WPI-CS-TR-97-07 August 1997

1997 Computer Science Department MQP Review

by

George T. Heineman
Robert E. Kinicki

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Abstract

This report presents results of a peer review of MQPs conducted within the Computer Science Department during the Summer of 1997 as part of a campus-wide MQP review. The goal of the report is to assess whether the department MQPs are accomplishing their educational goals. The report identifies problems that need to be addressed and trends that need to be continued to make the MQPs a worthwhile learning experience. It reflects data and evaluations for 29 MQPs, involving 57 computer science students, that were completed between the Summer of 1996 and the Spring of 1997. The report also makes comparisons to similar reviews done in 1991, 1993, and 1995.

Overall, the large majority of the projects are meeting the educational goals of the department as good learning experiences. The reviews indicate, however, that the overall quality of the projects is down from 1995. The reviewers believe that the faculty must set reasonable expectations for all project students. This will be increasingly important as the department continues to grow in size. The report draws a number of conclusions about the success of the projects based upon the data collected and evaluations done for this review. The report concludes with recommendations for future projects.

1 Introduction

1.1 Purpose

The Major Qualifying Project (MQP) is required of all undergraduate students at Worcester Polytechnic Institute. The MQP within the Computer Science Department is a capstone experience, requiring one unit of work, that gives students practice on applying the fundamentals and skills they have learned to a large problem in the field of Computer Science. The project may involve original research, data collection, analysis, or design of a system and often a software implementation. The approach is determined by the student/advisor team. The MQP allows students to study an area of Computer Science in depth, or allows them to combine areas into a single project.

This report presents results of a peer review of MQPs conducted within the Computer Science Department during the Summer of 1997 as part of a campus-wide MQP review. The goal of the report is to assess whether the department MQPs are accomplishing their educational goals. The report identifies problems that need to be addressed and trends that need to be continued to make the MQPs a worthwhile learning experience. It reflects data and evaluations for 29 MQPs, involving 57 computer science students, that were completed between the Summer of 1996 and the Spring of 1997. The report also makes comparisons to the following reviews:

Year	Number of MQPs	Number of students
1991	19	31
1993	26	44
1995	23	43
1997	29	57

1.2 Procedure

The peer review was conducted during the Summer of 1997 by Robert E. Kinicki, department head, and George T. Heineman, assistant professor. The review was to be for projects completed during the 1995-1996 and 1996-97 academic years. Rather than examine a sampling of reports for a two-year period, the peer review team examined projects completed between the Summer of 1996 and the Spring of 1997. The report for each MQP was obtained from either the project advisor or from the Gordon Library. Additional project information was gathered from CDR (Completion of Degree Requirement) records.

As in the previous review process [3], the reviewers conducted a detailed evaluation of all projects using the review form in Appendix A. The form contains information used in classifying the projects, questions quantified on a scale between 1 and 5, and has questions for written comments concerning the report. The form was designed to be easy to fill out with information that could be quickly collected and compared. Questions for written comments concerning the report were used to gather more detailed information about the project and give a means to express specific project strengths and weaknesses. Project grades and registration information was obtained from CDR records.

The MQP reports were divided between the two reviewers for evaluation. After all evaluations were completed, the data from the forms were collected and analyzed. This report is the outcome of the peer review process. Section 2 presents the results from the evaluation forms. Section 3 analyzes and correlates the results. Section 4 discusses conclusions and recommendations.

2 Results

This section presents the results of the Computer Science MQP evaluations. Along with presentation of the results are included reviewer comments (denoted by **Comment:**) which highlight the results and contrast them against those from previous reviews when appropriate. Note: All data are presented on a per project and not per student basis.

All percentages are represented in whole number amounts (i.e., 1/29 is represented as 3%), and all number averages are represented to one decimal accuracy (i.e., 1.97 is shown as 2). Because of this formatting, the percentages do not always total to 100%.

2.1 Faculty/Student Ratio

Table 1 shows the percentage of projects with the given numbers of students and faculty. Three (10%) of the projects had one or more advisors from outside the department (one Mechanical Engineering, one Civil Engineering, one Electrical and Computer Engineering). Three (10%) of the projects (two Electrical and Computer Engineering, one Computers with Applications, one Civil Engineering) involved a total of four students from other departments.

The average number of students per project was 2. The average number of faculty per project was 1.3.

Comment: The results show that only eight projects (28%) of the projects were done by a single student. This number is down sharply from the 1995 figure of 44% and the 1993 figure of 42%. This shows that the faculty is implementing a suggestion from the 1995 peer review to increase student team size. The number of projects advised by a single faculty member was 21 (73%), up from 67% in 1995. About the same number of projects involved faculty outside of the department as in 1995, but the number of students from other departments decreased by half to only four.

2.2 Faculty Project Load

Table 2 shows the distribution on the number of projects (co-)advised by each faculty member. There were eleven full-time faculty in Computer Science during AY96-97 (one faculty was on sabbatical, one started in January, one was on leave) plus two instructors

Table 1: Percentage of projects with the given number of students and faculty

	Students				
Faculty	**1**	**2**	**3**	**4+**	**Total**
1	7	11	2	1	73
2	0	4	3	0	24
3+	1	0	0	0	3
Total	28	62	17	3	100

who advised projects. Table 3 shows the same data, but in cases where projects were co-advised a weighting of one-half project load was given to each advisor. Note: The co-advisors from other departments are not shown in the tables.

Comment: The data show that one professor advised ten (34%) of the projects while two professors had no projects (one had returned from sabbatical, one first year faculty member was away B term). Aside from these, the project load was dispersed among the remaining eight faculty members. The average project load dropped slightly to 2.6, from 2.7 in 1993 and 2.8 in 1995. The comparable average loads shown in Table 3 remained at 2.2 projects/faculty, the same value as 1995, but up from 1.8 in 1993 and 1.5 in 1991.

These numbers are expected to significantly increase since the number of computer science majors at the Freshman and Sophomore levels is more than double the number of students in the Senior class. However, this should be offset by the recently hired faculty members in their second year.

Table 2: Distribution of projects advised or co-advised

Number of Projects (Co-)Advised	Number of Faculty
0	2
1	3
2	3
3	2
4	1
5	1
10	1

avg: 2.6 projects/faculty

2.3 Off-Campus Projects

Seven (24%) of the projects were sponsored or involved off-campus companies and organizations. The sponsors were Cabletron Systems, Planet Corporation, Stratus Corporation, Naval Research Laboratory, University of Puerto Rico Mayaguez (UPRM), and

Table 3: Distribution of load of projects advised

Load of Projects Advised	Number of Faculty
0	2
0.5	3
1	1
1.5	2
3.0	2
3.5	1
4.5	1
9.5	1

avg: 2.2 projects/faculty

Digital Equipment Corporation. One project started initially with Digital Equipment Corporation as a sponsor, but the students later completed the project independently. One project was associated with an off-campus organization: Worcester Art Museum. The remaining projects were done on-campus and not sponsored by off-campus companies.

Comment: This number of off-campus sponsored projects was slightly higher than the number in 1995.

2.4 Project Grades

In the projects reviewed, several student projects resulted in members on a given project receiving different individual grades. 72% of the projects (71% of the students) received a final grade of A, 27% of the projects (21% of the students) received a final grade of B, and 14% of the projects (7% of the students) received a final grade of C. These numbers are in line with the campus-wide historical averages where 70-75% of the students receive an A on their project.

Comment: These data indicate the number of A grades given to projects increased from the 1995 figures of 63% (60% of the students). The same number of projects received a grade of C as in 1995. These data seem to indicate that the faculty grading is in line with the campus as a whole. The number of projects with split grades reflects the increase in faculty attention to the individual performance of the students. The large percentage of A grades is noticeable, however.

2.5 Project Continuation

Two projects (7%) were continuations of prior MQPs and MS theses. The other projects were not directly related to other projects. These results are down from 1995.

2.6 Project Duration

Table 4 shows the duration of each project. 73% of the projects finished with one unit of work.

Comment: This number compares to 42% (1991), 54% (1993) and 63% (1995) projects completing with one unit of work. These figures indicate a trend of better efficiency by students and faculty in completing projects on time; the faculty is successfully implementing recommendations from previous peer reviews (dating back to 1991) to encourage completing projects on-time.

Table 4: Percentage of projects with the given duration

Total Units	Total	Percentage
1	21	73
1 1/12	1	3
1 1/6	7	24

2.7 Project Report Size

The average size of the project reports was 59 pages (± 26 with a range of 20–138), which excludes appendices and code. The average size of the appendices for a report was 33 pages (± 32 with a range of 0–106).

Comment: The length of reports is about the same as previous years 45 (1991), 49 (1993) and 50 (1995).

2.8 References

The average number of references was 14 (± 8 with a range of 1–32) for each report. Many projects did not have an explicit literature review section, but referenced additional work through the course of the document. Each MQP was graded on its literature review, as shown in Table 5

Table 5: Evaluation of literature review

Grade	Percentage
High	17
Good	24
Poor	28
None	31

Comment: These numbers are virtually the same as in 1995. Students could have done a better job on referencing prior work. More than half of the MQPs were rated as having an insufficient literature review.

2.9 Type of Projects

24 (83%) of the projects contained design and implementation of a piece of software with the other projects involving design without actual coding of software. 4 (14%) of the projects involved data collection. 5 (17%) of the projects involved evaluating other systems or having the developed system evaluated. Two (7%) projects involved original research. Five projects involved simulations.

Comment: As in previous years a significant number of the projects involved a design component and in most cases implementation of a program. The reviewers believe that more of the developed systems should be evaluated by other users as part of the project life cycle. The use of software written by others caused problems for students in integrating it with their own work. This problem points to the need for more system integration tasks in our curriculum.

2.10 Project Area

Table 6 shows the percentage of projects that involved different areas of Computer Science. In some cases a project involved only one area while in other cases it involved multiple areas (thus the percentages total to over 100%).

Comment: As the data show there is a variation in the sub-areas of Computer Science covered by the projects, but there is an increasing focus on network management and/or software engineering. The area of human-computer interaction also had a strong showing, although down from previous years. While some of these numbers may be biased by the reviewers – whose research interests focus on network management and software engineering – the reviewers believe the networked dorm rooms has increased the number of students working on distributed, network problems. The need to write the necessary software allowing programs to communicate over a network reflects the increase in software engineering projects. The world-wide interest in Java also appears on the WPI campus as many projects involved Java, either in the form of Java Applets or stand-alone applications. This trend is expected to continue.

2.11 Software Used

Table 7 shows the relative use of different programming languages and other software in the projects. Some projects used more than one software tool (e.g. MS-Windows and C++) thus the percentages total to over 100%.

Comment: The use of the C programming language continues in the projects with a significant number of projects evolving to use C++, as seen in 1995. The increase in Perl

Table 6: Project areas by percentage

41%	Networks
34%	Software Engineering (principally part)
21%	Human-Computer Interaction (principally part)
14%	Artificial Intelligence/Robotics
14%	Applications
14%	Distributed Systems
14%	Operating Systems
7%	Database
7%	Information Management
3%	Architecture
3%	Graphics/Visualization
3%	Languages
3%	Visual Systems

is most likely from its use in CGI-bin scripts within a web site; C is also used to write CGI-bin scripts. An increasing number of projects constructed a web-based front-end using HTML and CGI-bin scripts.

Table 7: Software used by percentage

34%	C language
28%	C++ language
21%	Java language
17%	HTML
7%	Perl
7%	X-Windows
7%	other software packages (VB, Oracle)
3%	Lisp or a Lisp dialect language
13%	no software

2.12 Hardware Used

Table 8 shows the percentage of projects that used different types of hardware platforms for their work. The numbers do not add to 100 since some projects involved different hardware platforms.

Comment: The data show that more projects were done using PCs than workstations, a continuing trend from the 1995 data. Reasons for these results may be the increased personal use of PCs by students, the wide availability of C++ compilers on PCs and the number of projects done with companies which are using PCs. Also, since more projects involved HTML and Java programming, there was less emphasis on the particular hardware used. In many cases, the projects did not describe which machines were used – this should be required for each project.

Table 8: Hardware used by percentage

45%	PC
20%	workstation (Digital, Sun)
3%	Stratus workstation
3%	neutral (i.e., Java)
38%	Unknown

2.13 Computer Science Classes

In previous reviews, an attempt was made to determine which courses contributed in some way to the success of the project. This is an important question, since it helps validate the selection of courses in our curricula. However, the reviewers felt it was not possible to accurately select a set of courses for each project. In many cases, the reviewers simply selected the courses in the project area (see Section 2.10). There is concern by the reviewers that the current curricula does not exactly match the student's interests as seen in the projects. The addition of new courses, such as WebWare, will help, but perhaps an overall assessment is needed.

2.14 Project Evaluations

The numerical evaluations of the projects are shown in Tables 9 and 10. The average and distribution (by percentage) of evaluation for each question is shown. Note: The "stat" level on the Math Level question represents any mathematics between calculus and the senior-level, such as probability and statistics or linear algebra.

Comment: Some projects (17%) do not adequately state the objectives of the project, a sharp increase from previous years. Most projects met or exceeded their objectives, although 17% failed to do so, perhaps because the objectives were too ambitious, the project simply did not complete the work, or the objectives changed during the project. These numbers reflect worse performance relative to 1995.

A surprising number (37%) of projects do not include an adequate abstract of their report. This is much higher than in past years (11% in 1995). Each project report must include a satisfactory abstract; there must be improvement in this area.

The motivation for the projects shows a wide distribution; 34% of the projects failed to describe proper motivation. Some of this can be attributed to the overall poor literature reviews (Section 2.8). Those poorly motivated projects often failed to explain why they were implementing a particular application. There was an increase from 7% in 1995 to

Table 9: Project evaluations by percentage

	1	2	3	4	5	avg.
Abstract accurate and complete	missing	poor	adequate		excellent	
	3	34	34	28	0	2.9
Clearly stated project objective	poor		adequate		excellent	
	3	14	41	38	3	3.2
Objective met	unk	no	mostly	yes	exceeded	
	0	17	59	21	3	3.1
CS Level	1000	2000	3000	4000	grad	
	0	17	41	38	3	3.3
Math Level	none	calc	stat	4000	grad	
	62	7	21	3	0	1.6
Motivate the project?	poor		adequate		excellent	
	17	17	31	17	17	3
Style, grammar, spelling	poor		adequate		excellent	
	10	17	34	34	3	3
Quality of Tables/ Diagrams/Figures		poor	adequate		excellent	
	0	31	38	21	10	3.1

Table 10: Project evaluations by percentage (cont.)

	1	**2**	**3**	**4**	**5**	**avg.**
Project Methodology	unknown	poor	adequate		excellent	
	7	31	34	24	3	2.9
Issues/Problems	poor		adequate		excellent	
Discussed	3	17	38	38	3	3.2
Overall report	poor		adequate		excellent	
organization	3	21	41	28	7	3.1
Programming Effort	none		some		considerable	
	7	17	28	48	0	3.2
Overall Effort Level	too little		about right		too much	
(worth 1 unit/student)	17	21	41	21	0	2.7
Quality of report	poor		adequate		excellent	
	7	28	14	38	14	3.2
Quality of project	poor		adequate		excellent	
	7	21	31	24	17	3.2
Quality of	unknown	poor	adequate		excellent	
presentation	59	14	21	7	0	2.8

17% this year for those projects rated as having excellent motivation.

The project methodology continues to remain a problem. 38% of the projects had either no methodology or one that was poorly described (as compared to 15% in 1995). On the positive side, 27% exhibited better than average methodology (up from 22% in 1995).

In general the reports do a satisfactory job in motivating and explaining the context of why the project was done. The reports were less thorough in discussing the design and methodology of how the project was carried out, and in discussing the issues and problems faced in the course of working on the project. The quality of report style and grammar was down sharply – this year 27% of the projects were less than adequate, up from only 4% in 1995.

The quality of reports exhibit a bi-modal distribution, with 35% being less than adequate, and 52% more than adequate. Contrasting with this, the quality of the projects decreased (28% less than adequate this year as compared to 19% from 1995). These results indicate that there is a widening disparity between project and report quality. The lower-evaluated reports needed better organization (24% were less than adequate) and motivation. The number of reports with poor figures increased sharply to 31%, perhaps contributing to the poor quality of the reports.

There was a surprising increase (to 17%) in projects whose Computer Science knowledge level was only at the 2000-level. This occurred, perhaps, because of the increase in web-based projects requiring only simple programming. This was combined with a decrease in projects requiring 4000-level classes. The reviewers feel there is difficulty in evaluating the accurate level for each project. The projects exhibited roughly the same mathematics level as in 1995.

The overall effort of the students appears the same as 1995; no projects were rated as requiring too much effort. The programming effort was also the same as 1995.

The overall project quality shows two trends. First, the number of less than adequate projects is continuing to increase, from 15% in 1993, to 19% in 1995, to 28% this year. Second, the number of above adequate projects remains somewhat constant 38% (8% excellent) in 1993 to 41% (15% excellent) in 1995, to 41% (17% excellent).

The quality of the presentations was difficult to judge for the reviewers and more than half of the evaluations were unknown. The average of 2.8 is for the 12 presentations for which the reviewers attended. In the future, the reviewers of the MQP reports should be decided in advance so they can attend the MQP presentations. The presentation facilities were adequate, using Perreault hall and the IMC television room. In only one case, the presentation machine crashed, forcing a delay in the presentation.

2.15 Project as a Learning Experience

Almost all of the projects (79%) were a good learning experience. The few projects that were not so rated resulted from: too simplistic a computer science component, little rationale for choosing the design of an implementation, lack of consideration of alternatives, not enough effort. The reviewers feel the students should be told that writing an application is not sufficient for an MQP.

Table 11: Project Strengths

Tried to utilize new interface technologies not taught in courses.
Embedded tags easy to understand by HTML-savvy professors.
Thorough Analysis.
Solid CS foundation.
Well thought out.
Report reads like an MS thesis. Excellent organization, well motivated, very useful to company.
Strong Java programming. Good distributed programming experience.
Full understanding of proprietary SECS/HSMS protocols. Well-created design document.
Attacks hard theoretical problem.
Off campus. Real life problem.
Actual working implementation. Design methods for OO.
Some very nice figures.
Good topic.
Implementation works. User feedback incorporated.
Experimental attitude. Selection of program.
Attacking topics not normally covered by undergrad curricula.
Shows difficulty when software and hardware are changing and must constantly interface with each other.
Good interdisciplinary project. Thorough presentation.
Solid understanding of ATM simulation experience.
Good division of workload into 4 communicating executables. Solid professional look/feel.
Tried to help existing organization with MIS.
Good motivation and clear scope. Integration with database. Interesting application.
Good future work. Good comparison with existing tools.
Very specific goal on which group kept their focus.

2.16 Project Strengths

Table 11 contains specific reviewer comments extracted from the evaluation forms concerning project strengths.

Comment: As in previous reviews, the projects were good when they were well-motivated, had a clear presentation indicating what was done, had a good design, and followed through on a particular topic.

2.17 Project Weaknesses

Table 12 contains specific reviewer comments extracted from the evaluation forms concerning project weaknesses.

Comment: As in previous reviews, projects with problems showed simplistic objectives, poor planning, and poor presentation of what was done.

Table 12: Project Weaknesses

Not well done. Structure of writing not good. Lack of sentences.
CS work not strong – especially Databases.
No evaluation. No methodology.
No real motivation. No real exams used. No grading engine developed, nor any discussion of the privacy/security issues. Plagiarism?
Student underestimated project. Should have been 3/4 students.
Some ad hoc treatment of rules. No example showing agent rules.
Not much development. Close to being an IQP.
No test application developed with the project.
Ad hoc treatment of problems/issues that any distributed system must face.
No characterization of how system would be built (if at all). No thorough evaluation of missing parts, strengths, weaknesses. No example of possible use/scenarios.
More discussion of the hurricane programs needed.
Report not well written. Didn't do much.
Motivation entirely from company. State case for reuse in general.
No figures of product they claimed to develop.
Flawed methodology.
No context or comparison. No design.
More thorough analysis needed. Summary of problems/issues. Needs Better design.
Report wanders a bit.
No formal architecture of real time system. No Software Engineering principles applied.
Not finished. No methodology. No testing.
Overestimated simulation part of the project.
More discussion of methodology and expected results.
Not enough evaluation of usefulness of 3d visualization. Too many programming errors.
Not enough modeling, quantitative analysis. No clear objectives. More like an IQP.
No discussion of design. Not enough examples.
Report was short and incomplete. No discussion of design.
Ad hoc approach. Very little design.
Ad hoc treatment of code. No design.

Table 13: Expected correlation between project quality and grade

	Project Quality				
Grade	1	2	3	4	5
C			■	■	■
B/C	■			■	■
B	■				■
A/B	■	■			■
A	■	■	■		

2.18 Interdisciplinary Work

There were four projects involving other departments: Physical Education, Mechanical Engineering (structural engineering and vibration study), Civil Engineering, and Electrical Engineering (Signals).

Comment: The number of non-computer science students involved in the projects dropped slightly from 1995.

3 Analysis of Results

This section correlates various aspects of the MQPs with the evaluations the projects received. This analysis is intended to help identify which project characteristics tend to yield good projects and which traits result in lower quality projects.

3.1 Correlation of Evaluations

The following correlations show the relationship between various results and the project evaluations. The project grades and project evaluations are shown for all projects. Note: For sake of comparison the value 4 is assigned to an A project grade, a value 3 to a B project grade and a value 2 is assigned to a C project grade. Recall the project evaluations had a 1 to 5 range where 1 is poor, 2 is fair, 3 is adequate, 4 is good, and 5 is an excellent project. Because of the difference in these scales, we set the standard for correlation as shown in Table 13. We suggest that an A should never be rated less than a 4, a B should receive an evaluation of 2, 3, or 4, and a C should receive a 1 or a 2. Each entry with a black box shows poor correlation. A project with a split grade of A/B should receive a 3 or 4.

To start our analysis, we compare the two evaluation criteria taken from the reviewer questionnaire: project grade assigned by the advisor and the project quality (PQ). Table 14 shows the correlation between the project evaluation and the project grade assigned by the advisor. The projects were evaluated before obtaining the project grade.

Table 14: Correlation of project grade with quality of project

	Project Quality					
Grade	1	2	3	4	5	Total
C	3	0	0	0	0	3
B/C	3	3	0	**3**	0	10
B	0	3	7	3	0	14
A/B	0	0	3	0	0	3
A	0	**14**	**21**	17	17	69
Total	7	21	31	24	17	100

Comment: There is a disparity between the two evaluation measures for the projects. There are three cases to consider:

- C1 The adviser and reviewer agree in their assessment of the project.
- C2 The adviser graded too harshly or the reviewers overrated the project.
- C3 The advisor graded too easily or the reviewer underrated the project.

The results show that while 62% of the projects have correlating evaluations (C1), 35% fall into case C3, and 3% (only one project) fall into case C2. In Table 14, cases C2 and C3 are represented by **bold** faced entries.

We now try to explain case C3. The peer review team divided the project reports between them; Heineman read nineteen projects while Kinicki read ten. On the key questions of quality of report (RQ) and quality of project (PQ) in Table 10, the average for all projects was 3.2 (± 1.2). When separated by reviewer, there is no significant difference:

Reviewer	Report Quality	Project Quality
GTH	3.2 ± 1.3	3.3 ± 1.3
REK	3.4 ± 1.1	3.1 ± 1

The greatest variance between the two reviewers appears on the results to q1 (Abstract accurate and complete) and q6 (Motivate the project):

Reviewer	Abstract	Motivation
GTH	2.5 ± .8	2.6 ± 1.3
REK	3.5 ± .5	3.8 ± 1.1

This may be attributed in part to the larger sample of projects reviewed by Heineman. There thus appears to be no reviewer bias regarding the projects falling under case C3.

For case C3, the reviewers agree that the quality of the projects is not entirely correlated with the individual grades assigned by the project advisor. 14% of the projects received an A grade although they were assessed to be less than adequate. 21% of the A projects were rated as being adequate, but the A grade should be reserved for those projects that are more than adequate. Either the reviewers did not fully comprehend the significance of the work or the students and advisors agreed upon a less than adequate project. There is room for improvement here, and as the number of MQPs in our department grows, the faculty needs to pay attention to standardizing the quality and effort of all MQPs.

For case C2, one project that was rated as above adequate received a B/C grade. This most likely reflects the dissatisfaction the project advisor had with one of the team members, so the weaker student member received the lower grade.

3.2 Correlation of Faculty Team Size and Evaluation

Table 15 shows the correlation between the number of faculty and the project evaluations. The two indicators are report quality (RQ) and project quality (PQ).

Table 15: Correlation of faculty team size and evaluation

Faculty Team Size	% of Projects	avg Grade	avg RQ	avg PQ
1	72	3.6	3.3	3.2
2+	28	3.7	3	3.2

Comment: The data show mixed results. Single-faculty projects receive slightly lower grades but have higher report quality ratings. The difference in project quality is insignificant.

3.3 Correlation of Student Team Size and Evaluation

Table 16 shows the correlation between the number of students and the project evaluations.

Table 16: Correlation of Student Team Size and Evaluation

Student Team Size	% of Projects	avg Grade	avg RQ	avg PQ
1	28	3.6	3.3	3.4
2	52	3.5	3.1	3.1
3	17	3.8	3.4	3.2
4	3	4.0	5.0	5.0

Comment: The analysis shows that two-student projects consistently fared the worst among all evaluations. This may be explained by the four projects this year (all with two students) with split grades. Another suggestion is that two-student projects are unable to define and complete large enough projects. The evaluations for single-student projects are much better than 1995 results.

3.4 Correlation of On/Off-Campus Projects and Evaluation

Table 17 shows the correlation between projects that were sponsored on/off-campus and the project evaluations.

Table 17: Correlation of On/Off-Campus Projects and Evaluation

Type	% of Projects	avg Grade	avg RQ	avg PQ
On	76	3.6	3.3	3.2
Assoc.	3	4.0	2.0	2.0
Off	21	3.8	3.2	3.5

Comment: Projects that were either off-campus or associated with an organization received higher grades than the on-campus projects. The project associated with, but not sponsored by, an outside organization received low reviewer evaluations even though the students received A grades.

3.5 Correlation of Project Duration and Evaluation

Table 18 shows the correlation between the project duration and the project evaluations.

Table 18: Correlation of Project Duration and Evaluation

Project Duration (Units)	% of Projects	avg Grade	avg RQ	avg PQ
1	72	3.6	3	3.3
1 1/12	3	2.5	2	2
1 1/6	24	3.8	4	3.9

Comment: Projects that were completed with more than one unit of work typically evaluated higher, although one project with an additional 1/12 unit scored rather low.

3.6 Correlation of Project Report Size and Evaluation

Table 19 shows the correlation between the project report size and the project evaluations. Note: The report size in Table 19 does not include code and appendices, which in some cases were larger than the report itself.

Comment: The results of this correlation shows that grades given by the advisor correlate directly to the size of the project report. Historically, shorter reports indicate that

Table 19: Correlation of Project Report Size and Evaluation

Project Report Size	% of Projects	avg Grade	avg RQ	avg PQ
0–30 pgs.	17	3.3	2.0	2.0
31–56 pgs.	38	3.6	3.4	3.4
57–82 pgs.	34	3.7	3.2	3.2
83–138 pgs.	10	4.0	5.0	5.0

students did not accomplish much or that they did not allocate enough time to write an adequate report. The results also show that the report quality and project quality are highly correlated with the size of the report. The clear marks from the reviewers for reports of less than 30 pages show a difference with the grade given for these projects. In this year's review, the mean project size was 57 with a standard deviation of 26.

3.7 Correlation of Computer Science Level and Evaluation

Table 20 shows the correlation between the Computer Science level and the project evaluations.

Table 20: Correlation of Computer Science Level and Evaluation

Computer Science Level	% of Projects	avg Grade	avg RQ	avg PQ
2000	17	3.3	1.8	3.4
3000	41	3.6	3.2	3.3
4000	38	3.8	3.8	2.9
grad	3	4.0	5.0	4.0

Comment: The data show that projects done at the CS 4000 level and higher tend to receive the best evaluations for report quality from the reviewers. The project grade is highly correlated to the CS level, as it should be. The project quality average for 4000 level projects was surprisingly less-than adequate. This could be from the large number of network projects that were completed this year. The reviewers feel that there is not enough communication between the different project groups working on the same area or topic; there is room for improvement.

3.8 Correlation of Math Level and Evaluation

Table 21 shows the correlation between the math level and the project evaluations.

Comment: Projects that involved some math received better grades and evaluations. Part of the reason may be that stronger students are taking on these projects. Another

Table 21: Correlation of Math Level and Evaluation

Math Level	% of Projects	avg Grade	avg RQ	avg PQ
none	69	3.6	2.9	2.9
calc	7	4.0	4.0	4.0
stat	21	3.6	4.2	4.2
4000	3	4.0	4.0	4.0

consideration is that the topic requires more effort. Current percentages are roughly the same as comparable to 1993 and 1995 results.

3.9 Correlation of Overall Effort Level and Evaluation

Table 22 shows the correlation between the overall effort level and the project evaluations.

Table 22: Correlation of Overall Effort Level and Evaluation

Overall Effort Level	% of Projects	avg Grade	avg RQ	avg PQ
1	17	3.3	2.0	1.6
2	21	3.3	2.0	2.5
3	41	3.7	3.8	3.7
4	21	4	4.5	4.5
5	0			

Comment: As expected, there is a strong correlation.

4 Conclusions and Recommendations

The 1997 review of Computer Science MQPs reflects data and evaluations for 29 MQPs, involving 57 computer science students, that were completed between the Summer of 1996 and the Spring of 1997. In this section, we attempt to draw some conclusions from the data collected during the evaluation process. Although 29 reports does not provide a large set of data points, some conclusions can be drawn from the data collected from the evaluation process.

4.1 Quality of Project

The overall project quality shows two trends. First, the number of less than adequate projects is continuing to increase, from 15% in 1993, to 19% in 1995, to 28% this year.

Second, the number of above adequate projects remains somewhat constant 38% (8% excellent) in 1993 to 41% (15% excellent) in 1995, to 41% (17% excellent). As the number of projects increase in the Computer Science department, we must try to both decrease the number of less than adequate projects and increase the number of exceptional projects.

Most of the MQPs were good capstone learning experiences for CS majors and meet the educational goals of the department. There was some concern on a few of the projects as good learning experiences. These problematic projects showed little rationale for choosing the design, displayed a lack of consideration for alternatives or indicated the students did not expend enough effort.

Many of the MQPs were judged to involve the appropriate amount of student effort. Typical Computer Science MQPs include the design and implementation of a large piece of software with many following the software life cycle from requirements gathering to implementation. Unfortunately not enough had results on testing and evaluation of the work. Also, the project methodology was poorly documented so the students could not explain why the students carried out their actions.

4.2 Quality of Report and Abstract

The quality of the reports themselves exhibited a bi-modal distribution this year – the number of less than adequate reports increased, as did the number of more than adequate reports. The quality of report style and grammar was down sharply – this year 27% of the projects were less than adequate, up from only 4% in 1995. The reports that were evaluated lower needed better organization and needed to be more complete.

Some of the reports lacked proper structure for a scientific paper or a technical report. Project goals were not always clearly stated, and the conclusion chapter occasionally did not evaluate how well the original objectives were met. Most of the reports could have been improved by a better literature review or by an explanation of how the MQP fits in with previous work, particularly other MQPs. The most common causes for weaker projects were lack of a clear plan of attack, insufficient work completed by the students, difficulties with the posed problem, underestimating the required work, or inadequate time allocated to writing the report. Not enough time and planning for the report was a problem with both some good and fair projects.

4.3 Students per MQP

The number of single student CS MQPs was down sharply to 28% (versus 44% in 1995 and 42% in 1993). This is important and shows that the faculty are successfully able to group students together on projects. However, in four years, the number of CS students will more than double, thus there will be increasing pressure to do more multi-student projects to keep faculty project load at a reasonable number. The results show that multi-student projects receive higher evaluations.

4.4 Distribution of CS Faculty over MQPs

This year one CS faculty member advised ten projects (almost half in the department). Since the department has hired several new faculty members, there needs to be a better distribution of CS faculty to the MQPs. However, next year the number of students involved in CS projects may likely jump to over 70. It will be important to maintain a proper distribution as the overall project load has increased and will continue to do so. The grades of co-advised projects were slightly better than single advisor CS MQPs.

4.5 Off-Campus Projects

In the 1991 review there were perceived problems in off-campus projects. The past three reviews indicate no difference between evaluations of on and off-campus projects. This is a positive result and indicates the quality of the two types of projects is comparable.

4.6 Project Resources

The project data show mixed results on the environment for software-oriented CS MQPs. As in the previous review many projects were done with C on Unix workstations, but others were done with C++ or on a PC platform. The C++ trend is a reflection of industry and has ramifications for our introductory curriculum. Reasons for the increased PC use may be the increased personal use of PCs by students, the wiring of the student dorms, the wide availability of C++ compilers on PCs and the number of projects done with companies that are using PCs.

More projects are being developed with web-based front ends, or using Java applets. Currently, there are no dedicated resources, such as Adobe PageMill, or Java environments such as Visual J++ and Symantec Cafe. The introduction of the WebWare course will address these issues, but perhaps the department should look into other resources.

4.7 Interdisciplinary Involvement

The number of interdisciplinary projects dropped slightly from previous years, but there was still involvement with four other departments on campus. This result is encouraging and indicates continued interest on interdisciplinary projects by our students and faculty.

4.8 Recommendations for the Next CS MQP Review

The evaluation process worked well. Again the biggest problem was evaluating oral presentations. Earlier identification of the oral presentations would allow for correlation with the projects themselves.

The reviewers had difficulty in determining which CS undergraduate classes contributed in some way to the success of each project. There is a concern that the current curricula does not exactly match the student's interests as seen in the projects. The addition of new courses, such as WebWare, will help, but perhaps an overall assessment is needed.

4.9 Recommendations for Improving CS MQPs

The following list of recommendations are drawn from the analysis and conclusions of this Computer Science MQP Peer Review. Most of the recommendations are aimed at CS MQPs, but a few may apply to the success of MQPs campus-wide.

- Increase student team size. There is still room for improvement as this recommendation will only become more important with increased enrollments. The results indicate that larger projects generally lead to better grades on the part of the students. Although optimal in a few situations, single student projects should be discouraged. Better mechanisms for bringing project groups together earlier need to be investigated. Working in project groups improves cooperative and communication skills of the students. Larger MQP teams offer more efficient use of a faculty member's time. It may be that more of the early CS courses should include group assignments. For some reason, this year two-student projects were not as successful.

- Encourage co-advising. The results show it leads to slightly better projects, and co-advising is a good mechanism for learning how to successfully advise projects. It is a way for faculty with expertise in unpopular subareas to become more involved and share the project load. It also encourages cross-pollination among our faculty both inside and outside of the department. Since the department has hired several faculty over the past two years, the reviewers feel that co-advising is a good introduction for new faculty members to experience the responsibilities of being a project advisor.

- Use better project planning. The project team (faculty and students) need to do a better job at planning the project and organizing the work. The report should document the planning stage of the project and give a better sense of problems, design considerations, and adjustments in both the direction of the project and work assignments. More emphasis should be given by faculty on expecting formal project proposals.

- Emphasize the testing and evaluation phase. Lack of adequate evaluation by external sources was a problem with many of the design and implementation projects. This was often a problem because students rushed to finish the project and did not have time for adequate evaluation.

- Pay attention to technical writing methodology. Standard technical writing issues such as clear objectives, adequate literature search, report structure, and a thorough review of the project in the conclusion need to be emphasized more when producing a project report.

- Require a satisfactory abstract from each student project. A surprising number of projects had less than satisfactory abstracts. It should be mandatory that all projects have an abstract that adequately describes the project.

- Emphasize the need for students to indicate why the MQP was a good experience and what experiences/courses the MQP builds upon. It was difficult with some projects for the reviewers to understand the significance of the work and upon which prior student work the project built upon.

- Allot more time for writing the reports. This recommendation was made in the prior review and needs to be emphasized again. Many of the shorter reports were from projects that were not as good, although some of the better projects were diminished by reports that were not as high of quality as the project. Part of the problem is that students spend too much time working on the project and not enough time in conveying its significance in the report.

- Continue to encourage off-campus and interdisciplinary projects. These type of projects broaden the background of our students and faculty and help to make contacts with companies and other departments.

- Strive to have MQPs build on previous MQPs and projects. In industry, our graduates will have to learn how to work with old code from old projects, and one way we can address this is through building upon previous MQPs and theses. One project this year continued experiments from a thesis in the Electrical Engineering department. Each faculty member should strive to create a pipeline of projects, so the students can see the larger objective for their individual project.

References

[1] Robert E. Kinicki and Craig E. Wills. Computer science department MQP review. Technical Report WPI-CS-TR-91-13, Worcester Polytechnic Institute, July 1991.

[2] Robert E. Kinicki and Craig E. Wills. Computer science department MQP review. Technical Report WPI-CS-TR-93-5, Worcester Polytechnic Institute, August 1993.

[3] Robert E. Kinicki and Craig E. Wills. Computer science department MQP review. Technical Report WPI-CS-TR-95-1, Worcester Polytechnic Institute, August 1995.

A Review Form

The following three-page form was used to evaluate all MQP reports.

Project Students:____________________ Reviewer:____________________

1997 Computer Science MQP Review Form

1. Number and department of advisor(s)____________________
2. Number, year and department of MQP student(s)____________________
3. On/off-campus project and sponsor____________________
4. Final grade given to report____________________
5. Distribution of units to complete MQP

E96	A96	B96	C97	D97	E97	Total

6. Report length in pages (excluding appendices and code)____________________
7. Pages of appendices ____________________. User manual? Y/N.
8. Quality of literature review? None/Poor/Good/High. How many references?____________
9. Circle the following types of work and areas of computer science that are relevant for this project.

   Analytic
   Data Collection (Emperical)
   Design
   Design/Implementation
   Evaluation
   Research
   Simulation
   Survey
   Other____________________

   AI
   Architecture
   DataBase
   Graphics
   HCI
   Languages
   Software Engineering
   Operating Systems
   Distributed Systems
   Other____________________

   Theory
   Info Mgmt
   Applications
   Networks

10. Circle the following software languages, tools, and hardware resources used for this project.

    C
    C++
    HTML
    Lisp/Scheme
    Lisp
    X-Windows
    Java
    Perl/Tcl/Tk
    Other____________________

    Macintosh
    IBM/PC
    wpi Alpha
    Sun workstation
    DEC workstation
    Other____________________

    on-campus/off-campus?

11. What Computer Science classes were background for this project?

Abstract accurate and complete	1 missing	2 poor	3 adequate	4	5 excellent
Clearly stated project objective	1 poor	2	3 adequate	4	5 excellent
Objective met	1 unknown	2 no	3 mostly	4 yes	5 exceeded
CS Level	1 1000	2 2000	3 3000	4 4000	5 grad
Math Level	1 none	2 calc	3 stat	4 4000	5 grad
Motivate the project?	1 poor	2	3 adequate	4	5 excellent
Style, grammar, spelling	1 poor	2	3 adequate	4	5 excellent
Quality of Tables/ Diagrams/Figures	______ quantity	2 poor	3 adequate	4	5 excellent
Project Methodology	1 unknown	2 poor	3 adequate	4	5 excellent
Issues/Problems Discussed	1 poor	2	3 adequate	4	5 excellent
Overall report organization	1 poor	2	3 adequate	4	5 excellent
Programming Effort	1 none	2	3 some	4	5 considerable
Overall Effort Level (worth one unit/student)	1 too little	2	3 about right	4	5 too much
Quality of report	1 poor	2	3 adequate	4	5 excellent
Quality of project	1 poor	2	3 adequate	4	5 excellent
Quality of presentation	1 unknown	2 poor	3 adequate	4	5 excellent

1. Was this project a good learning experience? What was learned by the student(s)?

2. Was this project a continuation of an earlier project, and if so, did the students indicate the part of the work that is theirs?

3. Project strengths: Project weaknesses:

4. Did this project involve any interdisciplinary work? What departments and subjects were involved?

5. Other comments.