# A Life-Cycle Project for a Software Engineering Course

**Janet M. Drake**
**University of Northern Iowa**
**drake@cs.uni.edu**

## Abstract

We have developed a software application that is used for student assignments in an introductory software engineering course. The project, Video Club Management System (VCMS), is a moderate size application that is partially completed. VCMS has a specification, test cases, and code. Students use a maintenance model and run through the complete life cycle. They start by inspecting the specification, then they test the application, next they propose changes, then they make changes to both the specification and the code, and finally they test their modifications.

The maintenance approach allows students to see already existing artifact such as a specification and test cases. The students use templates to make change proposals and fault reports. At the end of the course, students understand the power and need for a specification. The instructor has learned that students do much better when they have sample artifacts to follow.

## Introduction

In a software engineering course students should gain an understanding of the software development life cycle. Ideally they should do this by participating in a full life cycle software project. The project should be large enough to require some architectural design and should be realistic. The students should feel that the project is usable rather than just an exercise. In addition, students should work in a team environment. The team should create the following products:

- Specification: Including appropriate models and using an accepted specification standard
- Design: Including models, algorithms, and pseudo-code
- Test cases: Including step by step instruction and expected output
- Test results: Fault reports
- Code

It is difficult to create a project from scratch. Students in a semester software engineering course have only 15 weeks to cover all the topics in software engineering. A software engineering course is not a project course. In a project course we assume that students already understand the software development life cycle and the analysis, design, and testing techniques taught in software engineering. Therefore, in a software engineering course we need a small project that does not require 100 hours effort per team member. If we have a project starting from scratch that requires only 30 hours effort, the project will be too small to give a real life cycle experience.

Another problem is the specification. If the team has to write a specification, it can easily be a 30 hours effort. Most often students have never seen a real specification. Sample specifications are not shown in software engineering textbooks simply because they are too long. Another reason may be that they are never really correct and an author would have to be very brave to publish a specification knowing that it could never be perfect.

Creating projects from scratch does not represent industry. Today about 80% of professional work is maintenance. Most of our students can expect to start working on a maintenance project. Maintenance projects are not trivial. In most of the software we use, the difference between version A and version B is major. Old faults have been fixed, the software runs on new platforms and under new operating systems, and, most importantly, the new version has increased functionality.

The biggest advantage of working on a maintenance project is that documentation, code, and the business process already exists. When a new employee starts on a maintenance project they start with something and do not suffer the "empty paper" syndrome. They can use the existing materials to learn the problem domain. The existing materials also serve as a template for their new work. Even if they do not know the computer language that was used on the project, they have plenty of examples.

## Approach

For my software engineering course I use a maintenance project. The project, Video Club Management System (VCMS), has an existing specification, test cases, templates for fault reports and Engineering Change Proposals (ECPs), and code.

## VCMS History & Architecture

VCMS was originally written in 1999 (version 1) and a major update was done in 2003 (version2). Both were done as senior projects.

For version 1, the student wrote the specification and learned both Visual Basic 6.0 and Access 97 to create the code. We visited a video store and interviewed the manager to get information on videos and rentals. The architecture has five basic parts (see figure 1). Manage Customer Info, Manage Distributor Info, and Manage Movie Info were completed in version 1. Very little testing was done.

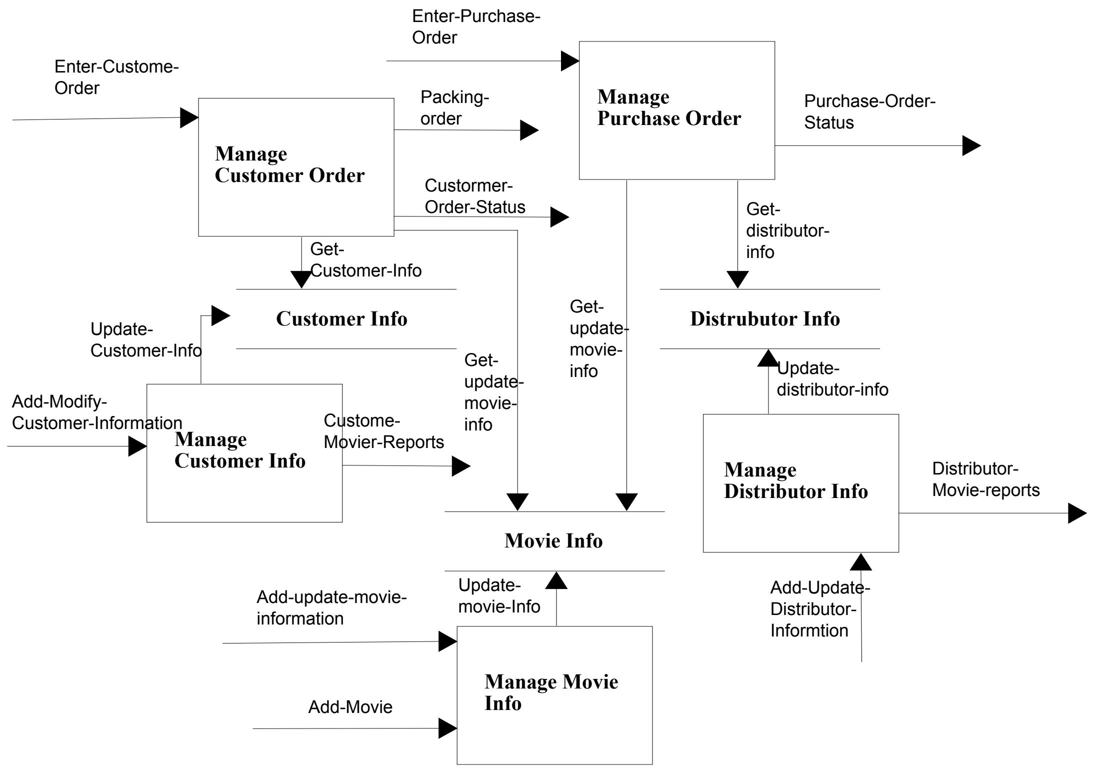

**Figure 1. Level 1 Data Flow Diagram of VCMS**

Version 2 retained the same general architecture. I rewrote the specification. This time Manage Customer Order and Manage Purchase Order were completed. Our CASE tool, Viable Analyst Workbench, generates schema for Access 2000 and Visual Basic 6.0 screens. These screens were used to stub in the remaining three modules. Again the testing effort was weak.

Next Version 2 of VCMS was used by a master's students in a software testing project. This resulted in a good set of test cases and a Version 2.2 of VCMS. In addition, the specification has been updated every time teams did an inspection.

## Student Teams

The software engineering students work in teams of 3 or 4. They begin working on the VCMS project in week 9 of the semester. They previously had 2 team assignments and have developed team relationships. There are three assignments in the maintenance project. As described previously, VCMS has five modules, two of which are completed. VCMS is too large for one team and is divided so that each team has one completed module and one incomplete module. There is some overlap between teams -- just enough so that teams can compare their efforts.

## VCMS Assignments

*Inspection:* At this point in the class the students have had a reading and lecture on software inspection. They are given the specification and the format for fault reports. They are instructed to read the specification before their team meeting and jot down the faults they find and any questions they have. When the team meets, they go over the specification and come to consensus on the faults. The resulting fault report is handed in.

*Testing and ECPs:* At this point in the class the students have studied structural (white box) and functional testing (black box). They have done an exercise on structural testing. VCMS provides an exercise on functional testing. The teams are given the code, an example of test cases, and a template for fault reports. The test cases examples were written by the master's student and provide a template for test case development along with good examples. The students create and run test cases for their portion of VCMS. They hand in a test document that includes a title page, an introductory page, the test cases, the fault reports, and a test summary report.

Through inspection and testing the teams have gained an understanding of VCMS. Now they must decide what changes they will make. Each change is written up as an Engineering Change Proposal (ECP). Each ECP describes the proposed change and time estimates for making the change. The changes are assigned to team members. The ECP can address both the faults found in testing and the uncompleted module of VCMS. Modification to the completed module can also be proposed. The ECP are presented in the order in which the team intends to complete them. The teams can present as many ECPs as they want. Presenting an ECP does not require them to complete the work. The ECPs are gathered into a document with a title page, an introductory page, and the ECPs are presented in the order in which they will be addressed.

Teams present their test results and ECPs to the class. Each team is given 15 to 20 minutes to present their results. Each team member is expected to participate in the presentation.

### Implement ECPs

While testing and creating ECPs the teams have developed a good understanding of their tasks. They have actually completed their analysis and have a good start on design. The next problem is implementation. VCMS is written in Visual Basic 6.0 and Access 2000. Not all the students are familiar with Visual Basic but usually one person on the team knows VB. The database is quite good and usually does not need modification. Sometimes a whole team will not know Visual Basic but VCMS provides many good examples. The students are familiar with several programming languages and can usually master Visual Basic by looking at the existing code and working together.

The teams retest their updated VCMS and update their original ECPs to reflect actual time and add any notes. The ECPs and notes are handed in. They also hand in any changes they have made to the specification. Again the teams present their final product. They demonstrate the changes they have made and discuss their experience.

## Advantages and disadvantages

*The students see a real specification*. They use the specification for the inspection exercise. They do not have access to the code and must depend entirely upon the specification. The specification is also used to create test cases. They learn that the specification is the cornerstone of software development and that the specification, no mater how well done, still has faults.

*The students participate in software inspection*. Before the meeting, each team member reads the material and makes notes. In the meeting the team discusses the faults they have found and create their fault list. The weakness of the exercise is that no producer is present. I believe this is the weakest part of the whole VCMS maintenance exercise.

*Examples of test cases improved testing.* Fall 2003 was the first time I passed out sample test cases examples and the resulting test cases were much better. From the sample the students saw the test case organization, layout, and granularity. Because of the example, the students knew what their product should look like and produced a better product.

*Test cases lacked detail input and expected values.* Somehow students have difficulty believing that the input values and the expected output values must be exact. They think that "Enter an address" and "Expect an address" is just as good as "Enter '1586 North Hamline Ave.'" and "Expect '1586 North Hamline Ave.'". Repeatability is not supported and it would be easy to accept any address or miss subtle changes.

*Any document is easier to both read and write when it is based on a template.* Besides the test case example, templates for the ECP and fault reports are given to the students. The template lets the teams know exactly what information should be included and puts the emphasis on solving the problem rather than designing and writing a document.

*Students work with a comparatively large program*. VCMS is not a large program but, when compared with code the students have previously written, VCMS is large. VCMS is large enough to show the need for a team and division of labor.

*Students see existing code.* Many students learn a little Visual Basic by following the existing code. On the best teams students work together on the code and all of them end up with knowledge of Visual Basic. They also learn that they can pick up a programming language quite easily.

*Weakness of the project*. The students have no customer contact. The domain knowledge for VCMS is from the developers and students. Our domain knowledge comes from renting videos rather than working in a video store.

Teams work on different parts of VCMS but there is no integration of the products. Changes to the code -- even though some teams make very nice modifications -- are not integrated into the product. The initial code is preserved for the next class.

*Version control:* We do not use a version control tool. Most teams use a group account but they get an introduction to the problems of configuration management. At least they leave the project with some appreciation of the configuration management problem.

**Lessons learned**

1. Teaching: Having existing artifacts is easier than making all new artifacts.

2. Architectural Lesson: Just like we should not have data redundancy, we should not have multiple screens in an application to modify the same data.

3. In a small, single user, data centered application like VCMS an ER model and STFs are adequate for analysis. When they are complete, they encompass design.

4. The SRS will never be perfect. I update the SRS based on each inspection and each new inspection yields faults.

5. The students fear a new language more than anything else. Sadly the students seem to get more reward from mastering a new language than anything else.

6. The teamwork experience is valuable. Most teams work together well. I estimate that half of the teams report that they have had a very good experience. Three of every eight teams find the experience acceptable and one of every eight teams has a problem. Usually the difficulty comes from one person who is unwilling to work and wants to take advantage of their team members. My advice to those teams is to work around that person and give that person a bad review at the end of the semester.

7. Teamwork takes time. Students must meet outside of class time and, between work, social life, and classes, students have a hard time meeting. Smaller teams make meeting easier.

## Conclusion

My students leave software engineering knowing that a specification is the key to software development. After working on VCMS they cannot imagine maintaining existing software without the specification. They understand that functional testing is testing to the specification and, finally, they have experienced a full software life cycle.

## Acknowledgement

I would like to acknowledge and thanks Gioa Vu who developed the original VCMS in 1999. Also thanks to Randy Klug who made version 2.0 in 2003. I also want to thank Zhibin (Eric) Wang who tested VCMS as part of his Master work. Randy and Eric worked together to produce version 2.2. Finally I want to thank all the Software Engineering students who have inspected, tested, and modified VCMS. Although the specification is not perfect, it is much improved from their efforts.