1982

Project management by checkpoint control

P. J. McKerrow
University of Wollongong, phillip@uow.edu.au

PROJECT MANAGEMENT BY CHECKPOINT CONTROL

Phillip John McKerrow

Department of Computing Science,
The University of Wollongong,
Post Office Box 1144,
Wollongong, N.S.W. 2500
Australia.

ABSTRACT

Students majoring in computing science are required to undertake a software project during their third year of study. The aim of the project course is to give students some "real-world" experience at designing and implementing a reasonably complex piece of software. Rapidly increasing student numbers has caused considerable supervision problems for academic staff.

A method of project management using checkpoint control meetings and a rigorous design discipline was used in 1981. This method allowed the staff to keep close contact with the students' progress without consuming large amounts of time while allowing students to develop individual and creative solutions to problems.

Keywords: Project Management, Checkpoint Control, Rigorous Design Discipline, Computing Science, Supervision, Students, Project Stages.

1. Introduction

Students majoring in Computing Science at The University of Wollongong are required to undertake a software project during their third year of study. Prior to this course the majority of their assignments consist of short programs designed to teach specific principles and algorithms. The aim of the project course is to give students experience at designing and implementing a reasonable size piece of software, typical of the jobs they are likely to get on entering the workforce [1]. It provides

them with an opportunity to acquire essential software engineering skills in the areas of time management, design methodology and idea communication.

For many students this is the first time they have to write a program larger than four pages of code. Thus as part of the course they are required to learn how to plan and execute a large project and, due to the time constraints, do it right the first time. This involves learning to think from a system design, application and user point of view rather than from a computer expert point of view.

Student numbers in computing science are increasing rapidly. Supervision of these projects can consume a considerable amount of staff time and pose complex evaluation problems. Therefore, the continuance of project courses depends upon finding management techniques that are efficient in staff time.

In previous years the students had all been excellent. The 1981 group included a broad spectrum of scholastic ability (table 1). An ad-hoc approach to project supervision had previously run into problems. A method of project supervision that allowed the following goals to be achieved was needed.

(i) Students should learn how to design and implement a large program within severe time constraints (14 weeks at 12 hours per week) that met the design requirements and was "bug free".

(ii) The work done should be useful either to the student, supervisor, department or employer in the future.

(iii) Students should be as independent and creative as possible.

(iv) A supervisor should be in complete touch with a project without it consuming excessive time.

(v) The evaluation procedure should be common to all projects and take into account both the effectiveness of the work done and the completeness of the project.

(vi) In addition to writing and documenting the code the student is required to hand in a project report and present a seminar to his peers, both of professional standard.

2. Concepts of Checkpoint Control

In an attempt to meet these goals an amalgum of project management methods [2] used successfully in industrial design laboratories was implemented. Management by checkpoint control consists of regular meetings between supervisor, student, and other interested parties throughout the course of a project. Checkpoint control differs from other management methods in the

very clear definition of the procedure at checkpoint meetings.

When a project is commenced an initial meeting is held at which:

(i) the goals of the project are defined,

(ii) all known aspects are discussed,

(iii) possible approaches are brainstormed,

(iv) the project is split into several logical stages,

(v) goals are set for the first stage,

(vi) resources are allocated,

(vii) dead lines are established, and

(viii) the next meeting is arranged.

At subsequent checkpoint meetings the design or code is walked through and the following topics are discussed:

(i) Progress since the last meeting.

(ii) Problems which have occurred and their solutions.

(iii) Reasons for failure to meet goals and corrective action to be taken.

(iv) Goals to be achieved by the next checkpoint.

(v) Time and place of the next checkpoint meeting.

The regularity of checkpoint meetings varies from project to project depending upon the ability and experience of the student. A meeting must be held at the end of each major stage and it may be necessary to meet more often. This method of project management has the following advantages:

(i) The supervisor can keep close contact with the project without it consuming large amounts of his time.

(ii) Due to lack of experience many students do not know how to start a project. Checkpoint control overcomes this problem through its structured approach, thus saving students time.

(iii) The student is free to develop his own ideas and to express himself within the framework of specific goals. As a result he is not frustrated by over supervision while still getting regular feedback on his progress.

(iv) Goals are set by the student in consultation with his supervisor, increasing his motivation to achieve them because he owns them.

(v) When a project starts to turn sour the supervisor is rapidly alerted to the fact.

(vi) Supervisors can easily gauge the amount of help a student needs and detect those students who cannot organise themselves.

3. Implementation of Checkpoint Control

The implementation of any project management scheme will vary from project to project due to particular characteristics of the project, the supervisor and the student. To maintain a commonality between projects, in order to simplify final marking, a structured implementation of checkpoint control was developed. This included formalising the concepts into strict guidelines for the checkpoint control meetings and a comprehensive description of the normal stages of a project.

In addition each student was required to hand in a progress report when each stage was complete. This forced students to get their thoughts onto paper in a coherent manner and simplified the production of the final report. Also it helped to teach good documentation techniques as well as providing a record of the work done for final project evaluation.

4. Stages of a Project

As many of the students had not previously attempted a project of this nature it was necessary to provide a project structure for them to work to. This gave a general indication of where the checkpoints should be and placed a heavy emphasis on design. Many students can code small tutorial problems directly and their natural inclination is to attempt a project in the same manner; normally resulting in disaster.

The design process deliberately flows from outputs, to algorithms, to data structure to inputs [3] in order to force the students to define clearly what his program is to do and then to determine what is needed to do it. This process is not always easy as the desired outputs may not be obvious. A group designing a personal computer operating system took several weeks to realise that the outputs are the results of the user commands and thus the design should be commenced by defining these commands.

The amount of time spent working on each stage was left to the student but a guide was given by setting final dates for the submission of stage progress reports. The six recommended project stages are outlined below.

1. Project Commencement - Following the initial checkpoint meeting the student has to:

(i) write a description of the project,

(ii) define the expected results,

(iii) split the project into major stages,

(iv) allocate completion dates for each stage, and

(v) compile a list of relevant literature, software and hardware.

2. System Design - define in concept a method of achieving the desired results.

(i) List the major sections and their underlying concepts.

(ii) Develop a block diagram of the system showing the major parts and the desired data flow.

(iii) Write a general description of each section.

(iv) Specify the necessary hardware and software.

(v) Define in concept and then in detail the program outputs including displays, reports, files and instruction sequences.

3. Detailed Design - fleshes out the system, starting with a precise definition of the outputs and working progressively toward the inputs.

(i) Complete the detailed design of the outputs. Specify exactly the interface to the person or program receiving these outputs (draw diagrams of displays).

(ii) Establish the algorithms needed to produce these outputs.

(iii) Define the data base needed by these algorithms.

(iv) Specify the inputs required from the user (or file etc.) to provide the data for this data base. The interface between the input source and the program must be defined clearly.

(v) Repeat the above steps several times until a clean design is achieved. It may be necessary to implement some things to check concepts.

(vi) Design program test procedures and test data sets for each section.

4. Program Design - The distinction between program design and detailed design is often fuzzy particularly if pseudo code is used as a design tool. In many cases there is a direct correspondence between the two. However in some systems, timing and language constraints result in a program design significantly different to the system design.

(i) Design the data base.

(ii) Split the program(s) into logical sections paying particular attention to data flow.

(iii) Design the main line.

(iv) Define and design common procedures.

(v) Design major procedures and sub-procedures

(vi) Draw a diagram to show the interconnection of all procedures.

5. Code and Debug Mainline - the main line is written and tested using a set of dummy procedures. It can then be used as a test bed for the rest of the code. As each procedure is written it can be plugged into the mainline for testing.

6. Coding and Debugging - The rest of the code is written, tested and debugged, using the previously defined test sets in the following order:

(i) Data base.

(ii) Common procedures as they occur.

(iii) Output procedures.

(iv) Calculation algorithms.

(v) Input procedures.

Finally the whole system is checked to see if it meets the design goals and produces the desired results.

5. Observations

At the completion of the project the staff members involved were interviewed in an attempt to measure the success of project management by checkpoint control. The following observations

were made:

(i) If a student has not written a similar program before then he may need to experiment with a simple implementation to get a feel for the complexity of the problem before launching into the design.

(ii) At the start of the project many students were obviously not aware of the importance of system design. They considered coding to be more important. Many students found thinking about system concepts difficult and tended to think in terms of code. As a result they confused detailed design and program design; often overlooking vital applications details.

(iii) The structured approach of having specific project stages and regular checkpoint meetings identified a clear path for the students to follow and consequently they had a better idea about what to do and how to do it.

(iv) Responsibility for completing the project was placed on the students in a more organised way and as a result many learned how to organise their own time.

(v) Time spent by staff supervising projects was 0.5 to 1 hour per week per student irrespective of the type of project. It took roughly the same amount of time to manage several students doing the same project as several students doing a joint project or several students doing different projects. Supervision time tended to be a function of the complexity of the project and the quality of the student.

(vi) Several students rebelled against any form of rigorous design discipline and tried to design by coding. After a few weeks work they had burnt their fingers badly and wanted to start again in accordance with the suggested project stages. They did not have time to do so and had to make their poor designs work. The major problem in each case was that the system had not been modularised cleanly.

(vii) Supervisors were able to detect students falling into traps and, when the students accepted the advice, help them to avoid them.
Traps included:

(a) Projects growing too large for the time available.

(b) Initial designs that were too complex and had too many options. Those who designed well were able to get a subset going to test the ideas before implementing all the extras.

Result Level	Number of Students at each level		Level variation from other computing to project		
	Other Computing	Project	no change	up	down
Fail	3	3	0	3	0
Pass	4	3	3	1	0
Credit	3	7	1	1	1
D	8	4	2	3	3
High D	5	6	3	0	2

Table 1. Distribution of scholastic ability in other computing subjects and the project for the twenty-three students who completed the course. The level variation shows the shift in student marks in the project relative to other computing subjects.

(c) Redesigning part way through to add a new "you beaut" feature.

(d) Getting bogged down at a particular stage and ignoring deadlines.

(e) Forgetting human engineering aspects and taking the implementation simple route.

(f) Getting wrapped up in a project and loosing track of time.

6. Results

The results in terms of student grades are shown in table 1. All the students who received high distinctions for the project readily accepted the guidelines and took responsibility for setting and sticking to their own goals. It could be argued that these students would have done well any way but the level variations show that those who failed the project had previously been high achievers and those who had previously been poor students improved their performance.

Examination of individual student results revealed that those whose performance dropped were either unable to discipline themselves to a rigorous design procedure or, acting contrary to their supervisors advice, fell into one of the traps mentioned above. In the main students who improved their performance were forced to work according to the guidelines by their supervisors.

7. Conclusion

Many students have remarked that they learned more from doing the software project than from any other course, making the heavy work load worth while. Some saw it as an essential prerequisite to obtaining employment. Some disputes arose over the final result because in most courses the result is directly proportional to the amount of work done where in the project the effectiveness of the work done is more significant.

Management of the projects by checkpoint control allowed the staff to keep in close contact with the progress of the projects without it consuming large amounts of time. Student creativity was encouraged and directed not frustrated.

Students, many of whom were not previously aware of the importance of system design, learned how to design and implement a significant piece of code and get it right the first time. Those who rebelled against the structured design rules suffered where those who willingly accepted checkpoint control did well. Project reports and seminar presentations enabled the development of communication skills.

8. Bibliography

1. Busenberg S.N., Wing C.T., An Academic Program Providing Realistic Training in Software Engineering, Communications of the ACM, Vol 22 No6, June 1979

2. Management Series, Electronic Design, Hayden, 1977-78

3. Orr K. T., "Structured System Development", Yourdan Press.