A COMPARISON OF PROGRAMMING TEAM PERFORMANCE ON SOFTWARE DEVELOPMENT PROJECTS

Donald L. Jordan
Computer Science Department
College of Engineering
Lamar University
Beaumont, TX 77710

Abstract

A senior level course for software project developments is offered every semester at Lamar University. The design and programming teams formed in the class are analyzed. The purpose of this paper is to present some statistics and key parameters from that analysis to compare the performance of large and small size programming teams on software development projects.

Introduction

At Lamar University the Systems Development Life Cycle methodology is covered by a two course sequence similiar to the ones described in the literature[1,2,4]. The first course, CS4311, emphasizes the requirements definition, systems analysis and general design phases. In this course, students are required to complete the analysis and general design of the information requirements for a small manufacturing company. They work in teams[4,5] and are required to submit as a deliverable product a systems reference manual that completely documents their analysis and design effort.

The second course, CS4312, emphasizes the detailed design, implementation and testing phases of the development life cycle. Again, students work in teams but are given as input the system reference manual produced by the previous CS4311 class. They must complete the design details, code and test the project software system. Depending on the overall class size and the complexity of the project assigned these design and programming teams can be small (3-6 members) or large (8-12 members). The deliverable products from each team is the object code, source code and a users manual.

The purpose of this paper is not to advocate the content, organization or sequencing of our courses. These topics have been thoroughly discussed and previously reported in the literature[2,6,7].

This paper examines and reports the performances of large and small programming teams from CS4312, which is our second course. CS4312 is offered every long semester and in the summer. It normally has an enrollment of 40 students in the Spring and Fall and 20 students in the summer.

Large Team Performance

The Spring 1986 class was chosen to illustrate the performances of large programming teams. This class had three teams of size 11, 11 and 12. All three teams were given the same general design and the first few days of the semester were devoted to the detailed design of that system. Each of the three teams met with the instructor, as a team, at least once a week to review the requirements and specifications. A critical design review (CDR) was held at the conclusion of this part of the course to insure that all teams understood the requirements of the system.

The teams were directed to organize themselves using either the Chief Programmer Team (CPT), Revised Chief Programmer Team (RCPT), or Surgical Team concept. They elected their own leaders, assistants, and prepared a project plan. Schedules were developed by each team with at least three software releases.

Teams were also allowed to choose an implementation language so long as it is supported on the existing Lamar University hardware configuration. We have available, for this course, a mainframe and several PC's. Traditionally, teams that wish to work on the mainframe choose from COBOL, FORTRAN or PASCAL. Teams that choose the PC have access to BASIC, COBOL, FORTRAN, Turbo PASCAL, and dBase III. The Spring class had three teams and all chose the mainframe. Two teams chose COBOL and one FORTRAN.

Since the software being developed was an interactive, menu-driven system, the menu subsystem was designated for the first release. Other requirements were equally divided between the remaining two releases. After the last release, about three weeks were left for system testing, demonstration and general use. In addition, a users manual was required.

At the conclusion of the semester, each team was required to gather data on the software that they had produced. Some selected parameters and their associated data is presented below. It should be noted that the COBOL systems were slightly larger in number of statements than the FORTRAN system. The number of modules between teams varied significantly; however, this was probably caused more by programming style than due to the language itself. It appears that the FORTRAN system is slightly more complex and would be harder to maintain due to the large number of compares per module and the number of modules with more than 50 statements.

Large Team Performance

Selected Parameters	Team #1 COBOL	Team #2 COBOL	Team #3 FORTRAN
Executable Statements	7029	6969	5179
Number of variables	234	87	101
Number of Arrays	10	26	16
% Comment Statements	19	26	9
*Number of modules	1518	570	71
Largest module size	98	77	91
Compares per module	1.2	1.2	5.5
Loops per module	1.7	0.3	1.0
Module size > 50 Statements	12	11	14
Modules with > 10 compares	0	4	2

* Paragraphs in COBOL and Subroutines in FORTRAN are considered modules.

Small Team Performance

The system software specifications and design that was delivered at the conclusion of CS4311 was reflected by a VTOC chart. Such a document shows the modularity of the proposed system. Each requirement can easily be equated to the necessary software modules. In this manner, a small programming team can be assigned an appropriate subset of the total system requirements. Further, since all programming teams in the same semester will have the same requirements then their performance can be compared.

These small size programming teams were organized in the same manner as large teams. In the Summer '86 term three programming teams of size 4, 4 and 5 were formed. These teams chose as their implementation languages dBase III, FORTRAN and PASCAL.

At the end of the summer term, the teams again gathered data on their software systems. This data is presented below. It is remarkable that all three teams wrote essentially the same amount of code for a given set of requirements. The PASCAL system would appear to be more complex when considering the number of arrays, largest module size, compares per module and the number of modules with more than 10 compares.

Small Team Performance

Selected Parameters	Team #1 dBase	Team #2 FORTRAN	Team #3 PASCAL
Executable Statements	1169	1044	1111
Number of Variables	47	47	38
Number of Arrays	12	09	16
% Comment Statements	2	8	18
*Number of Modules	31	19	22
Largest Module Size	106	157	135
Compares per Module	1.1	2.6	3.1
Loops per Module	0.8	3.0	1.5
Module Size > 50 Statements	0	6	5
Modules with > 10 compares	0	2	0

* Procedures in PASCAL, Subroutines in FORTRAN and programs in dBase III are considered modules.

Summary and Conclusions

No effort was made to instruct the teams on proper module size or complexity prior to the start of the coding phase. In most cases, the students have completed at least four programming courses prior to CS4312 to include BASIC, COBOL, FORTRAN and PASCAL. They have had a lot of experience in writing small toy programs as described by Oman[6] and others but no experience in programming a system of this magnitude. We do teach, at

all levels, concepts such as the top-down, structured approach to developing software and also emphasize the importance of modularity, error-checking and maintainability. Every student in the class had essentially the same programming background before they were assigned to a team. A few of the students chose an implementation language, such as COBOL, to refresh their knowledge and proficiency in that language just before graduation and entering the job market.

At the end of each semester, a course survey is conducted to determine student impressions and opinions. The critiques always give high marks to the course. Additionally, interviewers from both The Government and Industry continually give positive feedback. It is deemed by all that the type of problems assigned and the systems to be developed are very "real-world" and that the course should be continued using essentially the same format.

The results given in tabular form earlier in this paper show no significant differences between large or small teams or between teams in one language or another.

CS4312 can be improved and made a better course. Projects that are slightly smaller should be selected so that students have more time near the end of the semester to demonstrate their system. This would also permit the interchange of systems between teams so that one team becomes the users of another system. More software development tools for both the planning and coding activities could be incorporated into the course. Most interactive, menu-driven systems require considerable coding to generate menus and data entry panels. Several software packages are now available that assist in screen generation and error-checking of input data. We intend to require future classes to become familiar with and to use some of these packages. In this manner, the teams can become more efficient and productive. Also, software packages the improve system documentation and users manuals will be used.

References

[1] Boltz, Richard E. and Jones, Lawrence G., "A Realistic, Two-Course Sequence in Large Scale Software Engineering", SIGCSE Bulletin, Vol. 15, #1, February 1983, pp. 21-24.

[2] Collofello, James S. and Woodfield, Scott N., "A Project-Unified Software Engineering Course Sequence," SIGCSE Bulletin, Vol. 14, #1, February 1982, pp. 13-19.

[3] Perkins, Thomas E. and Beck, Lelanc L., "A Project Orientated Undergraduate Course Sequence in Software Engineering," SIGCSE Bulletin, Vol. 12, #1, February 1980, pp. 32-39.

[4] Sathi, Harbans L., "A Project Orientated Course for Software Systems Development," SIGCSE Bulletin, Vol. 16, #3, September 1984, pp. 2-4.

[5] Woodfield, Scott N. and Collofello, James S., "Some Insights and Experiences in Teaching Team Project Courses," SIGCSE Bulletin, Vol. 15, #1, February 1983, pp. 62-65.

[6] Oman, Paul W. Jr., "Software Engineering Practicums: A Case Study of a Senior Capstone Sequence," SIGCSE Bulletin, Vol. 18, #2, June 1986, pp. 53-57.

[7] Gillett, Will D. and Pollack, Seymour V., AN INTRODUCTION TO ENGINEERED SOFTWARE, Holt, Rinehart and Winston, New York, NY. (1982).

****

OVERVIEW-- continued from page 44

[2] Ghezzi, C. and Jazayeri, M., Programming Language Concepts John Wiley & Sons, Inc. 1982.

[3] Horowitz, E., Fundamentals of Programming Languages, 2nd Edition, Computer Science Press, 1984.,

[4] Pratt, T., Programming Languages 2nd Edition, Prentice-Hall, Inc. 1984.

[5] Tennent, R.D., Principles of Programming Languages, Prentice-Hall International, Inc. 1981.

[6] Marcotty, M. and Ledgard, H., Programming Language Landscape, 2nd Edition, SRA, Inc., 1986.