A Client/Server Case Study for Software Engineering Students

Shawn A. Butler, Ph.D. Student
Computer Science Department
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, Pennsylvania 15217
Shawnb@cs.cmu.edu

A goal of the Studio course in the Master of Software Engineering program at Carnegie Mellon University is to bridge the gap between experience and academics. One way to transfer experience to young software engineers is through case studies designed to focus students on specific software engineering problems. This paper discusses my experience with developing a case study to improve a student's analytical capabilities and introduce the importance of considering maintenance and implementation issues in software design. The case study, developed as a classroom assignment, proved an effective tool to teach software engineering students that there are more things to consider than performance specifications.

1: Introduction

Business schools have been using case studies for years to develop a student's analytical abilities, but they are rarely seen in software engineering courses. Case studies can also be used to develop the analytical abilities of software engineering students. Furthermore, case studies can help bridge the gap between the experienced software engineer and the inexperienced student who has difficulty applying what he or she has learned in the classroom. A carefully designed case study can focus the students on specific software development problems. This paper discusses my experience with developing and using a case study in the Software Architecture course for Master of Software Engineering [1] (MSE) program at Carnegie Mellon University. I used the case study to show that engineering analysis of software systems in more than knowing whether the system meets performance requirements and to teach students about middleware design. The case study gave me an opportunity to share my experiences of developing client/server systems with students who were just learning about client/server architectures. I also took advantage of the case study to bridge the classroom lectures with the studio experience. The *Case Study Summary and Analysis* section provides the background, a very brief summary of the case study, and design of the case study. It concludes with some observations and student feedback about using the case

[1] The MSE studio course is a yearlong project with real clients that take students through the complete software development life cycle.

study. The *Client Server Case Study* section is the case study and assignment as given to the students. Although the setting is fictional, the system presented in the case study is similar to real systems with which I have been involved.

2: Case Study Summary and Analysis

2.1: Background

One of the most difficult challenges for senior software engineers is to help young software engineers bridge the gap between academic training and experience. Many aspects of software engineering are best learned in an applied environment. Software engineering is such a multi-disciplined field that it is difficult to teach students all aspects. For example, students with little or no experience usually don't consider maintainability and implementation constraints when evaluating technically feasible alternatives. Experienced software engineers appreciate how those constraints affect the final outcome of a large software project and give them due consideration. Designed to help students focus on software system properties, the case study provided an opportunity to integrate two classes, the MSE Studio and the Software Architecture class.

2.2: Academic Context

2.2.1: MSE Studio

CMU's studio program is designed to allow students to work with experienced mentors on a real project applying many of the principles learned through their course work. As a mentor I quickly realized that students had difficulty performing engineering analyses that included more than performance considerations. One reason that students concentrate on performance requirements is that these requirements are usually quantifiable, therefore testable. It is much more difficult to assess the maintainability of a system. Issues of maintainability, implementation, and openess are important design criteria that students often overlook. The case study provided a situation where these criteria posed significant design questions.

2.2.2: Architecture Class

Students in the Master of Software Engineering program at Carnegie Mellon University are required to take the Software Architecture course. In this course, lectures are presented on several different types of architectures, one of which is client/server architecture. Pipe and Filter, Batch Sequential, Black Board are just a few of the other architectures covered. Almost all topics require students to read at least two background papers to have some some familiarity with the architecture prior to the lecture. Assignments are related to one of the lecture topics and are usually programming tasks that require implementation or modification of architectures. I presented the client/server lecture and developed the associated assignment. Despite the fact that each student is accepted into the program with at least two years work experience, most of the students were not familiar with client/server architectures.

The lecture's readings, *"Essential Client/Server Survival Guide"*[1] and *"Client Server Architecture"* [2], were specifically selected to familiarize the students with the basic concepts behind client/server architectures and to highlight particular design issues software engineers face when developing such architectures. During the lecture I pointed out differences in middleware products such as message-oriented middleware, CORBA, and transaction processing software. In order to prepare the students for the case study analysis, I shared some of my past experiences in designing and developing client/server systems.

2.3: Case Study Learning Objectives

2.3.1: Goals

The primary goal of using a case study was to focus students on implementation and maintenance factors in addition to performance requirements when designing and selecting components for a client/server architecture. A secondary goal of using the case study was to test its usefulness as a mechanism to develop a student's software engineering skills. The assignment was designed to encourage students to focus on implementation and maintenance issues. To achieve this focus, highly specific performance requirements were not stated, and a number of alternatives could have met the system non-functional requirements. The final learning objective was to provide students with some experience in doing the engineering analysis and making a recommendation based on that analysis.

2.3.2: Case Study Rationale

Properly constructed case studies, which eliminate distracting details, allow students to focus on specific software engineering problems. I believe they are particularly useful as vehicles which allow senior software engineers to transfer lessons learned to others who are less experienced. The case study gives the student experience with a simplified, but realistic problem that might be encountered after graduation.
As a mentor for the MSE studio program [2], I find that students have difficulty incorporating non- functional requirements into their engineering analysis of design alternatives. This may be caused by inexperience and lack of understanding of how skill availability, language, commercial software integration, product maturing, licensing, etc. affect the cost and schedule of a particular design. For example, an inexperienced designer may consider two designs equivalent because they both meet performance requirements, however, the designs could have very different maintenance costs. Experienced software engineers understand the significance of these factors and select designs that achieve the desired balance. Often in software development, it is difficult to determine non-functional requirements and even more difficult to rank their importance. The case study was designed to direct students' attention to a few of the factors that a software engineer might realistically encounter in developing a client/server system. Middleware selection is an excellent mechanism for teaching future software engineers about engineering analyses, since it is often the heart of a system. Software Engineers have many standards and products from which to choose, and middleware design can be one of his/her most difficult challenges. When more than one technically feasible solution exists, selecting the best design is often a function of weighing non- functional system attributes, such as maintainability, or system development risks.

[2] Mentors provide advice and guidance on all aspects of the project

To ensure that students understood the emphasis of the case study, maintenance and implementation considerations were discussed during the lecture so students had some exposure to them prior to completing their analysis. Also, the assignment specifically asked the students to consider maintenance and implementation in the comparison of alternatives.

2.4: Case Study Summary

The following is a summary of the case study given to MSE students as a homework assignment. The case study is divided into three sections: 1) A description of the system and high level requirements, 2) three middleware alternatives under consideration, and 3) assignment instructions. Students were asked to compare the three alternatives based on the information contained in the case study.

The first section of the case study described a geographically distributed system requiring high reliability for some transactions but not for all information flow. Additional system requirements included capability for distributing high volumes of non-persistent broadcast data, adequate security and data integrity, sufficient back-up and recovery, and a heterogeneous computing environment. Finally, the assignment required that the designed system should be scalable and extendable.

The second section of the case study discussed three design alternatives and the student's task was to compare and select from these three design alternatives. All alternatives were essentially the same except for the middleware component. The student was also given the option of creating a different solution as a hybrid of the alternatives. The first alternative relied on the functionality commonly found in commercial database management systems to distribute information throughout the network. The second alternative took advantage of the latest, but less mature, Common Object Request Broker Architecture (CORBA) products to distribute information. The third alternative used a messaging and queuing product as the middleware. Each alternative had different strengths and weaknesses that were not obvious in the case study narrative.

The final section of the case study offered students suggestions on how they should approach their analysis. Previous assignments in this course showed that the students greatly benefited from suggestions that helped focus their efforts in the right direction. From experience I knew that the students would find it very difficult, if not impossible, to find pricing information. In class I emphasized that the assignments would be graded on how well the student supported the recommendation and not on selecting the "best" solution, since there really wasn't a "best" solution.

Some non-functional requirements were explicitly stated in the case study while others could be easily inferred. These requirements were prioritized so it would be easier for the student to compare alternatives. In addition, issues related to legacy hardware and network capacity were eliminated to provide a simpler problem and allow the students to concentrate on the architecture of the middleware rather than the underlying hardware.

Students were encouraged to ask questions about the system especially if they found themselves floundering in any area. I estimated that each student should not spend more than 15 hours on the assignment. As a mentor I knew that many students would have difficulty at some point in the assignment might spend many hours pursuing irrelevant information. The time estimate did not allow for much wasted time, so I wanted them to ask questions early.

2.5: Outcome/Analysis of Assignment

Almost all the students recommended a hybrid design for the middleware architecture because they focused chiefly on performance requirements. Most notably, many students didn't consider the long term maintainability and implementation costs in their evaluations of alternatives. Many of the analyses used a matrix to list each system requirement and rate the alternatives relative to each other. The matrix provided a convenient way to compare alternatives, but the analysis was not complete unless it compared implementation and maintenance costs. Although specific cost data were not available, the students were capable of comparing relative costs for each alternative.

Students had trouble organizing data, and distinguishing between useful and non-useful information. Also, they had difficulty differentiating among different types of middleware, i.e., all products claimed to be able to meet the system requirements equally well. Most of the students seemed overwhelmed with the information they collected.

Some students complained of the imbalanced effort among group members or, that they had to do the research for each alternative or that their group never met. Other students said they spent many hours in group discussions without reaching any conclusion. In the end, most students claimed they spent 25-30 hours on the assignment rather than the 15 that I had predicted.

2.6: Student Feedback

I asked the students to provide feedback on the assignment since I was considering developing more case studies for software engineering students. The students felt the case study was an effective tool for teaching software engineering students about design tradeoffs in a client/server architecture. In follow-up discussions they said that the assignment helped them learn more about implementation and maintenance problems in software development. Several students included comments with the assignment. Some students asked for a follow-up session after the semester was over so they could get feedback on how I would have approached the case study and to give me feedback on how the assignment could be improved. Some students also provided feedback to me while their work was in progress. Their questions gave some insight into how they approached the engineering analysis. The following feedback and recommendations will be incorporated into the case study for future courses.

Several students suggested ways to improve the exercise. The most significant was the suggestion that the case study should be developed in two phases. In a two-phase approach, students should first synthesize the case study by developing a set of system requirements and an analysis plan. Once they provide feedback on the first phase they would then proceed to research and analyze the alternatives. The students felt that the feedback from the first phase would help them focus on the important and relevant information for each alternative.

Overall, the students felt that the assignment was very challenging and closely represented the types of problems they expected to encounter upon graduation. Although they cited several reasons for the difficulty of the assignment, the main reason seemed to be related to the amount of information available that seemed relevant to the assignment. Each alternative appeared indistinguishable because students focused on performance characteristics rather than implementation efforts. I believe, MSE mentors could have been particularly useful in helping students understand how customized software can lead to greater maintainability costs over the software system life cycle.

2.7: Analysis/Conclusion

Although the students didn't specifically request a solution for the case study, it would have been constructive to provide an example of an engineering analysis for middleware. Unfortunately, since the information contained in an engineering analysis can quickly become obsolete, sample analyses used for this assignment requires review (to determine currency) with each use of the case study. Product capabilities change very quickly. In lieu of a written engineering analysis, follow up to the assignment should include discussion of relevant points for inclusion in a good analysis.

As expected, dividing the alternatives within the group worked well in some cases and not as well in others. Unfortunately, only a few students took advantage of the faculty or mentors for additional help. Consulting with a mentor might have helped students use their time more effectively, but more importantly it would have been an opportunity for the students to consult with individuals with direct experience in the field. I believe future case studies should include greater emphasis on, even require, consultation with experienced developers.

Despite the fact that students didn't address maintainability and implementation issues, they nevertheless learned about software engineering and client/server architectures. Most importantly they gained some experience from doing the engineering analysis and making a recommendation based on that analysis.

Case studies may not be an appropriate teaching device for other architectural styles, such as pipe and filter designs, but they can certainly teach students about other software engineering challenges. In particular, software quality and requirements analysis might be good subjects for a case study. I believe that a well designed case study can challenge a student and better prepare him/her for the types of problems they are most likely to encounter when developing systems. It is a simple but effective tool for that allows software engineers to impart their experience and wisdom to inexperienced software engineers. Finally, this experience validated the use of the case study as teaching tool, student feedback provided suggestions for refinement that should be included and I suggest that such methods be used more often in software engineering curricula.

3: Client Server Case Study

3.1: Introduction

ACME Financial is a fast growing company that owes part of its growth to several recent acquisitions. ACME Financial now wants to consolidate the companies' information technology resources to eliminate redundancy and share information among the new companies. The Chief Information Officer (CIO) has oversight responsibility for the project and has hired Client/Servers R Us to develop the architecture for the new corporate information system. Joe Consultant of C/S R Us presented 3 client/server designs to the CIO and is requesting the CIO to select one. The CIO is not sure which middleware design is best for the company's goals. The CIO has asked Chris Consultant to present the advantages and disadvantages for each of the alternatives.

3.2: Background

ACME Financial Incorporated (AF Inc.) is an investment banking company that provides an on- line service that allows their clients to access account and market information. ACME Financial Inc. recently acquired several small and medium sized companies throughout the country, each with their own financial and accounting systems. Almost all of the companies have developed their own application software for their analysts' use in their daily jobs, but only a few provided on-line account service. The analytical tools rely on near-real time market data and historical market data. The CIO wants to consolidate the financial and accounting information into a corporate information system that can support decision support applications for corporate management. Naturally, since the computer hardware is different for different companies, the CIO expects to upgrade the hardware to accommodate the new Information Technology (IT) system. The CIO will select the best analytical software as the standard software used by all company analysts. Each local site will be expected to provide an on-line service for their customers. Customers will be given the necessary application software to access their account information. Finally, ACME Financial has developed special data mining software that gives them a competitive advantage. AF Inc. offers their customers investment advice based on the information derived by the data mining software. Each account manager receives the information and then provides tailored recommendations to each customer based on their portfolio.

3.3: System Requirements

The following list of system requirements reflects the system's relative priorities:

1. **Availability**: The CIO's number one priority is high availability. AF Inc. markets their reliability and feels that most clients choose them for their dependability. The CIO wants to maximize the system's availability. To achieve high availability, if a regional office cannot provide support then a customer must always have access to the on-line service through a different office.
2. **Data Integrity**: The requirement for data integrity varies within the system. The most important data are customer's transactions. It is essential that a customer's transaction is never lost and the system must guarantee that each transaction is completed. In contrast, data lost from the high data rate inputs, such as Reuter's and the NYSE, are easily recovered in subsequent broadcasts so it is not critical if some data are lost during a broadcast.
3. **Performance**: Financial markets are highly volatile; time sensitivity of data is measured in minutes. Millions can be lost if information is delayed getting to the analysts. The system must be able to support information broadcast throughout the network.
4. **Security**: The CIO is concerned about the security of the data mining software and the information produced by the data mining software. The Chief Executive Officer thinks the data mining information software provides a competitive advantage for the company. If an unauthorized user had access to the information they could steal the data mining applications or steal the information produced by the data mining software. In either case, the perpetrator could make the same investment recommendations as AF Inc. account managers. Therefore, if competitors had access to the information the results could be financially devastating to the company. The CIO is concerned that a competitor could pose as a customer and hack into the highly sensitive information through his on-line service account.

5. **Growth**: The CIO envisions an incremental migration process to install the new system due to the magnitude of the change. Also, he expects that AF Inc. will continue to grow and acquire more companies. The CIO wants to be able to develop more application software as new customer services are added. The CIO also wants to add more near-real time information sources to the system.
6. **Backup and Recovery**: The CIO understands that the system will encounter problems from time to time. A key factor in determining the system's success is how quickly the system can recover from a failure. Backup and recovery must be smooth and non-disruptive. One way to ensure that the system can easily recover from a system crash is to make sure the data is duplicated elsewhere on the system. The corporate database is the primary back up for each of the regional offices.
Each local office (Northeast, Northwest, Southeast, Southwest) has accesses a regional information hub. Local offices use client software to access the local application server. These application servers access the local databases for almost all of the information needed on a daily basis. For access to information needed less frequently the application software should access the central database at corporate headquarters. Each regional database has only the subset of information that is relevant for its area, whereas the corporate headquarters maintains all of the information from each region as well as data that is unique to corporate applications, such as additional accounting and company financial information. The corporate office is also responsible for the data mining software and information. Each of the regional databases is connected with high capacity links to the corporate database. Finally, the corporate office receives information from Reuter's, NYSE, NASDAQ, and other financial markets. The information flow fluctuates daily from 30 40 KBps to 4 5 MBps. Twenty-five percent of the information is immediately broadcast to the regional offices to support the on-line account service. All the information is filtered and stored in the database.

3.4: Architectural Alternatives

Alternative I: The Database Management System This alternative takes advantage of the extended functionality provided by the popular relational database management companies, such as Oracle and Sybase. All information is delivered into the system where it is immediately stored into one of the databases. The relational database management software is responsible for the distribution of information throughout the system. Clients communicate with the databases through Standard Query Language (SQL). Corporate and regional databases are kept synchronized using features supplied by the RDBMS software. Transactions are guaranteed by using special Transaction Processing Software. The vendor-supplied RDBMS software is responsible for back-up and recovery of all the databases. Data security is handled at the row level within each database. This means that clients can only receive records for which their user has permission. Existing application software may have to be modified to use SQL.
Alternative II: Common Object Request Broker Architecture (CORBA) This solution depends on CORBA to tie together the clients and databases. CORBA is responsible for distributing data across the system. The RDBMS software is still responsible for the back-up and recovery, but the databases are kept synchronized using CORBA as the primary transport mechanism for the data. Clients, application servers, and databases communicate to each other through CORBAs transport mechanism. Existing application software would

be wrapped in IDL to communicate with other applications. Special near-real time handling application software would send the information to each of the regional offices where it would be directed to clients that subscribe to the information.
Alternative III: Message and Queuing (M&Q) The message and queuing design uses commercial M & Q software combined with a transaction processing product to ensure customers transactions are completed. Dec Message Queue and MQ Series are some of the leading products for messaging and queuing software. Clients communicate to other entities using messages. Messages are deposited in queues and the message and queuing middleware is responsible for message distribution to the appropriate clients. The software applications will be modified to send and receive messages from queues.

3.5: Questions to Answer (Total 80 points)

As before you should discuss all of these questions with your teammates, but your final write-up should be yours alone. Doing research on specific products for the assignment should certainly be a team activity. The total length of the write-up should not exceed 5 pages.

1. Describe in more detail the architecture of each alternative. Some services are automatically provided when a product is purchased, others must be developed to satisfy the system requirements. You should describe what services are automatically provided by some of the products, which services would need to be developed, and how services should be distributed across the network. (30 points)
2. Evaluate each of the alternatives against the system requirements. Discuss the advantages and disadvantages of each of the alternatives. Assume that the hardware will support all solutions. In your analysis you might consider which alternative gives the system developers the most flexibility, which alternative provides easiest maintenance, and which alternative requires the least modification to the current system. (30 points) 3. Prioritize each alternative or suggest a different solution if you think it superior to the 3 presented alternatives. (20 points)

Suggestions on how to proceed:

1. There is not enough information to make an informed decision about each of the alternatives. As a team, allot a percentage of your time to discover which products offer what type of services. You do not have enough time to do a complete market survey so pick 2-3 products.
2. If you depend only on marketing information you may find that the alternatives are equivalent. So you might want to go beyond the market literature in doing your research for this assignment. 3. As you do your analysis, pay particular attention to some of the following kinds of issues:
3. How well does the architecture support the basic system functionality requirements? b. How much run time performance overhead does the architecture impose? c. How well will specific products handle the high volume of data? d. How will each architecture handle occasional peak loads? e. How easy is it to customize the system to new requirements?
4. In your analysis, do not consider the actual product cost. (It may be impossible to get actual product costs anyway, so do not waste time doing so.) Evaluate cost with respect

to the amount of customized software necessary to implement each alternative, long term maintenance, time to implement, flexibility, etc.
5. If you don't understand something, or think something is important that hasn't been mentioned, ask about it. My e-mail is shawnb@cs.cmu.edu . I will post clarifications on the class bulletin board.

References

[1] Orfali, Harkey, and Edwards. *The Essential Client Server Survival Guide, 2nd edition.* Van Nostrand Reihold. Chapter 10

[2] Alex Berson. Designing Distributed Data Management Systems. *Client Server Architecture* McGraw-Hill. Chapter 2