditions, the results might show shorter completion times for the good design but similar correctness and completeness results. On the other hand, it might be argued that this is would not be as interesting a finding given that it is not unusual for professional developers to have to work under tight time constraints when performing maintenance activities.

Additionally, we have tried to look at the relationships between completeness and correctness measures and time allocated to the task by the participants. No significant relationships could be identified for both the good and bad designs. This further supports our assumption that there is a large variation in participant skills and our preference for a within-subject design.

Given that the overall practical significance of Coad and Yourdon's design principles has been established through a significant decrease in participant proportions for the bad design, a series of independent experiments is now necessary to investigate each and every design principle independently. However, because of interdependencies between principles and their natural tendency to interact

within design documents and tasks, specific experimental materials and tasks will have to be devised to make such investigations possible. The main practical difficulty lies in the fact that a number of constraints will have to be addressed together when designing the experimental material and tasks: realistic questions and changes, a functional "bad design", realistic degradations, and independent effects of principles. Only the latter condition is was not fulfilled in the current experiment.

4 Threats to validity

This section discusses the study's various threats to validity and the way we attempted to alleviate them.

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. Furthermore, because it is more careful to use several different measures to capture a concept, we defined 2 and 4 measures for the concepts of understandability and impact analysis, respectively. Since our measures consistently support the stated hypotheses, as does the collected qualitative data, we can have confidence in their construct validity.

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. However, our approach has been based both on work in the literature on object-oriented design and measurement work on object-oriented systems which supports it, e.g., [2], [13]. 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:

1.) Missing time data occurred as a result of a subset of subjects not following experimental instructions. The concern here is that this data loss was not random and thus had an influence on the results. However, comparing the mean of the two times variables, Und_Time and Mod_Time shows

there were no significant differences between designs (as one would expect given the time constraints of the study). This is reassuring in that non-random differences would be expected to affect these variables somewhat. However, we cannot be sure whether the data loss was non-random so we can only state that we are unsure about the seriousness of this threat.

2.) An instrumentation effect may result from differences in the experimental materials employed. The threat to this study was that possible differences between the two software designs and the tasks to be performed were causing any performance differences. Regarding the latter threat, a serious attempt was made to ensure the tasks were very similar (i.e., similar questions and changes in terms of type and numbers). In addition, the authors pre-tested that they required similar amounts of time to complete. The former threat, i.e., difference in design cognitive complexity, is perhaps the most complex and important one. Our strategy was to select designs which, as discussed in Section 2.5, had similar metric values for typical counts of artifacts and standard object-oriented design measures. It should be noted that t is extremely difficult, if at all possible, to find two distinct functional designs which perfectly match with respect to all metrics one may possibly think of using. This problem also stems from the current state of the art in OO measurement, where it is difficult to determine what are the measurement dimensions relevant to cognitive complexity. In addition, it is interesting to note that analysis of the debriefing questionnaire data found no significant difference of opinion on the appropriateness of the size of the two design documents ($\rho = 0.29$) nor was any significance difference of opinion found on the overall difficulty of the tasks which accompanied them ($\rho = 0.39$).

3.) Similarly, a confounding effect between the design principles and the problem domains used would mean that subjects' performance was affected by both these variables. This might occur because one problem domain is more familiar to the experiment participants or is more suited to an object-oriented solution, 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 help counter this threat, the problem domains we used for OO were taken from examples in textbooks detailing OO design methodologies; as such, these domains were deemed to be amenable to an object-oriented design. In addition, based on the study curriculum of the participants, there was no reason to believe they would be more familiar with one application domain than the other. However, it is clear that to address the threat fully would require that only one problem domain be used and that functionally equivalent good and bad OO designs be generated from it. But this latter alternative has problems of its own which justifies why we did not adopt it. Using the same original design for both the good and bad designs

would imply a one run experiment (to avoid learning effects). This in turn would mean that we would have to halve the number of participants in each group (i.e., the good and bad designs) and the consequence on the statistical power of our testing would be substantial. In addition, by using different participants in the two groups, it becomes more difficult to control for human factors that we know have a tremendous effect on results. This stems from the fact that no paired statistical testing can be applied and participants do not act as their own control anymore. We know from experience [14] that random assignment helps but does not ensure comparable groups, especially in the case of small groups. In addition, blocking is not easy to perform in software engineering since we do not know very well which criteria to use to form blocks.

4.) 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.

4.3 External validity

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

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

2. The subjects who participated in this study are unlikely to be representative of software professionals. This is not to say that the results cannot be useful in an industrial context for several reasons. Laboratory settings such as this one allow the investigation of a larger number of hypotheses at a lower cost than field studies. The hypotheses that seem to be supported in the laboratory setting can then be tested further in more realistic industrial settings with a better chance of discovering important and interesting findings. Conversely, laboratory experiments can be used to confirm results obtained in field studies, where control and therefore internal validity is usually weaker.

It is 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., [26], [19].

4.4 Addressing threats to validity

In this section we provide guidance for improving our experiment and to address some of the threats to validity identified above.

Increase the task time. The time allocated for answering questions and performing impact analysis should be increased because many subjects stated the reason they did not complete all the tasks was due to time constraints. Alternatively, the experimental tasks should be made less complex. The result of this improvement may mean the experiment finds similar values for the dependent variables concerned with completeness and accuracy but shorter completion times and hence increased modification rates for the good design.

Improve time data collection procedures. The data collection procedures for collecting precise time data should be improved upon due to our experience with subjects not fully obeying experimental instructions. One method of achieving this manually would be to give subject only the understanding questionnaire to begin with. Once a subject has completed the questionnaire they indicate this to a monitor who records their time and then provides them with the modification tasks. A second option would be to automate the whole experimental procedure thereby making time collection trivial.

Develop different systems with same functionality. To fully address the threat of instrumentation requires two designs developed from the same requirements documents, one obeying the design principles of Coad and Yourdon, the other not. However, as discussed above, a within-subjects design would not be appropriate because of learning effects, so lack of power is likely to be a problem, except when a large number of participants is accessible for the experimenters. In addition, since controlling for human factors is more difficult in a between-subject design, much more care would have to be dedicated to the formation of homogeneous blocks.

4.5 Within- versus Between- Subject Designs

Our experiment illustrates a common but important problem when designing experiments with human subjects in software engineering. Due to practical constraints, we are often in a situation where the number of potential participants is small. We are therefore tempted to use all participants on all treatments to achieve satisfactory statistical power. This is called a within-subject design. Another, maybe more important, reason to do so is to control for the very strong variations in human capabilities that are commonplace in software

engineering. As we are dealing with small groups, random assignment or even blocking still entails substantial risk with respect to the comparability of groups.

On the other hand, the main problem with within-subject designs are learning effects. We cannot allow, for example, the same material to be used twice, or more, by the same subjects as their performance would be likely to grow from one experimental run to the other. This is why in this experiment we had to resort to two different designs. In such situations, we then have to make sure that the materials are comparable to prevent instrumentation threats to validity. As we rarely know, in software engineering, what constitute important differences between artifacts (e.g., requirements, designs, code), it is usually difficult to demonstrate, in an objective manner, the comparability of distinct experimental materials.

In such experiments, ultimately, one has to make a trade-off between which threat to validity is least likely to impact the experimental results: group differences in capability, combined with possible low statistical power, or material differences in cognitive complexity. This trade-off has to take into account the level of understanding of the artifact at hand and the number and backgrounds of participants.

5 Conclusions

The results of our experiment show that the system designed according to Coad and Yourdon's object-oriented design principles was significantly easier to maintain, i.e., complying with Coad and Yourdon's object-oriented design principles has led to a design which is easier to understand and perform impact analysis upon. This result is further supported by qualitative evidence provided by the experimental subjects in the form of information from their debriefing questionnaires. This is consistent with a previous experiment [5] where we tested the same hypothesis, although the experimental material was improved for this experiment. This provides us with a high degree of confidence in the finding that the maintainability of object-oriented designs are sensitive to poor design practices, as defined by Coad and Yourdon. In addition, it is important to note that such experiments allowed us not only to confirm but also to quantify the effects of (not) using such design principles.

From a more general perspective, if these results were further confirmed through external replication and complementary studies, organizations involved in object-oriented development should invest in the definition of design standards to ensure that their software products comply, to the maximum extent possible, to quality principles such as those provided by Coad and Yourdon. As a consequence, their maintenance costs would be likely to decrease significantly.

Last, this paper also provides general insights on the use of within-subject designs in software engineering experiments. Such designs typically entail instrumentation threats to validity but are necessary when using small but potentially heterogeneous groups of participants. In the current state of the art, we are therefore limited by our rough understanding of the important measurements to consider if we want to ensure the comparability of experimental materials such as requirements or design descriptions. The improvement of software engineering experiments and product measurement therefore goes hand in hand. Some of our related work on the topic of object-oriented measurement can be found in [3][6][27][28][29].

A replication package [9] is available for researchers interested in externally replicating our experiment. Improvements to the experimental procedure might include increasing the task time and improving the time data collection procedures.

Acknowledgments

The authors 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. Thanks are also extended to Dieter Rombach for motivating the students to participate, to Christiane Differding for help dealing with time consuming administrative issues, and to Michael Ochs for calculating the values of the various measures used for the two designs and for help ensuring the experiment ran as planned. We are also grateful to Stephan Thiel for his precious help with the editing of this document. Finally, we wish to thank the anonymous reviewers of this paper whose contributions helped to significantly improve this final version.

References

All ISERN technical reports are available from
http://www.iese.fhg.de/ISERN/pub/isern_biblio_tech.html.

[1] R. Arnold. Impact analysis - towards a framework for comparison. In Proceedings of the IEEE Conference on Software Maintenance, pages 234-243, 1993.

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

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

[4] L. Briand, C. Bunse, and J. Daly. An experimental evaluation of quality guidelines on the maintainability of object-oriented design guidelines. In *Proceedings of Empirical Studies of Programmers*, October 1997.

[5] L. Briand, C. Bunse, J. Daly, and C. Differding. An experimental comparison of the maintainability of object-oriented and structured design documents. *Empirical Software Engineering, An International Journal*, 2(3), 1997.

[6] L. Briand, J. Daly, and J. Wuest. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering, An International Journal*, 3(1), 1998.

[7] Brooks, J. Daly, J. Miller, M. Roper, and M. Wood. Replication of experimental results in software engineering. Technical Report ISERN-96-10, Department of Computer Science, University of Strathclyde, Glasgow, 1996.

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

[9] L. Briand, C. Bunse and J. Daly. *Replication package for Design Experiment SE I WS 96-97*. Fraunhofer Institute (IESE), Kaiserslautern, Germany, 1997.

[10] S. Chidamber and C. Kemerer. A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering*, 20(6):476-493, June 1994.

[11] P. Coad and E. Yourdon. *Object-Oriented Analysis*. Prentice-Hall, second edition, 1991.

[12] P. Coad and E. Yourdon. *Object-Oriented Design*. Prentice-Hall, first edition, 1991.

[13] Curtis. Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68(9):1144-1157, September 1980.

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

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

[16] H. Deubler and M. Koestler. Introducing object-orientation in large and complex systems. *IEEE Transactions on Software Engineering*, 20(11):840-848, November 1994.

[17] Everitt. *Cluster Analysis*. Edward Arnold, third edition, 1993.

[18] Jones. Gaps in the object-oriented paradigm. *IEEE Computer*, 27(6):90-91, June 1994.

[19] B. Kaplan and D. Duchon. Combining qualitative and quantitative methods in information systems research: A case study. MIS Quarterly, pages 571-586, December 1988.

[20] H. Kraemer and S. Thiemann. *How many subjects?* Sage Publications, first edition, 1987.

[21] J. Miller, J. Daly, M. Wood, A. Brooks, and M. Roper. Statistical power and its subcomponents - Missing and misunderstood concepts in empirical software engineering research. *Information and Software Technology*, 39, pp. 285-295, 1997.

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

[23] P. Spector. *Research Designs*. Quantitative Applications in the Social Sciences. Sage Publications, 1981.

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

[25] J. Welkowitz, R. Ewen, and J. Cohen. *Introductory Statistics for the Behavioral Sciences*. Academic Press, second edition, 1976.

[26] M. Wood, J. Daly, J. Miller, and M. Roper. *Multi-Method Research: An Empirical Investigation of Object-Oriented Technology*. To appear in the Journal of Systems and Software, 1999

[27] L. Briand, J. Wuest, J. Daly, V. Porter. *Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems*. To appear in the Journal of Systems and Software, 2000

[28] L. Briand, J. Daly, J. Wüst, J., 1999. *A Unified Framework for Coupling Measurement in Object-Oriented Systems*. IEEE Transactions on Software Engineering 25 (1), 91-121

[29] L. Briand, J. Wüst, S. Ikonomovski, H. Lounis. *A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study*. Technical Report ISERN-98-29, Fraunhofer Institute for Experimental Software Engineering, Germany. A short version is to be published in the proceedings of ICSE'99.

A Debriefing questionnaire (Translated from German)

The information you provide in this questionnaire may be very valuable to us. Please answer each question as honestly as you can. Anything you write down will be treated confidentially. Thank you.

Personal details and experience

Id. Number (e.g., A1): Qualifications (e.g., Informatik Vordiplom):

Please answer the following **four** questions based on this experience scale:

None	Little	Average	Substantial	Professional
1	2	3	4	5

Base your answers upon what you knew **before** attending the Software Engineering I course.

1. What is your experience with software engineering practice? (circle number).

   1 2 3 4 5

2. What is your experience with design documents in general? (circle number).

   1 2 3 4 5

3. What is your experience with object-oriented design documents? (circle number).

   1 2 3 4 5

4. What is your experience performing impact analyses? (circle number).

   1 2 3 4 5

Motivation and performance

1. Estimate how motivated you were to perform well in the study.

Not	Poorly	Fairly	Well	Highly
1	2	3	4	5

   Please explain your answer:

2. Estimate how well you understood what was required of you.

Not	Poorly	Fairly	Well	Highly
1	2	3	4	5

3. What approach did you adopt to the exercise? (tick only one).

   (a) Read the documents fully and then attempt the tasks.
   (b) Read the documents while thinking about the tasks.
   (c) Straight into the tasks, reading the documents as required.
   (d) Other — please specify:

4. Estimate the correctness (in %) of your answers to the understanding questionnaire.

0 – 20	21 – 40	41 – 60	61 – 80	81 – 100
1	2	3	4	5

5. Estimate the correctness (in %) of your answers to the impact analyses.

0 – 20	21 – 40	41 – 60	61 – 80	81 – 100
1	2	3	4	5

6. Estimate the accuracy (in %) of your answers to the impact analyses.

0 – 20	21 – 40	41 – 60	61 – 80	81 – 100
1	2	3	4	5

7. If you could not complete all the tasks, please indicate why. (tick only one).

   (a) Ran out of time.

   (b) Did not fully understand the task.

   (c) Did not fully understand the design documents.

   (d) Other — please specify:

8. In your opinion, what caused you the most difficulty to understand the design documents? (tick only one).

   (a) Nothing in particular.

   (b) The notation used.

   (c) Inheritance.

   (d) Cohesion in classes, i.e., the relationships within a class.

   (e) Coupling between classes, i.e., the relationships between classes.

   (f) (f) Other — please specify:

9. In your opinion, what caused you the most difficulty to perform impact analyses on the design documents? (tick only one).

   (a) Nothing in particular.

   (b) The notation used.

   (c) Inheritance.

   (d) Cohesion in classes, i.e., the relationships within a class.

   (e) Coupling between classes, i.e., the relationships between classes.

   (f) Other — please specify:

Miscellaneous

1. How do you judge the size of the design documents you had? (please circle).

Too small	Small	About right	Large	Too large
1	2	3	4	5

2. On a scale of **1 to 10** estimate, in terms of **understandability**, the quality of the design documents you had. (1 - barely understandable; 10 - easily understandable).

   Please specify number:

3. On a scale of **1 to 10** estimate your overall understanding of the design documents. (1 – very little; 10 - complete).

   Please specify number:

4. What did you understand least about the design documents and why?

   Please specify:

5. On a scale of **1 to 10** estimate, in terms of modifiability, the quality of the design documents you had. (1 - barely modifiable; 10 - easily modifiable).

   Please specify number:

6. On a scale of **1 to 10** estimate the overall difficulty of the tasks you have been asked to perform. (1 - very easy; 10 - very difficult).

   Please specify number:

7. Having performed the tasks, would you do anything different next time around?

   Please specify:

8. Have you learned anything from participating in this study?

   Please specify:

9. Any additional comments?

Thanks once again.

B Degradations Applied to the "Original bad" Design

This experiment compares a design that is not conformant to Coad and Yourdon principles with a design which is fully conformant. The former was obtained by performing degradations to an original design that was conformant. The changes involved in this degradation and according to each of the five guidelines are:

- **Coupling**: Interaction coupling was increased by introducing additional associations between classes, resulting in a higher number of messages a class has to handle. Additionally messages became more complex by defining additional responsibilities for a class, or by introducing new classes to communicate with. Furthermore additional, inappropriate, inheritance (e.g., by introducing new abstract classes to decrease inheritance coupling) was introduced to decrease inheritance coupling.
- **Cohesion**: Two classes were merged into one. Attributes and/or operations that were not necessary for the responsibility of a class were introduced in various classes.
- **Clarity of design**: Vocabulary was changed (but not the information content) to be somewhat inconsistent, by using acronyms or synonyms in different parts of the design for class, method, and attribute names. Furthermore, in some cases, class, attribute, or method names were used which did not correspond to the concept being modeled (this has also an effect on cohesion).
- **Generalization-Specialization depth**: Additional abstract and inherited classes were introduced by artificially subdividing some classes across two inheritance levels.
- **Keeping objects and classes simple**: Several unused but meaningful attributes were added to various classes, resulting in a class where several additional attributes could be potentially used by a single service Additionally, class definitions were made fuzzier (i.e., the class did not map to an entity in the problem description) by merging classes into one.

C Generic Descriptions and Examples of Questions

Understandability is measured by asking the subjects to complete a questionnaire with seven questions regarding the overall understanding, structure, and functionality of both designs. These questions were intended, within manageable limits, to capture the diversity of questions that a maintainer may ask about a design. Although the designs differ, the questions for both are comparable on a generic level (e.g., question 4 is aimed at the services a specific class offers). In the following we describe, at a generic level, each question in more detail.

Question 1. Describe the overall functionality of the system. This question is intended as a starting point for the subjects to get comfortable with the setting and the design.
Example: Please give a brief description of the overall system functionality.

Question 2. Describe the externally visible behavior of the system, e.g., in the case of the ATM systems, examples of externally visible operations are dispensing cash or printing checks.
Example: Please name the externally visible system operations of the ATM system and give a short description.

Question 3. Map domain requirements to classes (i.e., the subjects have to identify the classes which model a specific functionality of the overall system). To answer that question subjects have to understand the system functionality and the functionality of single classes before they can make the mapping. For the 'bad' design, we expect that inconsistencies in naming, merging of classes, new inheritance, etc. have a negative effect on that question.
Example: For each externally visible operation, please name the class(es) which model them,

Question 4. Describe the services of a specific class. For this question the subjects have to understand the functionality of a specific class in terms operations, return values, etc. For the 'bad' design, this question is directed towards a class that is the result of merging two different classes of the original system. Consequently, the functionality of that specific class is not that as easy to understand because there is no close logical relationships between the functionality of the two original classes.
Example: Please, describe the services provided by class 'Entry Station'.

Question 5. Describe the relationships (i.e., associations, aggregations) of a specific class. To answer this question, subjects have to understand the interactions between classes. Introducing new, unnecessary, relationships and increasing the depth of inheritance (i.e., changes done to the 'bad' design) make this question more difficult to answer.
Example: Please, describe the dependencies of class 'bank' to other classes of the system.

Question 6. Name the component that handles a specific system functionality. For this question subjects have to map system functionality onto the architecture of the system.
Example: Please name the component(s) of the ATM system which handle transaction-security related services.

Question 7. Describe the functionality implemented by a series of messages triggered by a given message invocation. This question requires the understanding of the systems architecture, components, classes, and their interaction.
Example: Please give a short description of how temperature information is processed.

D Examples for Impact Analysis Tasks

Change of System Functionality

The requirements for rain detection in the HEISS system and all accompanying functionality was removed (the house is now to be build in the Kalahari)

- Please mark all places within the system requirements which have to be changed.
- Please mark all places within the system design which have to be changed.

Adding new System Functionality

The HEISS system should be capable to print the current state of the house system in terms of temperatures, settings, and so on, on user request.

- Please mark all places within the system design which have to be changed.

.

Document Information

Title:	A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs
Date:	May 1999
Report:	IESE-002.99/E
Status:	Final
Distribution:	Public