A Comparative Case Study of Three Database Application Development Environments

Iftikhar Ahmed and Frederick H. Lochovsky

Department of Computer Science
University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

Abstract

This paper describes an exploratory case study which investigates the claim that the object-oriented approach to information systems development reduces development effort and improves maintainability. Software metrics for development effort, complexity and maintainability of three functionally similar 4GL database development environments, one objected-oriented and two traditional (relational), are compared. The results indicate that programs developed using the object-oriented environment needed fewer lines of code, had less volume, required less effort, were less complex and were more maintainable then those developed using the traditional 4GL environments.

Keywords: object-oriented environments, 4GLs, software metrics

1. Background of the Study

Some of the claims put forth by object-oriented development methodologies are that they cut development time [2], are more maintainable [2, 4], are easier to enhance [14], are more understandable [2], and have greater clarity of expression [11] than traditional fourth generation language (4GL) approaches. To investigate these claims, an object-oriented 4GL development environment, namely, OASIS [8, 9], and two relational 4GL development environments, namely, Oracle [13] and Access [12], were compared. These three development environments, while functionally similar are, of course, not identical in every respect. However, the goal of the case study was to compare, at a macro-level, the object-oriented and traditional 4GL approaches to software development and not to compare specific facilities or functionalities of object-oriented and relational 4GLs or database systems, *per se*. In doing so it is, of course, difficult to unequivocally isolate precisely which aspects of each environment contributed to the overall results.

The case study explored the differences between the two approaches from an application developer's point of view. Thus, our interest is in comparing the development of and the actual application code resulting from the use of each of the three development environments. Since procedural languages are used for application development in all three environments, traditional software metrics can be used to evaluate and compare the two approaches. Accordingly, the following software metrics were measured and compared (words in parenthesis indicate how these metrics are referred to in the rest of this paper): lines of code (*LOC*) in a program [3]; Halstead's Software Science (*Volume, Difficulty, Effort*), which is based on a count of operators and operands in a program [5]; McCabe's Cyclomatic Complexity (*CC*), which is a count of the number of independent logical paths through a program [10]; and Henry and Kafura's Information Flow Metric (*INFO*), which is a measure of the information flows into and out of a program [6]. The first five metrics are code metrics; the sixth is a program structure metric. In addition, since 4GL development environments are being compared, their user interfaces, from an application development viewpoint, should also be compared. A user interface metric, called *quant*, which is the quantity of data input through the user interface between two confirmations (e.g., Enter or Return keys being pressed), was used for this purpose [7].

Data for measuring and comparing these metrics were collected from the implementation and subsequent maintenance of an application in all three development environments. The application deals with the Postgraduate Administration System of the Computer Science Department in our University [1]. Requirements for the application were specified using the structured approach. The same structured design was used for developing application programs in all three environments. As this study was exploratory in nature and the resources to conduct it were limited, only one programmer was used to implement the application in all three environments. Since all three environments were new for the programmer, to minimize the learning curve effect, before starting to develop the real application programs a test application was implemented in all three environments. To control any bias introduced by the order of program development in the three environments, the application was divided into six modules which were implemented in the order indicated in Table 1, which also shows the size of each module, in lines of code (LOC), for each development environment.

	Implementation Order					
	First	LOC	Second	LOC	Third	LOC
Module A—Admission Process	OASIS	30	Oracle	101	Access	32
Module B—TA Assignment	Access	207	OASIS	153	Oracle	149
Module C—Student/Advisor Matching	Oracle	130	Access	148	OASIS	32
Module D—Course Registration	OASIS	69	Oracle	139	Access	187
Module E—Qualifying Exam	Access	36	OASIS	18	Oracle	30
Module F—Degree Requirement Checking	Oracle	3	Access	0	OASIS	12

Table 1: Implementation order among development environments and size of modules—development phase.

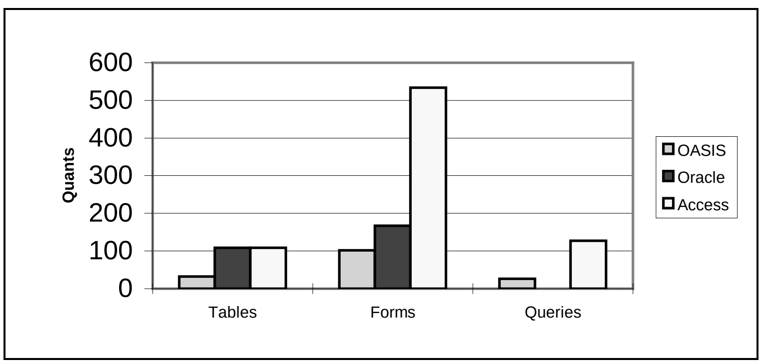
Section 2, presents the quantitative comparison of the software metrics for the initial application development in OASIS, Oracle, and Access. Section 3 discusses the quantitative comparison of the maintenance phase. Section 4 explores the statistical significance of our results. Finally, Section 5 summarizes and discusses the findings.

2. Software Development Phase

2. 1. Quantitative Analysis of the User Interface Metric

The comparison of quants for table design (Graph 1), shows that Oracle and Access, need the same number of quants while OASIS needs fewer since tables are automatically generated from forms in OASIS. For form design OASIS also needs fewer quants than Oracle and Access. For queries, Oracle needs no quants as it automatically generates the options for a query. OASIS requires only the selection of a form and the QUERY procedure, while Access requires the user to input SQL statements or use a Query-by-Example interface.

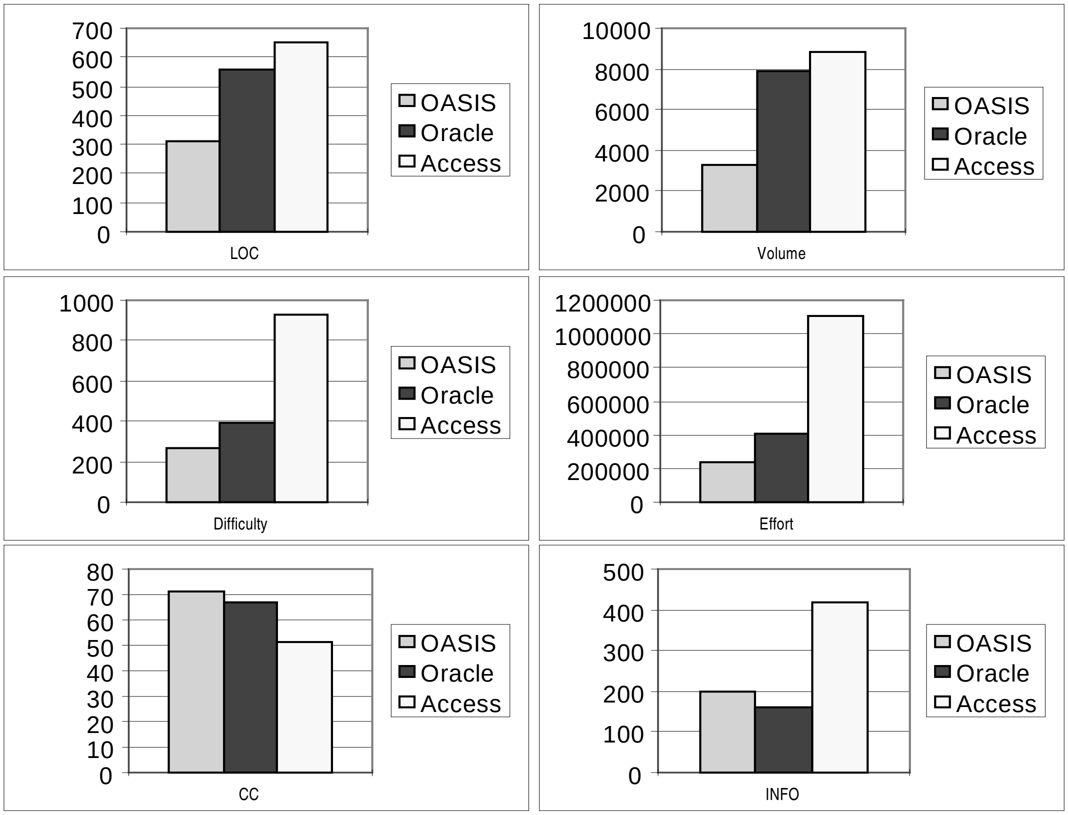
2. 2. Quantitative Analysis of Software Metrics



Graph 1: Comparison of quants for creating tables/forms/queries—development phase.

Graph 2[1] shows that, for the whole system development, OASIS took the fewest lines of code (LOC), had the least volume, was the least difficult, and required the least effort. However, its cyclomatic complexity (CC) was slightly higher than Oracle and Access. OASIS has more intermodule communication (INFO) than Oracle but less than Access. Access has a higher value than Oracle and OASIS for all the metrics except for cyclomatic complexity.

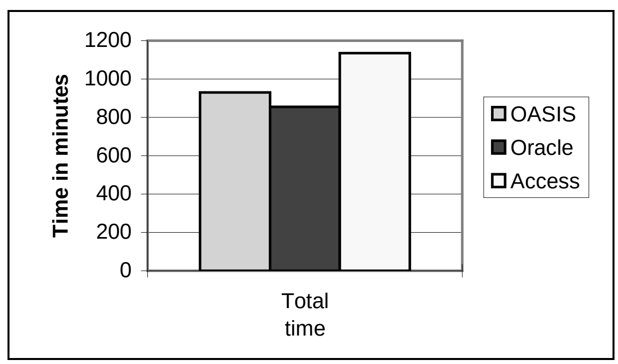
[1] The smaller the value, the better the measure for each metric.



Graph 2: Comparison of modules measured for overall system—development phase.

2. 3. Software Development Time

Log sheets were used to record the start and finish time for implementing each module. Graph 3 compares the overall development time of the modules among OASIS, Oracle, and Access.



Graph 3: Comparison of software development time.

3. Software Maintenance Phase

After completing the initial software development, some design changes were made to Modules D, E, and F and then those changes were coded in all three environments.

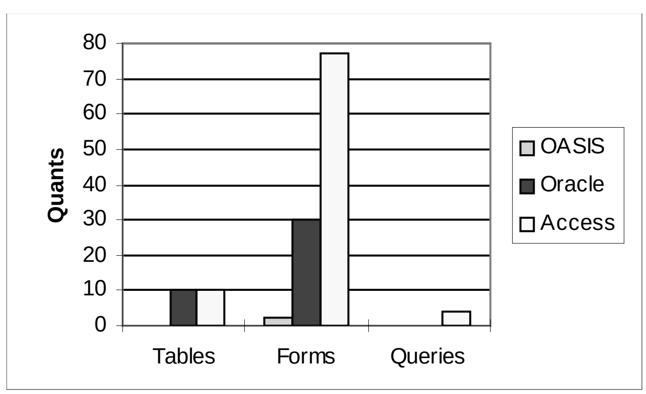
Module D processes the course registration for students. The input to the module is courses that a student wishes to take. In the initial version of the system, once the student requests course registration for some courses, the academic advisor and the PG Director examine the courses and approve or disapprove them. The change made in the design is to verify whether the student meets the pre-requisite(s) of the course or already has passed the course. Changes were also made in the input to the module. In the initial system, each student enters one record for each course. In the new system, instead of one record for each course in which a student wishes to register, one record is used for all the courses, which eliminates the need to query the course records.

Module E handles the PhD qualifying exam request made by a student. In the initial system, the qualifying exam request is mailed to the PG Director who manually checks whether the student qualifies to take the exam or not. In the new system, changes are made to automatically verify whether a student qualifies to take the qualifying exam. Changes are made in the module to check the number of attempts made for the qualifying exam and the result of those attempts.

Module F checks the course requirements of a student. In the initial system, the PG Director queries the courses taken by a student, and waivers of the course requirements given to a student, and, if a Ph.D. student, the result of his qualifying exam. All these queries are made one by one and the information checked. After querying all records, the PG Director enters a record for the student to indicate whether he is graduated or expected to graduate in the current semester, or can not graduate due to not meeting the course requirements. In the new system, changes are made so that the PG Director can query and display all this information together. This requires the system to query records from different tables and display the information on the screen.

3. 1. Quantitative Analysis of the User Interface Metric

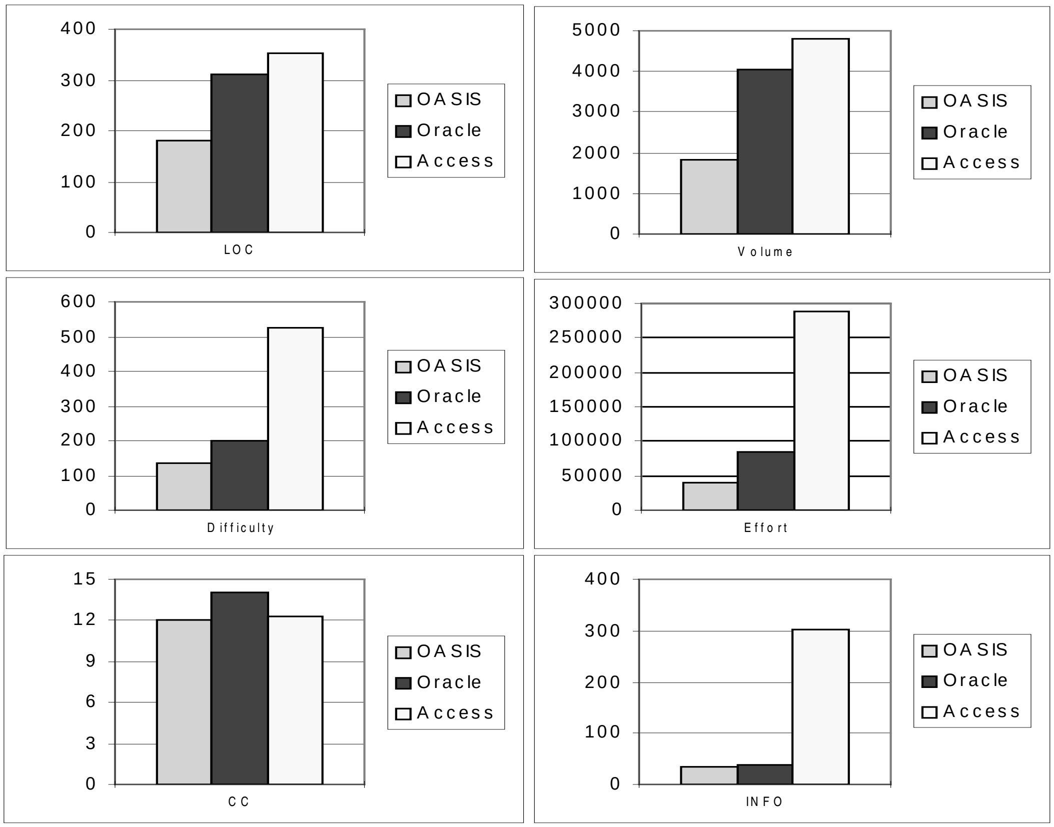
Graph 4 shows that the same number of quants need to be used in order to amend/add tables in Oracle and Access. OASIS does not require any changes to its tables as changes made in the forms are automatically reflected in the tables. In Oracle and Access it is necessary to make significant changes to the forms while in OASIS only a small number of changes are needed. In Access it is necessary to develop a query using the Query Tool for



Graph 4: Comparison of quants for table/forms/queries—maintenance phase.

retrieving information. However, in OASIS and Oracle this can be done with the help of non-procedural query statements using the programming language.

3. 2. Quantitative Analysis of Software Metrics



Graph 5: Comparison of metrics measured for modules D, E, and F—maintenance phase.

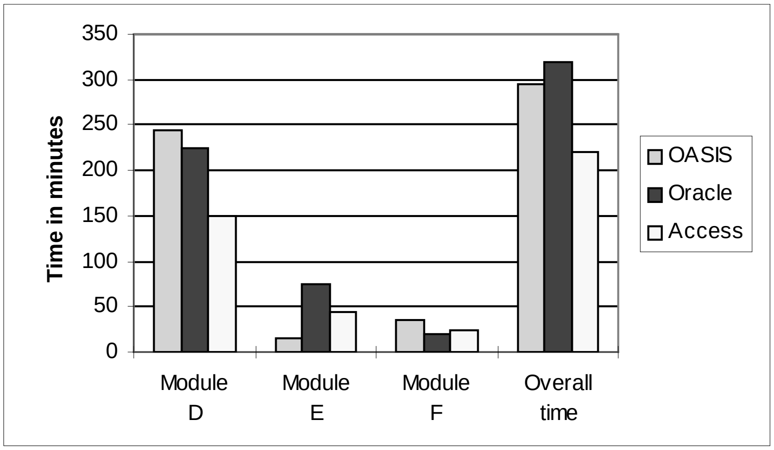
Graph 5 shows the comparison of the metrics measured for modules written in all three languages. For the entire system maintenance, OASIS has the lowest values for all metrics since, due to its object-oriented nature, it requires fewer changes to be made to accommodate the new requirements.

3. 3. Software Maintenance Time

The maintenance task grouping and module size are shown in Table 2 and the software maintenance time by module and overall in Graph 6.

	Implementation Order					
	First	LOC	Second	LOC	Third	LOC
Module D—Course Registration	OASIS	121	Oracle	239	Access	279
Module E—Qualifying Exam	Oracle	66	Access	69	OASIS	40
Module F—Degree Requirement Checking	Access	5	OASIS	19	Oracle	6

Table 2: Implementation order among development environments and size of modules --maintenance phase.



Graph 6: Comparison of software maintenance time.

4. Statistical Significance of the Data

Table 3 shows whether there are significant differences (at the 0.05 level) between the software metrics of the three environments and which environment had the better measure. The means of the combined metrics for the development and maintenance phases are compared in Table 3. It is assumed that the measurements between environments are independent, that the data are normally distributed and that they are in an interval scale. A two-tailed t-test was used to determine whether there was a significant difference between environments (since our H_1 is of the form $\mu_1=\mu_2$). A one-tailed t-test was used to determine which environment had the better measure (since our H_1 is of the form $\mu_1> \mu_2$). The table shows that OASIS, Oracle and Access have a significant statistical difference for the LOC, Volume, Difficulty, and Effort metrics with OASIS having a better measure than either Oracle or Access and Oracle having a better measure than Access. There is no significant statistical difference between OASIS, Oracle, and Access for the CC metric. A significant statistical difference is also shown between OASIS and Access for the INFO metric, with OASIS having the better measure, while OASIS and Oracle have no significant statistical difference in their results for this metric. This confirms our quantitative results

	OASIS versus Oracle				**OASIS versus Access**				**Oracle versus Access**			
Metric	*t*	*p*	Diff?	Better?	*t*	*p*	Diff?	Better?	*t*	*p*	Diff?	Better?
LOC	4.28	0.00	**Yes**	**OASIS**	7.45	0.00	**Yes**	**OASIS**	3.42	0.00	**Yes**	**Oracle**
Volume	4.68	0.00	**Yes**	**OASIS**	7.96	0.00	**Yes**	**OASIS**	2.91	0.00	**Yes**	**Oracle**
Difficulty	2.36	0.02	**Yes**	**OASIS**	7.19	0.00	**Yes**	**OASIS**	6.29	0.00	**Yes**	**Oracle**
Effort	2.98	0.00	**Yes**	**OASIS**	5.72	0.00	**Yes**	**OASIS**	4.80	0.00	**Yes**	**Oracle**
CC	0.42	0.67	No		0.60	0.54	No		0.21	0.83	No	
INFO	-0.53	0.59	No		2.23	0.03	**Yes**	**OASIS**	2.74	0.00	**Yes**	**Oracle**

Table 3: Differences between software metrics measured and development environment having the better measure.

which have shown that for all the code metrics, except cyclomatic complexity, the measures were indeed different. For the structure metric, Oracle does not show a big difference in the results as compared to OASIS, but Access shows a big difference in the results as compared to OASIS and Oracle.

To determine whether there is any correlation among the metrics, Pearson's correlation coefficient between pairs of metrics was computed (Table 4). As expected, all the code metrics correlate very highly indicating that they measure similar aspects of the code, namely internal complexity. The INFO metric does not correlate well with the code metrics, again as expected, indicating that it is measuring a different aspect of software complexity, namely, intermodule communication.

Metric	LOC	Volume	Difficulty	Effort	CC	INFO
LOC	1					
Volume	0.98	1				
Difficulty	0.92	0.96	1			
Effort	0.87	0.92	0.98	1		
CC	0.84	0.88	0.94	0.95	1	
INFO	0.43	0.40	0.39	0.36	0.54	1

Table 4: Inter-metric correlation for software metrics.

To investigate whether development time was dependent on the order of module implementation the F-test was used. Since the value of F is within the critical region for all the tests [1], the null hypothesis, that the development time does not depend on the order of implementation, is not rejected.

5. Summary and Discussion

The goal of this study was to compare, using software and user interface metrics, an object-oriented 4GL development environment (OASIS) with more traditional (relational) 4GL development environments (Oracle and Access) to investigate the claim that object-oriented approaches are better for software development and maintenance. For the initial system development, the quantitative analysis of the code metrics has shown that generally OASIS did have better values for these metrics (except for cyclomatic complexity) than did either Oracle or Access. For the maintenance phase, the quantitative analysis has shown that, generally, the system developed in OASIS is easier to maintain than those developed in Oracle or Access.

Given the above results, one might expect that the development and maintenance time for OASIS should be less than that for Oracle or Access. However, the observed results do not bear this out. There may be several explanations for this outcome. First, there may not be a direct correlation between development/maintenance time and the software and user interface metrics. The nature of the facilities provided in each environment (e.g., debugging tools) and directness and ease of accomplishing a task may greatly affect the time recorded, but not be reflected in the software metrics which measure only the final code produced. Second, the application was designed using the structured approach. This may be more easily translated into traditional 4GLs than into object-oriented 4GLs. Finally, while the programmer had no previous training in using an object-oriented 4GL, he did have significant previous training in traditional 4GLs. Thus, the longer time needed to develop OASIS programs may have been due to the time taken to understand and apply the object-oriented programming paradigm. Further studies to explore these hypotheses would be useful.

As indicated in Section 1, our goal was not to compare specific facilities of each development environment. Such a comparison is beyond the scope of this case study. Nevertheless, it is possible to elaborate on some of the facilities of OASIS which may have contributed to the quantitative results.

- Compared to Oracle and Access, OASIS has better support for rules, which makes it easier for a user to develop rule-based applications. Many 4GL applications are rule-based, as was the application used in this case study. Hence good rule development facilities can definitely ease the application development and maintenance effort.
- OASIS has an easier way to communicate between computers and people than does Oracle or Access. The commands and mechanism to send and receive electronic messages are very simple and straightforward. Again, many applications now require support for electronic-based communication and good support for such facilities can reduce application development effort and time.
- The development facilities in OASIS make client/server applications easy to build. In contrast to the standard static arrangements of Oracle and Access, OASIS supports dynamic situations where an application program can be both client and server depending on the particular task to be performed. Such flexibility, which results from the object-oriented nature of OASIS, can greatly ease the application development and maintenance effort.
- The object-oriented nature of OASIS provides encapsulation, which means that a program and its contained implementation of knowledge can be changed/moved/enhanced without affecting other programs that depend on it. This greatly eases the maintenance effort.
- Creating and changing a form is very simple in OASIS. OASIS forms provide inheritance from a super class form making it easier to create new forms without having to again define already existing objects. This object-oriented nature of OASIS forms simplifies their development and maintenance.

As is apparent from the above discussion, several of the facilities of OASIS, though by no means all, that ease development and maintenance effort derive from its object-oriented nature. Thus, we believe that the object-oriented nature of OASIS was a significant factor in the results reported in this paper.

This study has some obvious limitations. First, only one programmer was involved in the study. To further corroborate the results, a larger study with more programmers is needed. Second, only software and user interface metrics were compared. As the mixed results on development and maintenance time indicate, there may be many other factors that determine the effectiveness of a software development environment and these should also be studied and objectively compared. Third, the object-oriented development environment used is a prototype and not a commercial system as are Oracle and Access. Thus, its development facilities are not as rich or mature as those of Oracle and Access. Any further study should use a commercial object-oriented development environment. Despite these limitations, we nevertheless feel that this study has shed some useful objective light on the effectiveness of object-oriented technology in developing and maintaining information systems.

References

[1] Ahmed, I., *Quantitative Evaluation of 4GLs Using Software Metrics*. MPhil thesis, Dept. of Computer Science, U. of Science & Technology, Hong Kong, 1996.

[2] Brooks, F., "No silver bullet: Essence and accidents of software engineering," *IEEE Computer* **20**(5), 1987, pp. 10-19.

[3] Conte, S.D., Dunsmore, H.E., and Shen, V.Y., *Software Engineering Metrics and Models*. The Benjamin/Cummings Publication Company, Inc., Redwood City, CA, 1986.

[4] Cox, B., *Object-Oriented Programming: An Evolutionary Approach*. Addison-Wesley Publishing Co., Reading, MA, 1986.

[5] Halstead, M.H., *Elements of Software Science*. Elsevier North Holland, New York, NY, 1977.

[6] Henry, S.M. and Kafura, D., "Software structure metrics based on information flow," *IEEE Transactions on Software Engineering* **SE-7**(5), 1981, pp. 510-518.

[7] Kokol, P., Rozman, I., and Venuti, V., "User interface metrics," *ACM SIGPLAN Notices*, **30**(4), 1995, pp. 36-38.

[8] Martens, C., and Lochovsky, F.H., "OASIS: A programming environment for implementing distributed organizational support systems," *Conf. on Organizational Computing Systems*, 1991, pp. 29-42.

[9] Martens, C., and Woo, C.C. "OASIS2: An environment for developing organizational computing systems," *Intl. Computer Science Conf.*, 1992, pp. 460-466.

[10] McCabe, T.J., "A complexity measure," *IEEE Trans. on Software Engineering* **2**(6), 1976, pp. 308-320.

[11] Meyer, B., *Object-Oriented Software Construction*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1988.

[12] Microsoft Corp., *Microsoft Access: User's Guide*. 1994.

[13] Oracle Corporation, *Oracle7 Relational Database Management System: Feature Summary*. 1994.

[14] Rentsch, T., "Object-oriented programming," *SIGPLAN Notices* **17**(9), 1982, p. 51.