1995 Computer Science Department MQP Review

Robert E. Kinicki
Craig E. Wills

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

WPI-CS-TR-95-01

August 1, 1995

Abstract

This report presents results of a peer review of MQPs conducted within the Computer Science Department during the Summer of 1995 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 27 MQPs, involving 43 computer science students, that were completed between the Summer of 1994 and the Spring of 1995. The report also makes comparisons to similar reviews done in 1991 and 1993.

Overall, the large majority of the projects are meeting the educational goals of the department as good learning experiences. The reviews indicate the overall quality of the projects is good, about the same as in 1993 and a little better than 1991. 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.

Contents

List of Tables

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 1995 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 27 MQPs, involving 43 computer science students, that were completed between the Summer of 1994 and the Spring of 1995. The report also makes comparisons to similar reviews done in 1991 [1] with 19 projects, 31 students and 1993 [2] with 26 projects, 44 students.

1.2 Procedure

The peer review was conducted during the Summer of 1995 by Robert E. Kinicki, department head, and Craig E. Wills, assistant professor. The review was to be for projects completed during the 1993-94 and 1994-95 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 1994 and the Spring of 1995 (two projects were not evaluated because the reports were unavailable). 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 [2], 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, information 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. In addition, CDR records were used to obtain project grades and registration information.

The MQP reports were divided between the two reviewers for evaluation. After evaluations the reviewers checked assessments to ensure similar evaluations. 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.

2.1 Faculty/Student Ratio

Table 1 shows the percentage of projects with the given numbers of students and faculty. Four (15%) of the projects had one or more advisors from outside the department (two Mechanical Engineering, one Civil Engineering, one Management). Three (11%) of the projects (one Mechanical Engineering, one Civil Engineering, one Management) involved a total of eight students from other departments.

The average number of students per project was 1.8. The average number of faculty per project was 1.4.

Comment: The results show less than half (44%) of the projects were done by a single student. This number is about the same as the 1993 figure of 42%. As in 1993, most projects (67%) were advised by a single faculty member. About the same number of projects involved faculty and students outside of the department as in 1993.

Table 1: Percentage of Projects with the Given Number of Students and Faculty

	Students				
Faculty	1	2	3	4+	Total
1	33	26	7	0	67
2	11	7	11	0	30
3	0	0	0	4	4
Total	44	33	19	4	100

2.2 Faculty Project Load

Table 2 shows the distribution on the number of projects (co-)advised by each faculty member. There were ten full-time faculty in Computer Science during AY94-95 (two faculty were on sabbatical) plus one instructor who advised a project. Table 3 shows the same data, but in cases where projects were co-advised a weighting of one-half was given to each advisor. Note: The co-advisors from other departments are not shown in the tables.

Comment: The data show that the project load was dispersed among faculty as in 1993. Also, the average project load shows an increase from 1993 and 1991 when the comparable average loads given in Table 2 were 2.7 and 2.1 projects/faculty, respectively. Similarly, the comparable average loads in 1993 and 1991 for Table 3 were 1.8 and 1.5 projects/faculty, respectively. The reviewers expect these numbers to significantly increase as the number of computer science majors at the Freshman and Sophomore levels is nearly double the number of students in the Senior class and faculty resources are unlikely to keep pace with this increase.

Table 2: Distribution of Projects Advised or Co-Advised

Number of Projects (Co-)Advised	Number of Faculty
0	0
1	3
2	2
3	3
4	1
5	1
6	1

ave: 2.8 projects/faculty

2.3 Off-Campus Projects

Six (22%) of the projects were sponsored by off-campus companies and organizations. The sponsors were Digital Equipment Corporation, AT&T, 3Com Corp., Precision Software, Gilbane Building Co. and the Commonwealth Scientific and Industrial Research Organization (CSIRO) in Sydney, Australia. 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 1993.

Table 3: Distribution of Load of Projects Advised

Load of Projects Advised	Number of Faculty
0	0
0.5	1
1	3
1.5	1
2	1
2.5	2
3.0	0
3.5	1
4.0	0
4.5	2
ave: 2.2 projects/faculty	

2.4 Project Grades

In the projects reviewed, only one student project resulted in members on a given project receiving different individual grades. 63% of the projects (60% of the students) received a final grade of A, 22% of the projects (30% of the students) received a final grade of B and 15% of the projects (9% of the students) received a final grade of C. These numbers are lower than 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 and students decreased from the 1993 figures of 69% and 73%. In addition, four of the projects received a grade of C, versus two such projects in 1993 and none in 1991. These data seem to indicate a stricter grading policy on the part of faculty in response to campus-wide concerns of project grade inflation.

2.5 Project Duration

Table 4 shows the duration of each project. 63% of the projects finished with one unit of work.

Comment: This number compares to 42% (1991) and 54% (1993) of projects completing with one unit of work. These figures indicate a trend of better efficiency by students and faculty in completing projects on time.

2.6 Project Report Size

The average size of the project reports was 50 pages (range of 13–105), which excludes appendices and code. The average size of the appendices for a report was 24 pages (range of 0–123)

Table 4: Percentage of Projects with the Given Duration

Total Units	Pct.
1	63
1 1/6	30
1 1/3	7

Comment: The length of reports is about the same as the 1991 figure of 45 pages and the 1993 figure of 49 pages, which also excluded appendices.

2.7 Bibliography

The average number of references was 10 (range of 0–28) for each report. Many projects did not have an explicit literature review section, but referenced additional work through the course of the document.

Comment: These numbers are virtually the same as in 1993. A continuing comment is that students could have done a better job on referencing prior work, particularly prior MQPs.

2.8 Type of Projects

22 (81%) of the projects contained design and implementation of a piece of software with the other projects involving design without actual coding of software. One (4%) project involved surveying potential users as part of the design process. One (4%) project involved theoretical analysis. Nine (33%) projects involved evaluating/benchmarking other systems or having the developed system evaluated. One (4%) project involved original research.

Comment: As in previous years a significant number of the projects involved a design component and in most cases implementation of a program. More projects this year involved the evaluation of other systems. 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.

2.9 Project Area

Table 5 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.

Comment: As the data show there is a wide variation in the sub-areas of Computer Science covered by the projects. The area of human-computer interaction was the most identified area (about the same as 1993). Software engineering was identified less than 1993 indicating that students did not do as good of job in explicitly identifying the software engineering portion of the project. The number of Graphics, Artificial Intelligence/Robotics and Operating Systems/Networks projects were up from 1993. A new area dealing with building tools for exploring and managing information on the Internet Web had two projects. Another project that was not evaluated due to a missing report was also in this area. More projects in this area can be expected in the future.

Table 5: Project Areas by Percentage

15%	Artificial Intelligence/Robotics
0%	Architecture
7%	Database
22%	Graphics/Visualization
41%	Human-Computer Interaction (principally part)
0%	Languages/Assembler/Compiler Issues
0%	Numerical Analysis
19%	Software Engineering (principally part)
19%	Operating Systems
15%	Networks
4%	Theory
7%	Web/Electronic Documents
11%	other (Testing, Security/Risk Analysis Processing, Image Processing)

2.10 Software Used

Table 6 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++).

Comment: The use of the C programming language continues in the projects with a significant number of projects evolving to use C++. Many projects involved graphics or user interface packages not identified in Table 6.

2.11 Hardware Used

Table 7 shows the percentage of projects that used different types of hardware platforms for their work.

Table 6: Software Used by Percentage

44%	C language
0%	Pascal language
37%	C++ language
0%	Assembler language
0%	Lisp or a Lisp dialect language
4%	X-Windows
7%	MS-Windows
7%	MIDI (Musical Instrument Digital Interface)
11%	other software packages (SQL/JAM, ACE, VPExpert)
7%	no software

Comment: The data show that more projects were done using PCs than workstations, a reverse trend from the 1993 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.

Table 7: Hardware Used by Percentage

37%	workstation (Digital, Sun, SGI, IBM)
41%	PC
4%	Macintosh
4%	BBN Butterfly Parallel Processor
4%	Processor board
11%	none or unknown

2.12 Computer Science Classes

Table 8 shows the percentage of projects that built upon material in various Computer Science courses. Some projects involved material from more than one course.

Comment: In general, the figures are lower than in 1993 because it was difficult to determine if students took a particular class and the reviewers were less willing to indicate courses if it was not clear they had been taken. This classification is only an estimation and does not mean that the students took or did not take the indicated classes. A more explicit statement of what experiences and courses a project built upon would be good to include in the project reports.

Table 8: Computer Science Classes by Percentage

4%	CS4341 Artificial Intelligence
26%	CS3013/4513 Operating Systems
0%	CS3021 File Structures
30%	CS3041 Human-Computer Interaction
19%	CS3733 Software Engineering
0%	CS4231 Techniques of Simulation
4%	CS4431 Database Design
11%	CS4514 Computer Networks
0%	CS4533/4534 Programming Language Compilers
22%	CS4731 Computer Graphics
7%	CS4121/CS4123 Finite Automata/Theory of Computation
4%	Independent Study

2.13 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: The project reports do an adequate job in stating the project objective (only one project less than adequate), although five (19%) of the projects were judged to not meet their original objectives. Some projects did not meet the initial objectives because the objectives were too ambitious, while other projects simply did not complete enough work to accomplish the objectives.

Three (11%) of the projects either did not include or had a poor abstract in the report. Only one such project existed in 1993. A satisfactory abstract should be included with each project report and this is an obvious area for 100% compliance.

In general the reports do a good 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. These results were about the same as in 1993.

The overall quality of the reports themselves was better than adequate, but not quite as good as in 1993. Many more reports were reported to be adequate than in 1993 with both fewer less-than and greater-than adequate reports. These results indicate that some of the better projects did not have a report as good as the project itself. Style, spelling, and grammar were good with only one project receiving less than an adequate mark. The reports that were evaluated lower

Table 9: Project Evaluations by Percentage

Abstract accurate and	1	2	3	4	5
complete	n/a	poor	adequate		excellent
ave: 3.1	7	4	63	19	7
Clearly stated project	1	2	3	4	5
objective	poor		adequate		excellent
ave: 3.3	0	4	67	30	0
Objective met	1	2	3	4	5
	unk	no	yes		exceeded
ave: 3.1	0	19	52	30	0
Motivate the project?	1	2	3	4	5
	poor		adequate		excellent
ave: 3.4	0	4	59	30	7
Style, grammar, spelling	1	2	3	4	5
	poor		adequate		excellent
ave: 3.4	0	4	59	33	4
Quality of Tables/		2	3	4	5
Diagrams/Figures		poor	adequate		excellent
ave: 3.4	0	19	37	33	11
Design/Methodology	1	2	3	4	5
of project	unknown	poor	adequate		excellent
ave: 3.1	0	15	63	15	7
Issues/Problems Discussed	1	2	3	4	5
	poor		adequate		excellent
ave: 3.1	0	15	63	22	0

Table 10: Project Evaluations by Percentage (cont.)

Overall report organization	1	2	3	4	5
	poor		adequate		excellent
ave: 3.2	0	11	59	26	4
CS Level	1	2	3	4	5
	1000	2000	3000	4000	grad
ave: 3.7	0	0	30	70	0
Math Level	1	2	3	4	5
	none	calc	stat	4000	grad
ave: 1.5	70	7	22	0	0
Programming Effort	1	2	3	4	5
	none		some		considerable
ave: 3.2	7	19	33	30	11
Overall Effort Level	1	2	3	4	5
(worth one unit/student)	too little		about right		too much
ave: 3.1	4	7	63	22	4
Quality of report	1	2	3	4	5
	poor		adequate		excellent
ave: 3.2	7	4	56	26	7
Quality of project	1	2	3	4	5
	poor		adequate		excellent
ave: 3.4	0	19	41	26	15
Quality of presentation	1	2	3	4	5
	unknown	poor	adequate		excellent
ave: 2.1	52	7	26	7	7

needed better organization and needed to be more complete. The quality of tables, diagrams, and figures was a key aspect of good reports. Only two reports were less than adequate in this category.

All of the projects demonstrated a Computer Science knowledge of at least the 3000-level with 70% at the 4000-level. 70% of the projects demonstrated no explicit use of mathematics at the calculus level or above.

The overall effort of the students in the project was difficult to judge, particularly for multiple person projects, but only three projects appeared to be less than adequate in terms of effort. Many of the projects required a good amount of programming effort.

The overall quality of the projects was about the same as in 1993. In 1993 15% of the projects were judged less than adequate while this year 19% were judged as such. However the number of above adequate projects was up from 39% (8% excellent) in 1993 to 41% (15% excellent) this year.

The quality of the presentations was difficult to judge for the reviewers and many of the evaluations were unknown. To help with this evaluation the reviewers consulted Prof. David Brown who attended virtually all of the presentations and maintained records. According to his evaluations the presentations were: 17% excellent, 43% good, 35% adequate and 4% fair. These numbers indicate that overall the presentations are satisfactory, although it was obvious that some of the graphical presentations were hampered by less than desirable presentation equipment. Better presentation facilities need to be made available by the Instructional Media Center or the department.

2.14 Project as a Learning Experience

Almost all of the projects were a good learning experience. The few projects with problems resulted from a project that was too simplistic in its computer science component, gave little rationale for choosing the design, showed lack of consideration of alternatives or in which the students did not expend enough effort.

2.15 Project Continuation

One (4%) of the projects was a continuation of a prior IQP and three (11%) were a continuation of a prior MQP. The other projects were not directly related to other projects.

Comment: These results are about the same as in 1993.

2.16 Project Strengths

These are specific reviewer comments extracted from the evaluation forms concerning project strengths:

good practical programming/design project
experience working with previously created code
student(s) tackled a difficulty problem
sophisticated algorithms developed
part of a research team
color pictures are outstanding
potentially good topic
project required integration of a system
covered design to implementation
motivated work
work on a real problem
solved lots of real problems
dealt with code integration
understand network protocols
covered software life cycle
professional work
real system work
design portion
grasp of difficult topic
many interviews of users in design process

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

These are specific reviewer comments extracted from the evaluation forms concerning project weaknesses:

accomplishments were not a lot
report was terse and did not provide background
report seemed rushed
report did not give a good feel for how project fit together
students did not seem strong
not sure functions provided were the best
lack of closure, too ambitious
not clear of project success
work is too shallow
need a better summary
incomplete testing
did not complete project
lack of testing by potential users

lack of implementation and effort
report not written for reader not versed in area

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

2.18 Interdisciplinary Work

Ten (37%) projects involved interdisciplinary work. Two projects involved music, two projects involved mechanical engineering and two projects involved electrical and computer engineering. One project dealt with fire protection, one with a medical imaging, one with veterinary science and one with theater.

Comment: The number of non-computer science subjects involved in the projects was about the same as in 1993.

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.

Before analyzing various factors a comparison of the two evaluation criteria is shown. The two factors are the project grade assigned by the advisor and the project evaluation (PE), taken from the quality of project question, by the reviewers. Table 11 shows the correlation between the project evaluation and the project grade assigned by the advisor. Note: The project evaluations were done before obtaining the project grade.

Comment: There was a strong correlation between the two evaluation measures for the projects. As shown, one (4%) of the projects received only a fair evaluation, but received a grade of A. In this case, either the reviewers did not fully comprehend the significance of the work or the students and advisors agreed upon a less than adequate project. In contrast, one (4%) project received a good evaluation, but only a grade of B while one (4%) project received an adequate evaluation, but only a grade of C. In these cases, the reviewers judged the work

Table 11: Correlation of Project Grade with Project Evaluation

	Project Eval					
Grade	1	2	3	4	5	Total
C	0	11	4	0	0	15
B	0	4	15	4	0	22
A	0	4	22	22	15	63
Total	0	19	41	26	15	100

to be better than the grade given and either the advisor graded too harshly or the reviewers did not obtain a sense of other problems with the project.

3.2 Correlation of Faculty Team Size and Evaluation

Table 12 shows the correlation between the number of faculty and the project evaluations.

Table 12: Correlation of Faculty Team Size and Evaluation

Faculty Team Size	% of Projects	ave Grade	ave PE
1	67	3.4	3.3
2+	33	3.6	3.4

Comment: The data show slightly better evaluations for co-advised projects. Co-advising led to much better evaluations in 1991, but there was no correlation between co-advising and evaluations in 1993.

3.3 Correlation of Student Team Size and Evaluation

Table 13 shows the correlation between the number of students and the project evaluations.

Table 13: Correlation of Student Team Size and Evaluation

Student Team Size	% of Projects	ave Grade	ave PE
1	44	3.2	3.2
2	33	3.8	3.6
3+	22	3.7	3.3

Comment: The data show a larger student team size leads to better evaluations. This effect was even more pronounced in 1991 and less pronounced in 1993.

3.4 Correlation of On/Off-Campus Projects and Evaluation

Table 14 shows the correlation between projects that were sponsored on/off-campus and the project evaluations.

Table 14: Correlation of On/Off-Campus Projects and Evaluation

On/Off-Campus	% of Projects	ave Grade	ave PE
on	78	3.5	3.3
off	22	3.5	3.5

Comment: The projects that were sponsored on-campus and off-campus evaluated the same. This result is similar to 1993.

3.5 Correlation of Project Duration and Evaluation

Table 15 shows the correlation between the project duration and the project evaluations.

Table 15: Correlation of Project Duration and Evaluation

Project Duration (Units)	% of Projects	ave Grade	ave PE
1	63	3.4	3.2
1 1/6	30	3.6	3.6
1 1/3	7	4.0	3.5

Comment: Projects that were completed with one unit of work evaluated lower. This result is the opposite of 1993 when one-unit projects evaluated slightly higher. The reason for this change is unclear.

3.6 Correlation of Project Report Size and Evaluation

Table 16 shows the correlation between the project report size and the project evaluations. Note: The report size in Table 16 does not include code and appendices, which in some cases were larger than the report itself.

Table 16: Correlation of Project Report Size and Evaluation

Project Report Size	% of Projects	ave Grade	ave PE
0–39 pgs.	37	3.3	3.3
40–69 pgs.	41	3.4	3.5
70– pgs.	22	4.0	3.3

Comment: There is no clear difference in evaluations between the various report sizes in contrast to previous reviews. Historically, shorter reports indicated that students did not do a lot of work in the project or they did not allot enough time to write an adequate report. In this review there were a few good projects that had terse reports contained a lot of information in appendices.

3.7 Correlation of Quality of Tables and Evaluation

Table 17 shows the correlation between the quality of tables, diagrams, and figures and the project evaluations.

Table 17: Correlation of Quality of Tables and Evaluation

Quality of Tables/etc.	% of Projects	ave Grade	ave PE
poor	19	2.8	2.6
adequate	37	3.4	2.8
good	33	3.8	3.9
excellent	11	4.0	5.0

Comment: There is an even stronger correlation between the presentation of tables and figures and the quality of the project than was found in the 1993 review. These data indicate that well done and well explained projects naturally lead to the inclusion of tables and figures.

3.8 Correlation of Computer Science Level and Evaluation

Table 18 shows the correlation between the Computer Science level and the project evaluations.

Comment: The data show that projects done at the CS 4000 level and higher tend to receive the best evaluations, particularly from the reviewers. This effect was even more pronounced than for the 1993 review.

Table 18: Correlation of Computer Science Level and Evaluation

Computer Science Level	% of Projects	ave Grade	ave PE
2000	0	0.0	0.0
3000	30	2.9	2.4
4000	70	3.7	3.8
grad	0	0.0	0.0

3.9 Correlation of Math Level and Evaluation

Table 19 shows the correlation between the math level and the project evaluations.

Table 19: Correlation of Math Level and Evaluation

Math Level	% of Projects	ave Grade	ave PE
none	70	3.3	2.9
calc	7	4.0	5.0
stat	22	3.8	4.2
4000	0	0.0	0.0

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. There was a little more math used in these projects than in 1993.

3.10 Correlation of Overall Effort Level and Evaluation

Table 20 shows the correlation between the overall effort level and the project evaluations.

Table 20: Correlation of Overall Effort Level and Evaluation

Overall Effort Level	% of Projects	ave Grade	ave PE
too little	4	3.0	2.0
below about right	7	3.0	2.5
about right	63	3.4	3.1
above about right	22	4.0	4.5
too much	4	4.0	5.0

Comment: As expected, there is a strong correlation.

3.11 Correlation of Quality of the Report and Evaluation

Table 21 shows the correlation between the quality of the report and the project evaluations.

Table 21: Correlation of Quality of Report and Evaluation

Quality of Report	% of Projects	ave Grade	ave PE
poor	7	2.0	2.0
fair	4	4.0	2.0
adequate	56	3.4	3.1
good	26	3.9	4.0
excellent	7	4.0	5.0

Comment: As expected, there is a strong correlation.

4 Conclusions and Recommendations

The 1995 review of Computer Science MQPs reflects data and evaluations for 27 MQPs, involving 43 computer science students, that were completed between the Summer of 1994 and the Spring of 1995. Although 27 reports does not provide a large set of data points, a few conclusions can be made based on the data collected from the evaluation process.

4.1 Quality of Project

The overall quality of the projects was good, about the same as in 1993. In 1993 15% of the projects were judged as only fair while this year 19% were judged as such. However the number of more than adequate projects was up from 39% (8% excellent) in 1993 to 41% (15% excellent) this year.

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 a significant 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.

4.2 Quality of Report and Abstract

The quality of the reports themselves was better than adequate, but not quite as good as in 1993. Style, spelling, and grammar were good with only one project receiving less than an adequate mark. The reports that were evaluated lower needed better organization and needed to be more complete. The quality of tables, diagrams, and figures was a key aspect of good reports. Only two reports were less than adequate in this category with good projects evaluating well in this category.

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, 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 stayed about the same at 44% versus 42% in 1993. This result indicates that faculty are doing an adequate job of grouping students together on projects. However, with more students in the pipeline and about the same number of faculty there will be more pressure to have multi-student projects to keep the faculty project load at a reasonable number. The results show that multi-student projects tended towards higher evaluations.

4.4 Distribution of CS Faculty over MQPs

The distribution of CS faculty over the MQPs was reasonably spread. It will be important to maintain this 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 in contrast to 1993 when there was little difference between the two.

4.5 Off-Campus Projects

In the 1991 review there were perceived problems in off-campus projects. The past two 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 wide availability of C++ compilers on PCs and the number of projects done with companies that are using PCs.

4.7 Interdisciplinary Involvement

The projects this year again did a good job of involving students, faculty and topics from other disciplines. 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. The reviewers consulted Prof. David Brown who attended virtually all of the MQP oral presentations. Earlier identification of the oral presentations would allow for correlation with the projects themselves.

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.

- 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.

- 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.

- Faculty and students need to better 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.

- Faculty and students need to 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 an MQP report.

- Advisors must require a satisfactory abstract from each student project. Only a few projects had less than satisfactory abstracts, but it should be mandatory that all projects have an abstract that adequately describes the project.

- Advisors need to 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.

- Students need access to better equipment for oral presentations. Some of the graphical and interactive presentations did not successfully reflect the quality of the work done because of limitations of the display equipment.

- 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. There was no improvement this year from the previous review.

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.

A Review Form

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

Project Students:____________________ Reviewer:________________

1995 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

E94	A94	B94	C95	D95	E95	Total

6. Report length in pages (excluding appendices and code)________________
7. Pages of appendices ________________. User manual? Y/N.
8. Is there a literature review? Y/N. How many references?____________
9. Check the following types of work and areas of computer science that are relevant for this project.

Analytic	AI	Theory
Data Collection (Empirical)	Architecture	Info Mgmt
Design	DataBase	
Design/Implementation	Graphics	
Evaluation	HCI	
Research	Languages	
Simulation	Software Engineering	
Survey	Operating Systems	
Other____________________	Networks	
	Other____________________	

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

C	Macintosh
C++	IBM/PC
HTML	wpi Alpha
Scheme	Sun workstation
Lisp	DEC workstation
X-Windows	Other____________________
yacc/lex	on-campus/off-campus?
Other____________________	

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

Abstract accurate and complete	1 n/a	2 poor	3 adequate	4	5 excellent
Clearly stated project objective	1 poor	2	3 adequate	4	5 excellent
Objective met	1 unk	2 no	3 yes	4	5 exceeded
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
Design/Methodology of project	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
CS Level	1 1000	2 2000	3 3000	4 4000	5 grad
Math Level	1 none	2 calc	3 stat	4 4000	5 grad
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.