https://doi.org/10.48009/2_iis_2009_148-156

AN EMPIRICAL METHOD FOR MEASURING THE EFFECTIVENESS OF AN OBJECT-ORIENTED DESIGN PROCESS

Stevan Mrdalj, Eastern Michigan University, smrdalj@emich.edu
Vladan Jovanovic, Georgia Southern University, vladan@georgiasouthern.edu
Joseph Scazzero, Eastern Michigan University, jscazzero@emich.edu

ABSTRACT

The effectiveness of a design process can be critical to the success of information system development projects. There have been very few empirical studies on the effectiveness of design processes that use unified modeling language (UML). This paper establishes a model for measuring the effectiveness of diagram derivation as a way of measuring the effectiveness of the object-oriented design (OOD) process. It utilizes the defect density approach to evaluate an OOD process in terms of how correct the results are that it produces, how consistent those results are and how scalable the process is. To demonstrate how applicable the proposed methodology is, we conducted an empirical evaluation to determine the effectiveness of the user interface driven design process. This study calculated the defect densities for four UML diagrams and measured the effectiveness of the design process on three levels: individual type of defect, type of diagram and the entire project. The results show that the proposed methodology can be reliably used to evaluate an OOD process.

Keywords: measuring effectiveness, defect density, object-oriented design process, UML.

INTRODUCTION

Information systems departments are under enormous pressure to cut costs while maintaining productivity in software development projects. Improving software productivity has become a critical issue for all organizations. However, according to the Standish Group CHAOS report [18], project failure rates are as high as 65% (n=13,522 projects) with an estimated 51% average coverage of required features and functions. Information system design can be critical to the success of the entire project. If an information system has a flawed design, it is likely to have other quality problems. Conversely, an effective design is likely to enhance the quality of the system's other parts. Therefore, one way to improve the productivity of software development is to improve the design process.

The Object Management Group standardized the unified modeling language (UML) in late 1997 [3]. UML has become an industry standard for the presentation of various artifacts in the object-oriented design (OOD) process. UML is a method-independent modeling language. Its method-independence and widespread use has created an opportunity for establishing formal OOD processes. Since the standardization of UML, many OOD processes have emerged among which the Object-oriented Process, Environment and Notation (OPEN) and the Rational Unified Process (RUP) are best known [10,11]. The existence of several OOD processes poses the question of how to compare the effectiveness of these processes.

This paper proposes a method for the empirical evaluation of the effectiveness of an OOD process that uses UML. The aim of this method is two-fold. First, it can serve as a foundation for comparing and selecting an OOD process. Second, it can be used to improve an OOD process by detecting its weak points. The proposed method evaluates the effectiveness of an OOD process by determining how:

- correctly it produces UML diagrams,
- consistent it is with respect to the application area and development team experience, and
- scalable it is with respect to the size of a project.

In order to measure the effectiveness of creating UML diagrams, we established a model for measuring diagram derivation effectiveness (MDDE). It defines a set of defects for each type of diagram that could violate a well-specified system design [5, 7] and uses defect density [8] as a measurement tool. Defect density, which measures the correctness of derived diagrams, is an indicator of the effectiveness of an OOD process. We recognize that the correctness of derived diagrams can only provide a partial answer for effective Object-Oriented Analysis and Design (OOAD). In order to evaluate how consistent the results are with respect to the application area, development team experience and size of a project, we propose statistical tests of hypotheses on defect densities for the entire project, for each type of diagram and for each type of defect within a diagram.

To illustrate the usage and applicability of the proposed measurement method, we conducted an in-depth evaluation of diagrams which were produced using the user interface driven design (UIDD) approach [12]. The study analyzed projects developed by senior-level graduate students taking a required systems analysis and design class over a period of seven semesters.

RELATED RESEARCH

Early studies related to object-oriented methodologies focused mainly on a comparison to traditional methodologies that highlight their relative strengths and weaknesses. The second wave of studies focused on comparisons of different object-oriented diagramming notations. The standardization of the Unified Modeling Language (UML) eliminated the disputes over using different variations of diagrams for the same purpose. There are several empirical studies directed towards the evaluation of UML notational elements [16] and the comprehension of UML diagrams [14].

Widespread acceptance of` UML has also created an opportunity to evaluate the effectiveness of the different design processes used to create UML diagrams. There were studies on qualitative comparisons of OOD processes [10], but only a few studies that used quantitative comparisons. Cao [4] reports an empirical study related to the performance of an OOD process by measuring the productivity of a UML-based OOAD. It is based on the UML complexity metrics developed by Rossi and Brinkkemper [15] and the related calculations developed by Siau and Cao [17]. Another study developed an empirical model of project performance [21] in terms of both project completion and budget. Yet another study investigated the nature of the process of information system development. Gouilelmos [9] reports the findings of an empirical study that examines the ineffectiveness of information system development in terms of the approach adapted rather than the methodology used.

There have been only two studies to date that employ empirical measurement of OOD processes. The first one, done by Yacoub, Ahmmar & Robinson [20], addresses the problem of measuring the quality of OOD. It proposes a measurement that evaluates coupling metrics for an OOD expressed in UML. The limitation of this method is that it examines only the design-coupling structure in discovering errors. The second study, done by Tsagias & Kitchenham [19], employed defect rates to evaluate the use of business objects and business components. The use of defect rates in this study was limited, however, to a comparison between the development from scratch and the development using business components.

While these studies have made a significant contribution to the management of OOD processes, none of them empirically measure the OOD process in terms of how correct the results are that it produces for all types of diagrams, how consistent those results are and how scalable the process is.

A MODEL FOR MEASURING DIAGRAM DERIVATION EFFECTIVENESS

In general, researchers attempt to relate the size, structure, quality, and reliability of software to its complexity, cost, adoption, or success. They frequently used defect density as the common measure of software quality [8]. Consequently, the effectiveness of an OOD process can be measured by the number of defects made in deriving the diagrams. Thus, defect density (DD) will be used as a measure of diagram derivation quality where:

$$\mathrm{DD} = \frac{number\ of\ known\ defect}{number\ of\ use\ cases}$$

Since this study purports to measure the effectiveness of the UML modeling process, the traditional software metrics such as function points (FP) and lines of code (LOC) do not apply here [1]. Therefore, in our study we used the number of use cases to estimate the project size instead of a traditionally used measure such as FP and LOC. The use of the number of use cases as a measure of the project size can be argued from a perspective that use cases may widely vary in their complexity but they do reflect needed functionality and most importantly, they are available early in the design process.

The issue of the use cases' complexity can be at least partially addressed by the granularity of use cases. It is intuitively obvious that the abstraction mechanism and the divide and conquer strategies make the average use case complexity somewhat the same across systems of varied complexity. The usual desire to handle very complex systems without overwhelming details in each use case effectively minimizes the use case's inherent complexity to the level of comfort for appropriately trained (competent) humans. Therefore, we believe that the overall system size can be indeed estimated by the number of use cases.

Types of Defects

The empirically found defects can be used to determine the apparent correctness of each type of diagram. Table 1 lists common types of defects that can be found in class, sequence, communication, and statechart diagrams. Similar empirically observable defects can be defined for all other UML diagrams.

A diagram is considered correct when the above defects are not found. The correct design model does not assume that there is one best way to model the system. In this paper, we are not interested in comparing functionally "equivalent" models but rather the effectiveness of deriving diagrams using an OOD process.

Table 1. Types of Defects for Different Types of Diagrams

	Type of Diagram			
	Class	***Sequence***	***Communication***	***Statechart***
Type of Defect	*Missing a class*	*Missing an object*	*Missing an object*	*Missing a state*
	Missing an attribute	*Missing an operation*	*Missing a link*	*Missing a transition*
	Missing an association	*Missing a control structure*	*Missing a message*	*Missing a start/end*
	Needless class	*Needless object*	*Needless object*	*Needless state*
	Needless attribute	*Incorrect object name*	*Incorrect object name*	*Needless start/end state*
	Incorrect multiplicities	*Incorrect type of the control structure*	*Wrong order of messages*	*Wrong superstate name*
	Wrong class name	*Incorrect order of the control structure*	*Needless link*	*Wrong state*
	Wrong association name	*Needless message*	*Incorrect message*	*Incorrect superstate*
	Needless associations	*Wrong message*		*Needless transition*
	Incorrect aggregation			*Wrong transition name*
	Incorrect inheritance			*Missing superstate*
	Incorrect attribute name			
	Missing inheritance			
	Missing aggregation			

Evaluation Instruments

Scoring tables were developed for each of the diagrams to tabulate defects. Figure 1 shows the layout of the scoring table for class diagrams. The scoring tables for other diagrams have a similar format with columns named for the concepts associated with that type of diagram presented in Table 1.

Measuring Diagram Correctness

We suggest measuring how correctly an OOD process derives UML diagrams by at least the following three types of defect densities:

- the defect density for the entire project;
- the defect density for a particular type of diagram; and
- the defect density for a certain type of defect within a diagram.

A measure of overall project level effectiveness is the defect density for the entire project (DD_P) which can be defined as the total number of defects found in all diagrams divided by the number of use cases in the project. Equivalently, the project defect density is equal to the sum of defect densities for all UML diagram used in a project.

$$DD_P = \frac{total\ number\ of\ known\ defect}{number\ of\ use\ cases}$$

Another important indication of the effectiveness of an OOAD is how correctly it produces individual diagrams. The correctness of the diagrams is demonstrated by the defect density for each type of diagram (DD_D) where

$$DD_D = \frac{number\ of\ defects\ for\ certain\ diagram}{number\ of\ use\ cases}$$

A third indication of the effectiveness of an OOD process is the defect density for the individual types of defects (DD_C) within each diagram where

$$DD_C = \frac{number\ of\ defects\ for\ individual\ type\ of\ defect}{number\ of\ use\ cases}$$

Class				Attribute				Association				Multiplicity		Inheritance			Aggregation/ Composition		
Count	Wrong name	Missing	Needless	Number	Wrong name	Missing	Needless	Number	Wrong name	Missing	Needless	Number	Incorrect	Number	Missing	Incorrect	Number	Missing	Incorrect
2				3			1	2				4		1					
4		1		8				11				22					1		
4				9				10				20	1	1		1			
2				7		1		1				2							

Figure 1. Scoring Table for Class Diagram

Measuring Consistency of an OOAD

It is important to determine if an OOD process consistently produces the defect densities discussed above regardless of application area or implementation environment. An OOD process can be considered consistent for a project, diagram or defect if the average defect density for that project, diagram or defect was the same over time for projects with different application areas or implementation environments. Thus, in order to determine consistency of an OOAD, the following null hypotheses can be tested:

Hypothesis 1a: The average defect density for entire project (DD_P) is the same across projects.

Hypothesis 1b: The average defect density for a certain type of diagram (DD_D) is the same across projects.

Hypothesis 1c: The average defect density for an individual defect in the given diagram (DD_C) is the same across projects.

We propose using the one-way analysis of variance (ANOVA) to test for the equality of these population average defect densities across projects.

Measuring Scalability of an OOAD

Large projects are likely to have a higher complexity, resulting in significantly increased number of diagrams that need to be developed. We also wanted to determine if an OOD process is scalable with respect to project size. An OOD process can be considered scalable if no relationship exists between defect densities for that project, diagram or defect and the size of project measured by the number of use cases in the project. In other words, scalability implies that the defect density for a project, diagram or defect does not change as the size of the project increases. We hypothesized that an increase in project size will not be associated with an increase in defect density and tested the following null hypotheses.

Hypothesis 2a: No relationship exists between the defect density for the entire project and the number of use cases.

Hypothesis 2b: No relationship exists between the defect density for a given type of diagram and the number of use cases.

Hypothesis 2c: No relationship exists between the defect density for a given type of defect and the number of use cases.

We recommend that scatter plots be constructed to determine the type of relationship between the above defect densities and the number of use cases. For example, if a linear relationship is present, the sample Pearson product moment correlation coefficient r, which measures the strength of the linear relationship between two variables, can be used to test the above hypotheses.

APPLYING THE MDDE

The usage and applicability of the MDDE is demonstrated by applying it to measure the effectiveness of the User Interface Driven System Design (UIDD) process [13]. This study analyzed 43 projects developed by a total of 211 senior-level graduate students enrolled in a Master's of Science in Information Systems program. The projects were collected over a period of seven semesters. Participants self-divided into teams that had an average size of 4.9 students. Each team had to find a local business for which they had to design an information system. There were no multiple projects for the same company. No particular order was used to assign subjects to teams and teams randomly selected their project application area. The duration of the project was the entire semester. All teams followed the UIDD process and the guidelines on how to develop diagrams [13]. All teams used the same CASE tool with minor differences due to different versions.

Data Analysis and Findings

Descriptive statistics were obtained for these projects on their characteristics and their defect densities. The projects averaged 11.5 use cases, 6.0 actors, 16.3 interactions, 1.3 includes, 1.1 extends, and 0.4 inheritances. The number of use cases per project showed an almost uniform distribution, ranging from 6 to 19 use cases per project. The complete analysis and findings can be found in [13].

The projects had a total of 557 class diagrams, resulting in an average of 13.0 diagrams per project. The class diagrams had an average of 8.8 classes and 13.6 associations. There were a total of 483 sequence diagrams, averaging 11.2 diagrams per project. Each sequence diagram had on average 4.1 objects and 17.3 operations/messages. We examined a total of 108 communication diagrams, an average of 2.5 diagrams per project. The smaller number of communication diagrams relative to the number of sequence diagrams is a result of the subjects' selection preference among comparable tools. The communication diagrams averaged 4.7 objects, 4.8 links, and 7.0 operations/messages. Subjects developed a total of 68 statechart diagrams, an average of 1.6 diagrams per project, where each statechart diagram averaged 7.1 states and 11.2 transitions/events.

Measuring Correctness for the UIDD

The overall correctness of diagrams derived using UIDD for entire project is presented using a boxplot as illustrated in Figure 2. Boxplots are used to show the distributional characteristics of DD_P. The line in the box is drawn at the median while the bottom of the box is at the first quartile (25th percentile) and the top of the box is at the third quartile (75th percentile). Points outside the lines from the box are considered outliers and are indicated by asterisks.

It is extremely difficult to assess the significance of the obtained results since the authors are not aware of any similar study and therefore are unable to perform real comparisons. For illustration purposes, we used the quantitative targets for managing US defense projects [6], according to which, an effective design method should produce a defect density that is less than 4 whereas an ineffective method produces a defect density that is greater than 7. Based on such assumptions, Figure 2 shows that all DD_P are well within the effective level except two outlier projects that were outside the effective range, but still below the ineffective level. A 95% confidence interval showed that the average project defect density is between 1.56 and 2.29. The commonly used confidence level of 95% represents the percentage of times that the confidence interval will contain the parameter of interest, in this case the average project defect density, under repeated sampling.

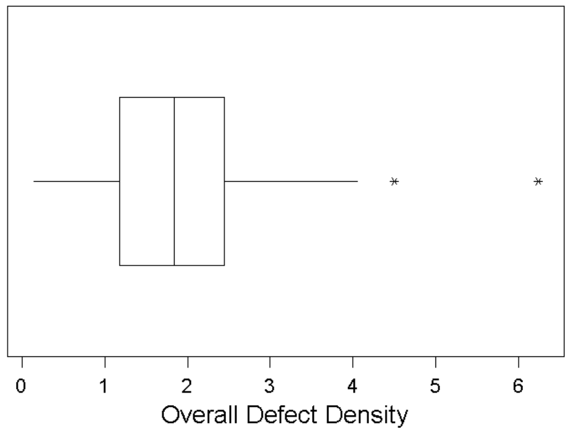

Figure 2. Boxplot of DD_P values

The correctness of the individual diagrams produced using the UIDD is demonstrated by the defect densities for each type of diagram (DD_D) as shown by the boxplots in Figure 3. Again, all diagrams are well within the effective level even though class diagram had one outlier project that was outside the effective range but still below the ineffective level.

Another way to assess the effectiveness of an OOD process is to consider confidence intervals for the average defect density (DD_D) for each type of diagram. In this illustration, the 95% confidence interval for the average DD_D for class diagrams is between 0.84 and 1.43, for sequence diagrams between 0.30 and 0.70, for communication diagrams between 0.12 and 0.29, and for statechart diagrams between 0.04 and 0.13.

The third indication of how an OOD process correctly the produces the UML diagrams is the low defect densities for the individual types of defects (DD_C) within each diagram. In the case of UIDD, the average DD_C found in class diagrams are given in the Table 2. The types of defects are ordered in decreasing order of their average defect densities.

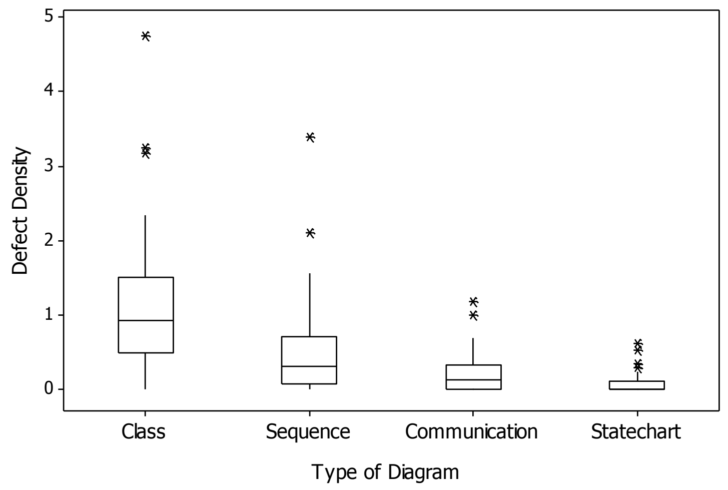

Figure 3. Boxplots of Defect Density Values by Type of Diagram.

Similarly, the average DD_C could be calculated for all other diagrams.

Table 2. Average Defect Density by Type of Defect in Class Diagrams

Type of Defect	Average DD_C
Incorrect multiplicities	*0.528*
Needless associations	*0.122*
Missing an association	*0.114*
Wrong class name	*0.078*
Wrong association name	*0.066*
Needless class	*0.060*
Missing a class	*0.056*
Needless attribute	*0.037*
Missing an attribute	*0.021*
Incorrect inheritance	*0.021*
Incorrect aggregation	*0.014*
Incorrect attribute name	*0.003*
Missing aggregation	*0.003*
Missing inheritance	*0.002*

The very low average defect densities for all types of defects in the class diagrams seem to support the effectiveness of the UIDD approach. These defect densities certainly compare favorably to the average grades, ranging from 1.7 to 2.2 on the scale of 0 (lowest) to 4 (highest), obtained through reverse engineering for 42 case studies involving database design and modeling [2]. It should also be noted that these low defect densities were achieved for projects that varied widely with respect to application area, system types, and size.

Measuring Consistency for the UIDD

Next, we wanted to determine if the UIDD consistently produced the low defect densities shown above regardless of application area or implementation environment. Hypothesis 1a was supported since the average project defect density was the same across all projects (F(6,36) = 1.19, p-value = 0.334).

Hypotheses 1b was formulated for each type of diagram, class, sequence, communication, and statechart. Based on the ANOVA results for class diagrams shown in Table 3, we do not reject Hypothesis 1b at α = 0.01 and conclude that the UIDD produces consistent results for class diagrams. Similarly, we also conclude that the UIDD produces consistent results for sequence, communication, and statechart diagrams.

Table 3. ANOVA Results for Testing the Equality of the Average DD_D across Semesters

Diagram	*F(6,36)*	*p-value*
Class Diagram	*1.04*	*0.419*
Sequence Diagram	*1.43*	*0.230*
Communication Diagram	*2.83*	*0.023*
Statechart Diagram	*2.89*	*0.021*

To determine if the UIDD is consistent with respect to the types of defects associated with each type of diagram, we developed Hypothesis 1c for each type of defect in all diagrams. Table 4 shows the ANOVA results for testing the equality of the average DD_C across semesters by type of defect in class diagrams shown in Table 2. For example, for the first type of defect in Table 4, we did not reject the hypothesis that the average defect density of missing classes in class diagrams is the same across semesters at α = 0.01 since F = 0.69 with a p-value of 0.657.

Table 4. ANOVA Results for Testing the Equality of the Average DD_C across Semesters by Type of Defect in Class Diagrams

Type of Defect	*F(6, 36)*	*p-value*
Missing a class	*0.69*	*0.657*
Missing an attribute	*2.30*	*0.056*
Missing an association	*0.29*	*0.939*
Needless class	*0.92*	*0.492*
Needless attribute	*1.84*	*0.118*
Incorrect multiplicities	*2.91*	*0.020*
Wrong class name	*1.46*	*0.219*
Wrong association name	*0.67*	*0.677*
Needless associations	*1.49*	*0.211*
Incorrect aggregation	*0.60*	*0.729*
Incorrect inheritance	*0.77*	*0.601*
Incorrect attribute name	*0.39*	*0.880*
Missing inheritance	*0.59*	*0.734*
Missing aggregation	*1.03*	*0.420*

Thus, we can conclude that the UIDD consistently produces low average defect density with respect to missing classes in class diagrams. Similarly, consistency was found to exist for the other types of defects in class diagrams listed in Table 4. This implies that the UIDD produces consistent average DD_C for all types of defects in class diagrams. However, an in-depth examination of the data showed that for $\alpha = 0.05$, there is a significant difference across semesters for the incorrect derivation of multiplicities. This indicates that a more comprehensive study is needed to improve the process for deriving multiplicities in the UIDD.

Table 5 shows the ANOVA results for testing the equality of the average DD_C across projects for all types of defects in sequence diagrams. For $\alpha = 0.01$, we can conclude that consistency exists for each type of defect in sequence diagrams except for the deriving of needless messages. The significant difference between averages for this defect was primarily due to a higher average of "needless messages" in the early use of UIDD. This was corrected by improving the derivation procedure in subsequent projects. Thus, except for this one type of defect, the average number of defects does not change across semesters for sequence diagrams. Similar tests at $\alpha = 0.01$ showed that the UIDD was also consistent for each type of defect in communication and statechart diagrams.

Based on the facts that all hypotheses except one for one defect type were supported, we can conclude that the UIDD was consistent for all projects with respect to application area and implementation environment.

Table 5. ANOVA Results for Testing the Equality of the Average DD_C across Semesters by Type of Defects in Sequence Diagrams

Type of Defect	*F (6, 36)*	*p-value*
Missing an object	*1.21*	*0.325*
Missing an operation	*2.57*	*0.036*
Missing a control structure	*1.21*	*0.322*
Needless object	*1.92*	*0.105*
Incorrect object name	*0.97*	*0.461*
Incorrect type of the control structure	*2.79*	*0.025*
Incorrect order of the control structure	*2.23*	*0.038*
Needless message	*4.58*	*0.001**
Wrong message	*1.27*	*0.294*

**Significant at $\alpha = 0.01$*

Measuring scalability of the UIDD with respect to the size of the project

As mentioned previously, the size of the projects in this study ranged from 6 to 19 use cases per project. To determine if the UIDD is scalable for the given range of project sizes, we tested Hypothesis 2a using the sample Pearson product moment correlation coefficient r. This hypothesis was supported since the project defect density did not change as the size of the project increased (r = -0.13, p-value = 0.400). It should be noted that only tests for linear relationships were necessary since an examination of the scatter plots showed that only linear relationships were present between the defect density for entire project (DD_P) and the number of use cases.

Consequently, we developed Hypothesis 2b for each type of diagram to determine if the UIDD is scalable for those diagrams. For example, we did not reject the hypothesis that no linear relationship exists between defect density for class diagrams and the number of use cases at $\alpha = 0.01$, Table 6. Thus, we can conclude that the UIDD is scalable for class diagrams, i.e., the defect density for class diagrams does not change as the size of the project increases. Similarly, we also found that the defect density for sequence, communication, and statechart diagrams does not change as the size of the project increases.

Table 6. ANOVA Results for Testing the Relationship between DD_D and Number of Use Cases

Type of Diagram	*r*	*p-value*
Class Diagram	*-0.36*	*0.018*
Sequence Diagram	*0.17*	*0.275*
Communication Diagram	*0.23*	*0.140*
Statechart Diagram	*0.054*	*0.732*

Similar tests of hypothesis showed that the UIDD was invariant for each type of defect found in these diagrams, i.e., no linear relationship was found between the defect density for each type of defect, DD_C, and the number of use cases. Thus, we can conclude that the diagram derivation using the UIDD is scalable with respect to tested project sizes.

CONCLUSIONS AND LIMITATIONS

This paper contributes to the limited empirical research in measuring the effectiveness of the object-oriented design processes by introducing a model for measuring diagram derivation effectiveness. This model uses a defect density (number of known defects per use case) approach to measure how effective is an OOD process in deriving UML diagrams. The level of defect densities for individual defects, individual diagrams and entire projects indicate how effective is an OOD process in deriving UML diagrams. Based on statistical tests of hypothesis, we can assess how consistently an OOD process produces defect densities on all levels (project, diagram and individual defect) with respect to the application area and implementation environment. Lastly, based on correlation tests of hypothesis, we also can conclude how an OOD process scales with respect to the tested project sizes for all three levels.

One of the strengths of the MDDE is its wide applicability to various types of OOD processes and its ease of use for all UML diagrams. Results from applying MDDE to the UIDD indicate that MDDE can be viable not only to compare several OOD processes, but also to detect inefficiencies of an OOD process and to provide insightful information and guidance in terms on how to improve an OOD process.

The proposed method for measuring effectiveness of an OOD process would be incomplete without an examination of the limitations of its research method. In calculating defect densities, we used the defects found in the design phase of the SDLC versus the defects found in an operational system. Since the absence of defects in the system design does not guarantee correct system operation, the operational defect densities will certainly differ from our results. This method also did not measure inter-rater reliability, since there were very few disagreements with respect to defects during the pilot evaluation. We acknowledge that one possible area of disagreement might be in the categorization of the defects. Since the categorization of defects is essential to the MDDE, this type of disagreement should be addressed as a part of the adaptation process.

Another commonly perceived validity threat to the MDDE is use of students in its application to the UIDD. The use of students in this empirical study may only limit the generalization of the conclusions regarding the UIDD. In our case, it is irrelevant who performed the design because our focus was not on the effectiveness of the UIDD. Instead, our goal was to demonstrate the strength and usability of the MDDE and, therefore, the use of students should not undermine the conclusions of this study.

REFERENCES

1. Basili, V. R., Briand, L.C., Melo, W. L. (1996) A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*, **22**, 751 -761.
2. Blaha, M. (2004) A Copper Bullet for Software Quality Improvement. *IEEE Computer*, **37**, 21-25.
3. Booch, G. Rumbaugh, J. and Jacobson, I. (1998) *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.
4. Cao, Q. (2004) Assessing Productivity of UML-based Systems Analysis and Design: A DEA Approach. In: *Proceedings of the Tenth Americas Conference on Information Systems*, New York, NY, 1652-1660.
5. Chidamber, S.R., and Kemerer, C.F. (1994) A Metrics Suite for Object-Oriented Design. *IEEE Transaction on Software Engineering*, **20**, 476-493.
6. Department of the Navy (US), (1995) Software Program Managers Network. *NetFocus*, No. 207.
7. Dromney, R.G. (1996) Cornering the Chimera. *IEEE Software*, **13**, 33-43.
8. Fenton, N.F. and Pfleeger, S.L. (1997) *Software Metrics: A Rigorous & Practical Approach*, 2nd ed. PSW, London.
9. Goulielmos, M. (2004) Systems development approach: transcending methodology. *Information Systems Journal*, **14**, 363-386.
10. Henderson-Sellers, B., Collins, G., Due, R. & Graham, I. (2001) A qualitative comparison of two processes for object-oriented software development. *Information and Software Technology*, **43**, 705.
11. Kruchten, P. (2004) *The Rational Unified Process: An Introduction 3/e*. Reading, MA, Addison Wesley Longman.
12. Mrdalj, S. and Jovanovic, V. (2002) User Interface Driven System Design. *Issues in Information Systems,* **3**, 441-774.
13. Mrdalj**,** S.**,** Scazzero, J. and Jovanovic, V. (2004) Effectiveness of the User Interface

Driven System Design Using UML. *Computer Science and Information Systems*, **1**(2), 153-172.

14. Otero, M.C. & Dolado, J.J. (2004) Evaluation of the Comprehension of the Dynamic Modeling in UML, *Information and Software Technology*, **46**, 35.
15. Rossi, M. and Brinkkemper, S., (1996) Complexity Metrics for Systems Development Methods and Techniques. *Information Systems*, **21**, 209-227.
16. Shen, Z. & Keng, S. (2003) An Empirical Evaluation of UML Notational Elements Using a Concept Mapping Approach. In: *Proceedings of the Twenty-eight International Conference on Information Systems*, Seattle, Washington, pp. 194-206.
17. Siau, K. & Cao, Q. (2001) Unified Modeling Language (UML) - a complexity analysis. *Journal of Database Management*, **12,** 26-34.
18. Standish Group CHAOS Report (2003) http://www.standishgroup.com/ sample_research/index.php
19. Tsagias, M. & Kitchenham, B. (2000) An Evaluation of the Business Object Approach to Software Development. *The Journal of Systems and Software*, **52**, 149.
20. Yacoub, S.M., Ammar, H.H. & Robinson, T. (2000) A Matrix-Based Approach to Measure Coupling in Object-Oriented Design. *Journal of Object-Oriented Programming*, **13(7)**, 8-20.
21. Yetton, P., Martin, A., Sharma, R. & Johnston, K. (2000) A model of information systems development project performance. *Information Systems Journal*, **10**, 263-289.