

Adventure Cycles - A Software Engineering Approach

John Paynter, Emma Sharkey

School of Business and Economics
University of Auckland
Private Bag 92019 Auckland, New Zealand

j.paynter@auckland.ac.nz; e.sharkey@auckland.ac.nz

Abstract

In this paper we discuss using a case study to demonstrate the software engineering process from requirements, specification, preliminary user manual, prototyping, design, implementation and testing as well as some post-implementation details such as maintenance and extendibility/reusability. Each semester a case is developed that is pivotal to a final year Software Engineering course. Groups, usually of four students, undertake the entire systems development life cycle. The project is developed over one semester (12 weeks) in four separate phases with an assignment completed for each phase.

The Adventure Cycles case was run differently from the cases used in previous semesters in that it was done during the three hours of lectures each week. These exercises replaced the standard (PowerPoint-driven) lectures. In parallel the groups developed a separate project (for 2004, in the first semester Dogs R Us and in the second Residential Tenancy). Examples from past semester projects are also made available to the students. The lessons learnt from these cases are available here. Overall the student performance in the class improved, as did their assessment of the course and its components. This has encouraged us to continue with this case-driven approach to teaching software engineering.

Keywords: Information System Case studies, Software Engineering, IEEE Standards.

1 Introduction

In the past the author had used case studies in the teaching of the introductory Information Systems course (Paynter and Ong, 1997). The case studies that were utilised followed the format suggested by Kroenke and Hatch (1993). This format requires students to ascertain why an information system is required in the first instance, along with the problems the information system is expected to solve. Students are also required to

identify the components of the Information System, and how the Information System has been implemented. These may be referred to as non-project based cases (Cappel and Schwager, 2002) and are mostly used to facilitate discussion. More latterly project-based cases have been used in assignments, mid-semester tests and final exams, with the students developing solutions for specific exercises (e.g. drawing an ERD). All the cases used were based upon real-life situations that have been experienced by actual organisations.

1.1 Teaching Information Systems using Case Studies

Case-based teaching is based on the opinion that learning will occur when an individual teaches himself or herself by working on their own problems. It is believed that the individual will obtain a greater understanding of the subject, and have increased levels of judgement through actively working through an issue, as presented in a case study, as opposed to just listening to subject material in a lecture setting. In addition, the use of a group project simulates actual work based settings and introduces real life problems such as can occur when working in teams.

1.2 Advantages of this approach

There are a number of advantages in the use of case studies. The use of cases:

- 1- Exposes students to issues and problems that are experienced in 'real life'
- 2- Allows students to apply any theory discussed in lectures to a real life setting
- 3- Assists in enhancing analytical skills of students
- 4- Develops skills in understanding what information is important, and what additional information is actually required.
- 5- Identifies any values you hold, and any assumptions that you may use.
- 6- Assists in enhancing individual communication skills – i.e. communicating ideas with respect to a case in a small tutorial group.
- 7- Exposes students to the viewpoints of others on a given issue
- 8- Assists in the development of concise, reasonable and consistent solution to a given problem.

In addition a number of these skills are considered valuable when students commence employment (Cappel and Schwager, 2002).

1.3 Disadvantages of using this approach

- 1- Students, due to a lack of exposure to such a teaching method, will be uncomfortable with the open-endedness of such an approach.
- 2- A great deal of care needs to be exercised during the construction of a case.

2 Teaching Software Engineering

The course stresses the need to consider the most appropriate approach to the SDLC to take for each case using a strategy of looking at alternative and complementary methods. The themes of utilising standards (e.g. IEEE) where appropriate and maintaining traceability and visibility throughout the project are promoted. The project scope ignores the commercial justification of the project, as that is deliberately omitted from the software engineering course, as it is considered to be covered in Systems Analysis classes.

2.1 Course Structure

The present format of the Software Engineering course was adopted along with the Schach (2002) Software Engineering text. In the first semester that the text was adopted the Air Gourmet case (present in the earlier 4th Edition) that was incorporated with it was used. In the following semesters cases were built based on actual or hypothetical organisations. Often these cases reflected topical events or businesses with which the instructor had some contact. Cases included: Americas Cup (timed for the 2000 defence in Auckland), Kia Ora Productions (as the Zena serial was largely set in the environment around Auckland), Norsand (a sand supply operation in Auckland), and in the year under discussion, Dogs R Us (a hypothetical pet shop) and Residential Tenancy (second semester group assignment case). Dogs R Us was to represent a retail chain selling puppies, accessories and dog services (e.g. grooming, walking, sitting), and was the basis for the first semester group assignment.

Previously feedback from the students indicated that they felt that there was insufficient link between the theoretical material delivered in three hours (in one two-hour and one one-hour slot) of lectures per week (largely based on the PowerPoint slides developed for the course text) and the project case (four, 5% assignments). In 2002 what was to become the transition to the present format was begun. Exercises were developed to supplement the lectures and developed on-line during an optional weekly workshop that followed the one-hour lecture slot, as the lecture room was free at that time.

The other course components are: weekly tutorials (10%) that go over lecture topics and help prepare for the assignment, and weekly quizzes (10%) run on the computer assisted learning environment (Cecil) that test lecture coverage. The students can also practice the multi-choice questions based on the text at the publisher's

(McGraw-Hill) site. The remaining assessment is made up of a mid-semester test (10%) and final examination (50%). These examined components are open-book and are based on the assignments and exercises developed in the year. A new case is developed for each test/exam.

2.2 Systems Project

Many IT majors offer a project course as a capstone for their offerings. Clear et al (2003) offer some guidance on the different ways such courses can be analysed. The paper outlines the availability of resources for such courses on the web.

The capstone course for Information Systems majors is the double-semester Systems Project course undertaken by groups, each assigned to a different business. Many of the students who had done the Software Engineering course, or whose friends had, asked to use the templates developed in the course. They found that they did not receive sufficient guidance and support for their major project.

3 Case Studies

Figure 1 illustrates potential research strategies. The figure refers to using case studies for the purposes of research, as opposed to the purposes of teaching. The three points (indicated by the letters A, B and C) refer to the maximal points for what is referred to as being the 'research hallmarks' of generalisability, precision and realism. The above figure indicates that it is not possible to concurrently maximise all three research aspects. If a researcher decides to maximise one of these aspects, then they can do so at the expense of the other two. For example, using case studies as a mechanism for conducting research will maximise the amount of realism within the study, but this will be at the expense of research generalisability and precision. This may be counteracted for instance by conducting surveys, as illustrated in Figure 1.

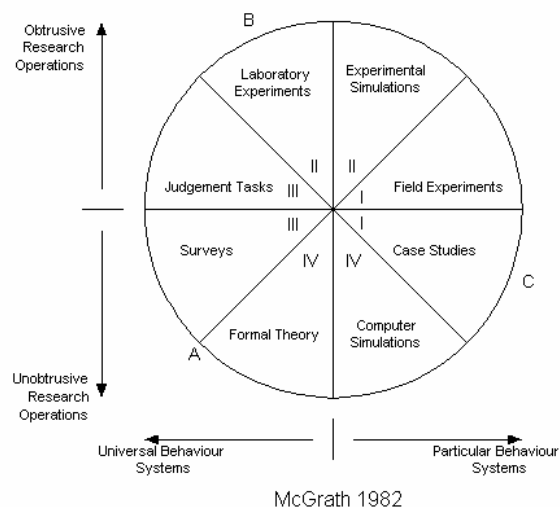


Figure 1: Research Methods (Source: McGrath et al, 1982)

The same conundrum exists when choosing a teaching method, hence the use of the weekly quizzes to ensure generalisability and precision.

4 Adventure Cycles

Adventure Cycles is a small business located in the Auckland downtown area of the CBD. It operates out of a run-down temporary premise that is adjacent to a large downtown renewal project. It is difficult to access with little adjacent parking on a major traffic route and the set up of the rabbit warren building makes the storage of bikes, parts and accessories difficult to track. Adventure Cycles is part of the Auckland (New Zealand) adventure network <http://www.adventure-auckland.co.nz>. The Auckland Adventure Operators Group is an organisation of Tourism and Adventure Companies who offer an exciting range of activities in and around Auckland. The adventures range from full-on, adrenalin-pumping excitement, to sedate, peaceful visits to some of Auckland's most tranquil and beautiful bush, harbour and beach destinations.

Adventure Cycles sells, hires and repairs bikes as well as running bike maintenance classes. It specialises in mountain bikes, touring bikes and student road racing bikes, stocking a huge range of new and second hand bikes and accessories -everything from top mountain bikes to children's bike bells. They have tandems, trailers and child safety items. In addition they have guaranteed buy-backs, great for tourists. Adventure Cycles sells bikes and pannier bags, as well as having a large selection of rental gear.

Adventure Cycles is typical of many small businesses in that it cannot afford the large sums of money and time to set up complex information systems, yet its need for information is wide ranging and complex.acerbating this problem is the fact that for the most part the employees are young and/or recent immigrants wishing to gain business experience. The owner manager then is tied up providing the hands-on day-to-day running of an operation that involves sales, rentals, repairs and maintenance classes as well as building bicycles from parts.

The author was aware of Adventure Cycles and has purchased bicycles there on two occasions and had them repaired on others. He had also seen the company over the years at the annual Bike to Work Day. Further information about the Adventure Cycles operation was gleaned from the web site. Examples of this included the hire details and the fact that they hold maintenance classes.

The author sent the case (9 pages including photos of the premises) and exercises to Bruce O'Halloran, the owner, for comment: "This is a most interesting and comprehensive exercise which works well with students who relate to a bicycle shop and therefore can get into the subject with more enthusiasm, tenacity and thoroughness." In addition Bruce added "You cover many of the things we now do on an ad-hoc basis without formal computer systems" He was impressed with the initiative of some of the students "One of your students

has been in and already offered consulting services to us. I like that kind of ambition!"

4.1 Teaching Suggestions.

The case was used in different components of the course. Exercises were developed during the lectures, and midway through the course Bruce was invited to discuss the case and to answer any questions from the students. Finally a different aspect of the case (repairs) from that used during the lecture workshops (hires) was used in the final exam. In this way, students were not new to the case, but were asked to solve a different set of problems.

4.1.1 Weekly lecture workshops

The normal flow is based on the specification of the four assignments. In addition some exercises (e.g. creation of a CRUD matrix) might be done in the lectures or tutorials. For instance in preparing the Software Requirements Specification (SRS) examples of poorly written requirements were shown. The students were asked to say why these were faulty and to rewrite them. For the assignment itself they would be expected to write them.

In assignment 1 there are two main deliverables. They are the Software Project Management Plan (SPMP) and the Software Requirements Specifications (SRS).

In assignment 2, there are again two main deliverables. They are the specifications written in UML and a Preliminary User Manual. The UML diagrams to be prepared include: Class Diagram, Use Cases as shown in Appendix I (and associated Use Case Template), State Transition Diagrams and Sequence Diagrams.

In assignment 3, the students are expected to present a prototype of the system. Additional questions may include ones on maintenance, reuse or software evaluation depending on the exact nature of the case and how it lends itself to each approach.

In assignment 4, the final deliverables are a Software Test Plan and systems manual.

For each assignment the students are also expected to produce product, process and quality metrics and to update the SPMP as required. In addition the assignments must show the traceability from SRS to User Manual, Design and Test by developing matrices allowing the (individually numbered) requirements to be tracked through to the final product.

4.1.2 Proposed Solutions

Developing a systems solution for a project-based case requires considerable effort and the solution is significantly broader than for non project-based cases (Cappel and Schwager, 2002). They are generally developed in response to the system requirements and entail creating artefacts such as diagrams, charts, models, documentation, prototypes and indeed an entire system. An example is a system analysis and design case published by Chappel (2002) in *JISE*.

A subset of solutions using different methods are developed in the software engineering course. They include from assignment 1: the scoping document, as most IT projects fail due to problems with the scope (Paynter et al, 2001); example SRSs (3 functional and 2 non-functional), and SPMP as these is the marking schedule for the mid-semester test and exam. From assignment 2 we have the class diagram of the main classes involved in the hire, a Use Case Diagram and associated template and from this a sequence diagram. No prototype was developed in the proposed solutions, but the students can refer to the Adventure Cycles web site, constantly under development, to see some of the functionality that might be developed. Finally some test specifications are shown, cross-referenced back to the original SRS.

Each set of exercises is set in context. In the first we talk of planning. Planning is important for all software development projects. Scheduling is always done as part of the planning process. Scheduling is the process of deciding in what sequence a set of activities will be performed, as well as when they should start and be completed (Lethbridge and Lagniere, 2001). A well-organised plan is important because this enhances the tracking process, which determines how well you are sticking to the cost estimate and schedule.

In the class an example Gantt is created at part of the Software Project Maintenance Plan (SPMP). It is a compromise between the plan for the real case and the needs of the class to meet the (assignment) deadlines with their deliverables (Table 1).

4.1.3 Guest appearance

Bruce was then invited to talk to the Software Engineering class and the students urged to prepare for his visit "Bruce O'Halloran from Adventure Cycles will be present at the lecture. Be prepared to ask any questions of him in your role as analyst. The exercises that we have done in class are all in Cecil. You may care to refresh your memory of them in preparing questions for Bruce. The mid-semester test that you will sit on your return will be largely based on these exercises."

Bruce used to teach "Accounting for Engineers" a Civil Engineering 3rd pro elective. As he stated, "I always found when lecturing at Auckland University in past years that posing questions relevant to students which they could relate to provided much more spirited and insightful answers to the questions. These are always much more interesting to mark."

When the owner visited the class we broke from the normal scheme of things in which we simply explored the technical aspects of the case. Bruce discussed the rationale behind his organisation and his philosophy. The software engineering class is not designed as a "Systems Analysis" class or a "Management of Information Systems" course. However if it was non-project based, it would be easy to write questions on management issues to do with the case. These could include the objectives of computerising the business, package versus custom

solutions and the unique problems that might be caused by the staffing issues in the case.

4.1.4 Final exam

Due to the limited time available in most classes it is stressed that the students should concentrate on the major aspects of the case. Only model those aspects that are specific to the case (e.g. do not model log in scripts) and those that exhibit specific behaviour. The students are told that in doing this they should use the technique of showing the "I"s and "T"s. That is, if doing a Class Responsibility Collaboration (Wirfs Bock et al, 1990) all the classes should be shown in an inheritance hierarchy, with only the key classes depicted using CRC cards. This is an example of a "T". An example of an "I" would be writing the SRS (showing breadth), developing the main specifications in a prototype (depth) then the test specifications (against the SRS) forming the base of the "I". They were told to prepare for the final exam with the same principals in mind. The exam case included a connection diagram from which it was possible to perform product estimates and complexity metrics and a bike repair sheet to aid in understanding the application as well as a one-page description of this aspect of the case.

5 Evaluation and Discussion

The response of the students to Bruce's appearance was outstanding. We were worried that the discussion would not be fruitful and would soon peter out. Instead it was a pity that we had to close it off after the one-hour class. Bruce had deemed that the exercises, developed for the class by the instructor, were the most part very realistic. Of more value though was Bruce's discussion as to how small businesses had to implement a mixture of manual and computerised systems on a shoestring. He also linked the Adventure Cycles domain to that of the parallel assignment (Dog R Us) in saying that much of the required functionality was similar. When we repeated this exercise with the second semester class he again drew a parallel. This time it was between evaluating the trustworthiness of potential bike hirers and would be residential tenants.

Figure 2 illustrates the response of the students to different course components. The tutorials had the most favourable mean Likert scores, with the class exercises being ranked a close second. The sample size in the survey conducted was 27.

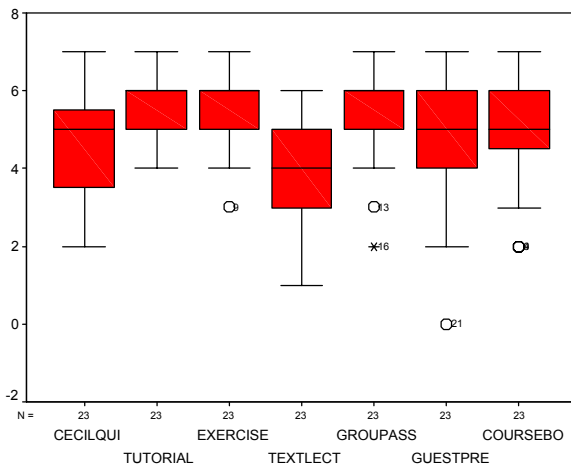


Figure 2: Likert scales¹ of the course components

The approach used in the teaching seemed to pay off in the semester. Attendance was higher than usual with the students maintaining interest throughout, although it was not so high towards the end of the second semester. This was reflected in the final exam with a 100% pass rate in both semesters compared to 72-85% in previous semesters. The independent teaching evaluation conducted by the University also demonstrated that the course achieved higher ratings than comparable courses (Figure 3).

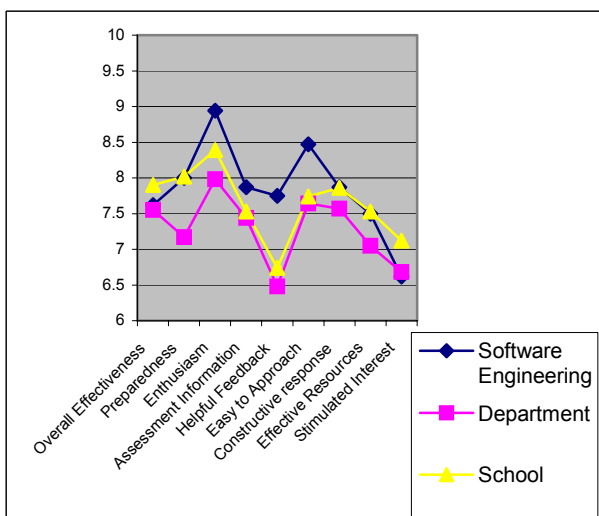


Figure 3: Course evaluation

The only area judged not to be above average was in objectives. It is likely that this is a problem of taking a case-based approach, where each exercise has multiple objectives and there is a large degree of overlap.

Post-graduation too the Software Engineering course received good evaluations, albeit anecdotal. E.g. students reported back that it was the most relevant course as far as their jobs were concerned. Several past students

¹ The scale shown is 7 extremely valuable, to 1 not very valuable at all and 0, no participation by the student. Median, quartiles, ranges and outliers are shown.

actively recruited from the following semesters' classes or reported that they got their jobs on the basis of what they learnt in the course.

The workload in preparing the exercises was higher for the instructor than in using pre-packaged presentations. The case was refreshed with new examples in the second semester. In addition aspects of the cases prepared for the mid-semester tests were further explored in the second half of the course. Not every case can cover all aspects in the system lifecycle. It was found for instance that the case used for the semester 2 test (see www.virtualnpc.co.nz) was excellent for teaching testing (Paynter and Sharkey, 2003).

6 Conclusions

The case-based approach seems to be well received by students. This is apparent in the in course and external course evaluations and in the results achieved by the students in the final examination. On this basis we are persevering with this approach and are using this case for teaching in the current semester along with a new case developed for the class project. Exercises are performed using the Adventure Cycles case and solutions provided. The instructor then primes the class to begin the solution for the assignment case at the end of each segment of instruction.

7 References

Auckland Adventure Operations Group, Auckland New Zealand. <<http://www.adventure-auckland.co.nz/>>. Accessed April 2003.

Cappel, J. A Systems Analysis and Design Case. *Journal of Information Systems Education* 2002, 12(4) 233-243.

Cappel, J. and P. Schwager. Writing IS Teaching Cases: Guidelines for JISE Submission. *Journal of Information Systems Education* 2002, 13(4) 287-293.

Cecil. <www.cecil.edu>. Accessed April 2003.

Clear, T., Young, F., Goldweber, M., Leidig, P., & Scott, K. Resources for Instructors of Capstone Courses in Computing *ITiCSE 2001 Working Group Reports - SIGCSE Bulletin* (2001), 33, 93-113.

Fowler, Martin. *UML distilled: a brief guide to the standard object modeling language*. Reading, Mass. Addison Wesley, 2000.

Institute of Electrical and Electronics Engineers. *IEEE standards software engineering*. New York, NY: Institute of Electrical and Electronics Engineers, 1999.

Kroenke, D and R. Hatch Management Information Systems McGraw-Hill, 1993.

Lethbridge, T and Laganier, R. *Object-Oriented Software Engineering*, McGraw Hill, 2001.

McGrath, J.E., Martin, J. and Kulka, R.A. *Judgement Calls in Research*, Sage Publications, New York 1982.

Management Science and Information Systems, *Software Engineering course book*, University of Auckland, 2003.

Paynter, J., M.D. Ahmed, A.Everett, and V. Llanes. An Analysis of Software Development Project Management Failure: Two New Zealand Cases. In *Business Case Studies in Operations Management*, ed by Tom Batley, Pearson Education, Auckland, NZ, 2001: 152-161.

Paynter, J. and J. Ong, 'Pan Pacific Cases in Information Systems', In: *Restrategising the Asia-Pacific Region Towards a New Millennium*. 14th Pan Pacific Conference. 3-5 June 1997. Kuala Lumpur, Malaysia, 145-147

Paynter, J. and E. Sharkey, 'Visualising Software Algorithms to demonstrate complexity and testing',

Proceedings of the 38th Annual Conference of the Operational Research Society of New Zealand, Waikato, Nov 21-22 2003, forthcoming.

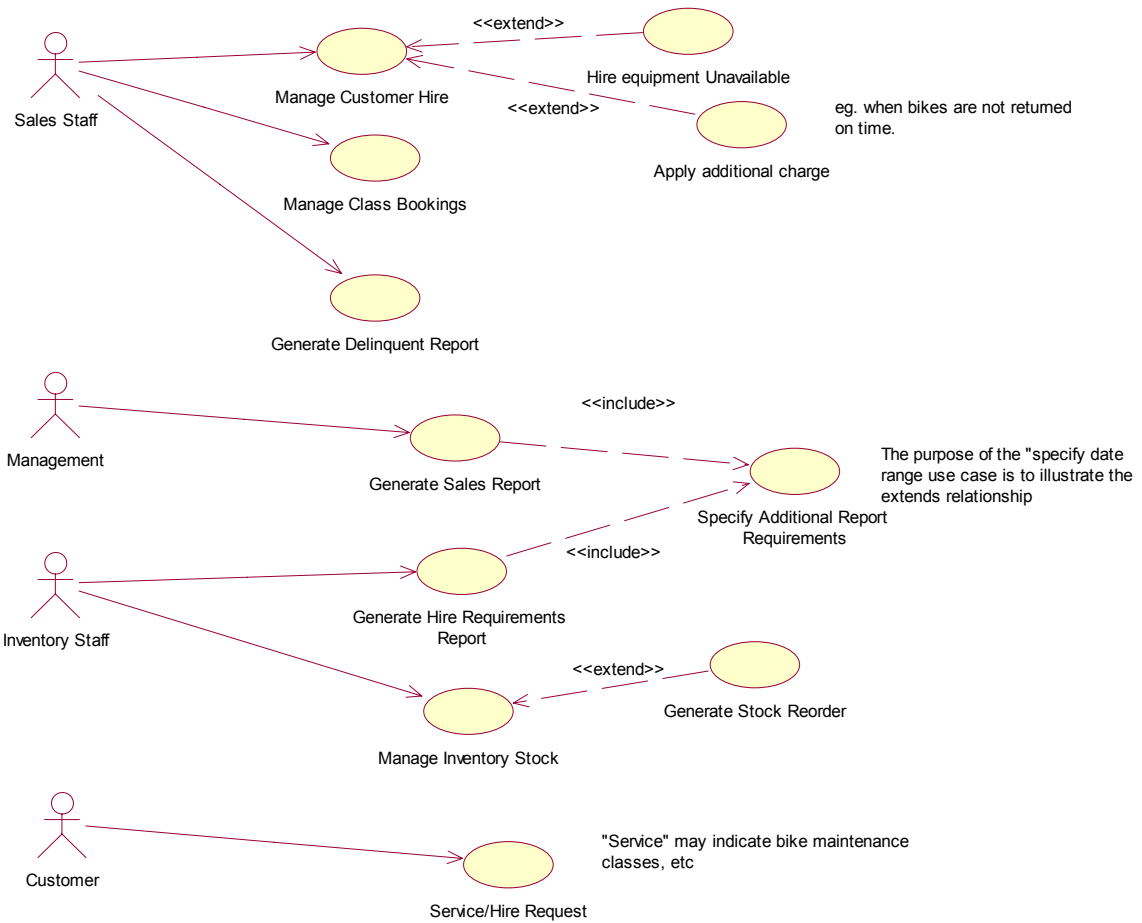
Schach, Stephen. *Object Oriented and Classical Software Engineering*. McGraw-Hill. 5th Edition, 2002.

The McGraw-Hill Company Website. <http://www.mhhe.com/engcs/compsci/schach5/quizzes.mhtml> Accessed April 2003.

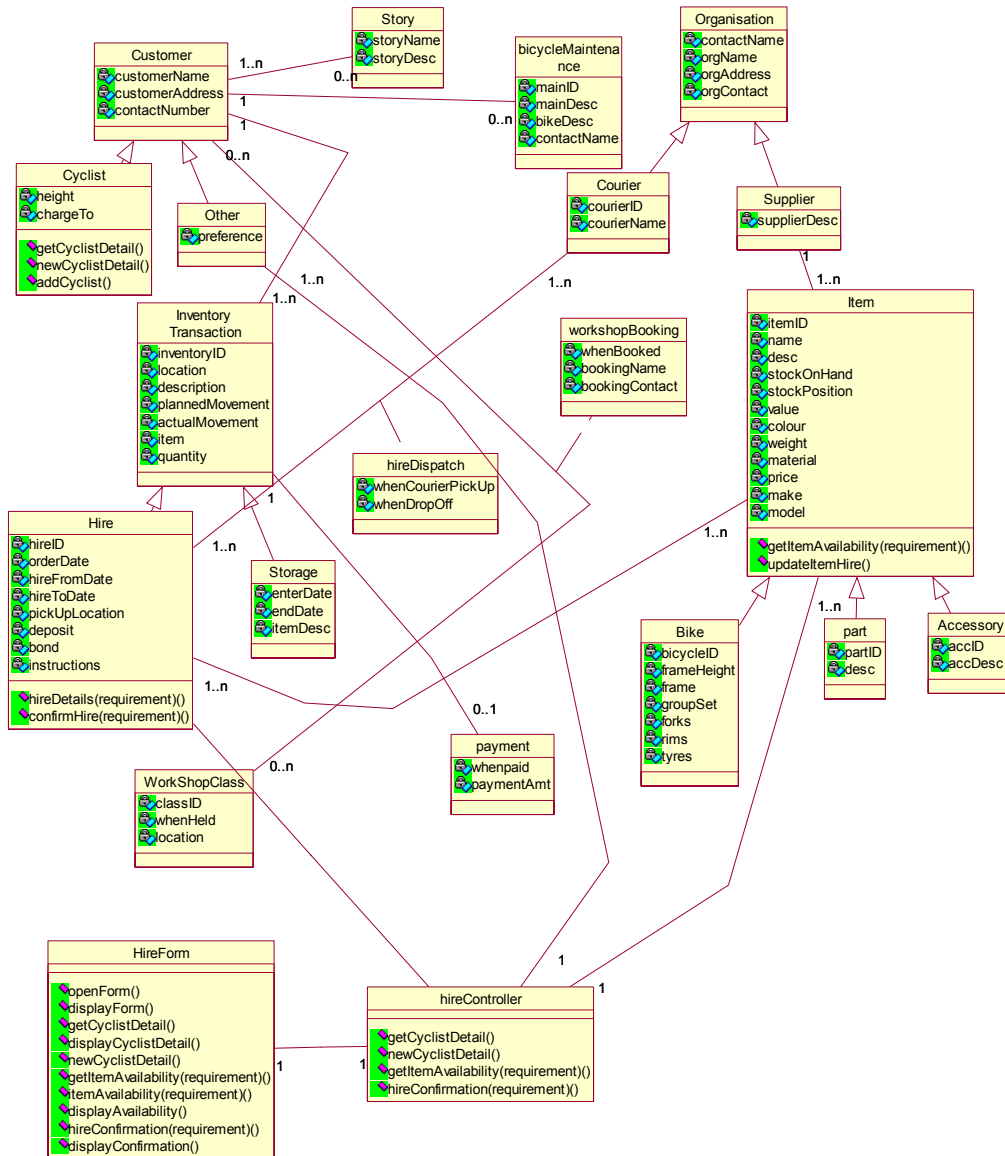
Wirfs-Brock, Rebecca *Designing object-oriented software*, Englewood Cliffs, N.J. Prentice Hall, 1990.

	Tasks	Duration (in days)	Start	Finish	Predecessors
1	Document existing system	8	3 Mar 03	12 Mar 03	
2	Write scope document & agree	1	13 Mar 03	13 Mar 03	1
3	Prepare project plan	1	14 Mar 03	14 Mar 03	2
4	Write requirements document & accept	5	17 Mar 03	21 mar 03	2
5	Write specifications	10	24 Mar 03	4 April 03	4
6	Prepare preliminary user manual	5	7 April 03	11 April 03	4
This is the break, so I will pretend that the team is having a holiday or working on something else.					
7	Design database	2	28 April 03	30 April 03	5
8	Prototype web interface	1	2 May 03	2 May 03	2
9	Design Hire System	5	5 May 03	9 May 03	7
10	Install inventory system	5	5 May 03	9 May 03	4
11	Implement Hire System	5	12 May 03	16 May 03	10
12	Integrate Hire System with Inventory	5	19 May 03	23 May 03	11
13	Write test specifications	5	19 May 03	23 May 03	4
14	Execute integration testing	5	26 May 03	30 May 03	13
15	Parallel run	12	3 Jun 03	17 Jun 03	14
16	Conduct post-implementation review	2	23 Jun 03	24 Jun 03	15
17	Update plan for next stage	1	30 Jun 03	30 Jun 03	16

Table 1: Gantt Chart of Adventure Cycles project adapted to fit a one-semester 12-week course.



Appendix I: Use Case for Adventure Cycles



Appendix II: Class Diagram for Adventure Cycles