December 2006

An Empirical Investigation of a General System Development Model

Dan Turk
Colorado State University

Leo Vijayasarathy
Colorado State University

Jon Clark
Colorado State University

An Empirical Investigation of Models Versus Prototypes in Systems Development

Dan Turk
Colorado State University
dan.turk@colostate.edu

Leo Vijayasarathy
Colorado State University
leo.vijayasarathy@colostate.edu

Jon Clark
Colorado State University
jon.clark@colostate.edu

ABSTRACT

It is generally accepted that models are useful in systems development and a variety of modeling tools and techniques are taught in information technology academic programs and used by software professionals. However, the recent emergence of agile systems development approaches, including extreme programming, SCRUM, and pair programming is challenging the value of modeling. Surprisingly, other than anecdotal evidence, there is little empirical data to support or refute the conventional wisdom that models are useful in systems development. This paper reports the results of a pilot study that examined the preference for diagrammatic models and a working prototype among student subjects, who participated as potential end users of a stock trading system. Surprisingly, the results show that non-technical users found the models to be more useful than the prototype. Implications of the study are discussed, and planned modifications for a follow-up experiment that addresses the limitations of the pilot study are presented.

Keywords: Agile development, prototype, models

INTRODUCTION

Numerous types of models are used in information systems development (Booch, Rumbaugh and Jacobson, 1999; Firesmith and Henderson-Sellers, 2000; Larman, 2004; MDA, 2006). Some of the more common models are entity-relationship diagrams (ERDs), data flow diagrams (DFDs), UML (Unified Modeling Language) class diagrams, UML use case diagrams, and UML sequence diagrams. For years it has been assumed that these types of models are inherently useful and that there are tangible benefits that accrue to the organization and/or customer when such models are developed and maintained. However, in the past decade or so, this assumption has been challenged. One of the greatest challenges has been put forth by the Agile development community; XP (eXtreme Programming) (Beck, 2000) is probably the most well-known of the Agile development approaches, but numerous other agile development approaches have developed in recent years. While traditional development approaches have generally assumed that modeling is a useful thing to do, agile developers tend to assume that modeling should not be done for it's own sake; rather, it should be done only as much as is necessary, and only to help the developer gain a better understanding of some development issue and move on. Agile developers point out that customers do not pay for models to be developed; they pay for running code. Traditional developers tend to assume that models are worth developing and maintaining in their own right, because it will make future development easier, and it will be easier for new developers to get up to speed on the system and for customers to determine if their requirements are being met.

Unfortunately, most articles published on this subject are anecdotal, conceptual, or experience reports, with very few containing hard empirical data that could be used to assess the claims that have been made for or against either traditional or agile system development approaches.

This paper reports on part of a research stream by the authors which attempts to address this deficit in empirical data regarding the value of modeling and the benefits and drawbacks of traditional versus agile development techniques. Specifically, this paper reports the results of a pilot study that examined the preference for diagrammatic models and a

working prototype among student subjects, who participated as potential end users of a stock trading system. Surprisingly, the results show that non-technical users found the models to be more useful than the prototype. Implications of the study are discussed, and the protocol for a new experiment, to address the lesson-learned from the pilot study, is presented.

In the next section we review the literature on the value of modeling and agile development. Then we discuss the research questions posed in this study. The research method is then discussed, followed by an analysis of the data gathered in the study. After this we discuss the results, propose a protocol for future studies of this type, and end with conclusions.

LITERATURE REVIEW

While much has been written in recent years about modeling and agile development (Beck, 2000; Booch, Rumbaugh and Jacobson, 1999; Firesmith and Henderson-Sellers, 2000; Jacobson, Booch and Rumbaugh, 1999; Larman, 2004; MDA, 2006; Rumbaugh, Jacobson and Booch, 1999; Turk, France and Rumpe, 2002; Turk, France and Rumpe, 2005), and there is even an annual conference on the evaluation of modeling in systems analysis and design (EMMSAD, 2005), so far very few have written about empirical studies that have collected and analyzed data about the usefulness and ease of use of these software development techniques. Recently, however, a few studies have begun to appear that are based on empirical data.

Rumpe and Schroder (2002) reported on a quantitative survey on extreme programming projects describing the companies in which such projects have been done, characteristics of the XP projects, and the companies' plans for future XP projects. At the Symposium on Research in Systems Analysis and Design, Turk, Vijayasarathy, and Clark (2003a) laid out a research agenda that specified a variety of types of empirical studies that are needed for analyzing the value of modeling in systems development. Later that year, Turk, Vijayasarathy, and Clark (2003b) reported on an initial experimental investigation into the value of conceptual modeling in database development which found that the use of models helps improve the quality of the schema and reduces the time taken to complete it. More recently, Layman, Williams, and Cunningham (2004 & To Appear) and Pikkarainen, Salo, and Still (2005) have reported on industrial case studies that explore extreme programming in their organizational contexts, identifying mixed results, and Vijayasarathy, Clark, and Turk (2005) have reported on an empirical study of the use of models in programming in which they found that those who spent less time modeling performed better, comprehended the problem better, and developed their solutions more quickly than those who spent more time modeling.

These more recent studies have relied on empirical data, but more studies are needed. These studies have been initial forays, pilot studies, explorations, into how to collect data on the value of modeling and the value of agile development approaches. Many more similar studies with more depth and breadth are needed. One attempt to add rigor and depth, and to support comparison across a broad range of studies is the Extreme Programming Evaluation Framework (XP-EF) for Object Oriented Languages that has been proposed by Williams, Layman, and Krebs (2004). This framework includes three main sections – context factors, adherence metrics, and outcome measures – which provide guidance for how to approach the collection of empirical data from agile development projects.

This paper reports on some current empirical research on the value of models versus the use of running code for customers who are attempting to assess whether a system meets its specified requirements.

RESEARCH QUESTIONS

In general, the research questions we are interested in are whether models are useful in system development, and, if so, how, how much, and in what circumstances? The specific research questions for this study were:

1. Do end-users find conceptual models or actual running software more useful in determining whether a system meets requirements?
2. Do end-users find conceptual models or actual running software easier to use in determining whether a system meets requirements?
3. Is the preference for conceptual models or actual running software moderated by the skill/comfort level of end-users with computer-based information systems (CBIS)?

METHOD

Undergraduate students majoring in business at a mid-size public university in the western United States participated in our study. These student subjects were enrollees in a senior-level information management course that is a core requirement for all business majors. In the two weeks prior to the experiment, students in this course were learning about systems development methodologies including the use of graphical models and prototypes in the development process. A few days before the experiment, one of the co-authors who was the instructor for the course, informed the students that they would be participating in a simulated systems development project. He distributed a description of a stock trading system (STS) (provided in Appendix A) and instructed his students to read and familiarize themselves with the requirements. On the day of the experiment, during their regularly scheduled class time, students were randomly assigned to two groups. One group stayed in the class-room (hence, referred to as the diagram group), while the other group (hence, referred to as the prototype group) was asked to go to a desktop computer lab located in the same building.

Subjects in both groups were first given a brief overview of the study, and asked to read and sign a waiver form that is required by the University's Regulatory Compliance Office. Next, a copy of the requirements for the proposed STS was redistributed and students were asked to become familiar with the requirements, if they had not already done so. After about ten minutes, as a check on the subjects' understanding of the requirements, they were asked to list the functionalities of STS on a sheet of paper. After returning the functionalities list, subjects in the diagrams group were given a use case diagram and a class diagram (provided in Appendix B), while subjects in the prototype group were given access to a prototype of STS built with Visual Basic .Net 2003 (sample screen shot is provided in Appendix C). The prototype group simulated customers who would be looking at an actual running system in order to evaluate whether their requirements had been met or not. Subjects were then asked to examine the diagrams/prototype and evaluate them for their completeness and correspondence to the requirements outlined for STS. After they had examined the diagrams/prototype, the subjects were asked to inform the proctor, who handed them a questionnaire. The questionnaire sought the subjects' perceptions about the usefulness and ease of use of the diagrams/prototype (Davis, 1989) and a self-assessment of their skills and comfort level with computer-based information systems. Subjects were also asked to list potential changes and additions to STS. The experimental sessions ranged from 40 to 50 minutes.

DATA ANALYSES

Our subject pool consisted of 57 female and 62 male students. All majors from the College of Business were represented in the sample.

In order to assess convergent and discrimiant validities, the 12 items (provided in Appendix D) used to measure usefulness and ease of use were subjected to principal components analysis with varimax rotation. The Bartlett test of sphericity was significant (chi-square: 776.23; significance: 000) and the Kaiser-Meyer-Olkin measure of sampling adequacy (index: 0.87) was satisfactory confirming the appropriateness of proceeding with this analysis (Norusis, 1985). After dropping three items that had high cross-loadings (i.e., > 0.4) on more than one factor, a two-factor solution (Table 1) that cumulatively explained 66.57% of the variance with Eigen values greater than one emerged.

Scale Items	**Factors**	
	Ease of Use	**Usefulness**
EOU 2	0.824	0.214
EOU 3	0.863	0.112
EOU 5	0.828	0.322
USE 1	0.079	0.829
USE 2	0.076	0.690
USE 4	0.341	0.672
USE 5	0.226	0.742
USE 6	0.338	0.790
USE 7	0.254	0.756

Table 1. Results of Principal Components Analysis

Additional checks on the validity of the scales were conducted through reliability analysis and the Cronbach Alphas for both scales were found to be above 0.8. Table 2 lists the research variables along with relevant descriptive statistics.

Variable	Number of Items	N	Mean	S.D.	Alpha
Ease of Use	3	119	5.26	1.26	0.84
Usefulness	6	119	5.24	1.03	0.87

Table 2. Descriptive Statistics for Research Variables

Analysis of Variance (ANOVA) tests were conducted to test for differences in perceptions of ease of use and usefulness between the diagrams and prototype groups. The results shown in Table 3 indicate that there are significant differences between the groups on perceived usefulness. Specifically, the diagrams group's perception about the usefulness of diagrams was higher than that of the prototype group and its perception on the usefulness of prototypes. However, there was no significant difference on perceived ease of use between the two groups.

Variable	Statistics	Diagram (n=60)	Prototype (n=59)	Source	df	Sum of Squares	Mean Square	F	Sig.
Ease of Use	Mean	5.27	5.25	Between	1	0.01	0.01	0.01	0.938
	S.D.	1.24	1.30	Within	117	188.31	1.61		
Usefulness	Mean	5.44	5.04	Between	1	4.89	4.89	4.79	0.031
	S.D.	0.97	1.05	Within	117	119.53	1.02		

Table 3 – ANOVA Results by Treatment Groups

To address our third research question, we had intended to analyze the potential moderating effect of skill/comfort level of the participants with CBIS and the relationship between the treatment groups and the study variables. However, a systematic bias in how the subjects rated their skill/comfort level depending on their assigned treatment group precluded us from conducting this analysis. We believe that these unexpected results with respect to skill/comfort level perceptions may have implications for information systems research and, therefore, have chosen to report them in this paper.

The five items (presented in Appendix D) used to measure the skill/comfort level with CBIS loaded on a single factor and had a reliability coefficient of 0.94. Table 4 shows the ANOVA results comparing the participants' perceptions about their skill/computer level with CBIS. Interestingly, subjects in the prototype group rated their skill/comfort level significantly higher than those in the diagrams group. As the assignment of the students to the treatment groups was completely random, the difference in perceptions is rather perplexing. We believe that possible explanations for the difference might be a "halo effect" or an "immediacy effect" triggered among the prototype group by virtue of their location in a computer lab and recent use of a CBIS (i.e., the prototype). This surprising result may be worthy of further study to isolate and understand the causes for the differential perception.

Variable	Statistics	Diagram (n=60)	Prototype (n=59)	Source	df	Sum of Squares	Mean Square	F	Sig.
Skill/Comfort level with CBIS	Mean	4.01	4.73	Between	1	15.44	0.01	1.18	0.005
	S.D.	1.38	1.37	Within	117	220.79	1.89		

Table 4 – ANOVA Results by Treatment Groups

DISCUSSION OF RESULTS

As can be seen for the results of the analysis contained in Table 3, the perceived usefulness was significantly different for the two groups (P=0.031). Surprisingly, those with access only to the diagrams felt that these were more useful than those who had access to the prototype. There was no perceived difference across the treatment sets regarding the ease of use of either the diagrams or the prototype itself. In addition, there appears to be no impact on this task performance with regard to either the Myers-Briggs category or the Index of Learning Style. The significance of usefulness across the treatment sets, however, indicates that a refinement of the treatment methodology might divulge additional patterns that may well be important and significant.

LIMITATIONS

Some of the limitations of our study include the use of students as experimental subjects, possible disparity in subjects' knowledge of stock trading systems, and their proficiency in reading and interpreting use case and class diagrams, potential variations in subjects' motivation level to take the experimental task seriously and complete it diligently, selection of the type of diagrams used in the study, and the design choices made in the development of the prototype. While the prototype attempted to capture all of the functionalities outlined in the requirements, it was by no means a fully-functioning system. And even though it was explained to the student subjects that the prototype was a preliminary simulation of the requirements, they may have evaluated it as if it was the final product.

FOLLOW-UP TO PILOT STUDY

We are currently designing a new experiment that will expand on the pilot study and address some of its shortcomings listed above. Some of the planned changes include the following:

- Rather than giving the study participants a pre-written set of requirements for a system development project, one of the researchers will lead/engage the students in developing the requirements for a selected project as part of a class exercise. This strategy should generate more "buy-in" for the project among the students and increase their interest and motivation level when completing the experimental tasks.
- The task will be a graded assignment for the students, thus motivating them to take the project more seriously and to increase "buy-in" even more.
- The spotting of errors and omissions will be included as another dependent variable by deliberately omitting certain requirements or misrepresenting them in the diagrams/prototype. By grading them on how well they find incorrect or missing features, the participants will be more motivated to do well and be engaged in the research tasks.
- In order to ensure participant familiarity with the domain of the experimental tasks, a project based on the students' domain knowledge will be selected (i.e., replace the stock trading system with course registration system or point-of-sale system), or the domain will be explained more thoroughly ahead of time.
- A new treatment group that is given both the diagrams and the prototype will be added.
- Longitudinal data will be collected by running the experiment twice. After the first experiment, students will be engaged in a class discussion to come up with a list of additions/modifications to the system. In the second experiment, students will be presented with the modified diagrams/prototype.

CONCLUSION

The conclusions reached in this study are significant, interesting and contain an apparent conflict with one another suggesting the need for further research. On the one hand, the group using only diagrams (and not having access to the prototype software) perceived a greater degree of usefulness for this approach while the skill/comfort level was higher for the prototype group (who did not have access to the diagrams). What drives this conflict between usefulness and skill/comfort? Due to the level of significance of both findings there must be other factors contributing to the performance on the task.

This is difficult research, combining the vagaries of measuring human task performance that can be generalized from a student population to a professional setting. It isn't surprising that so little work has been done in this regard, and yet, it is critical in our field to better understand how and why we model rather than simply making assertions that it is warranted or giving up and claiming that it is not.

REFERENCES

1. Beck, K. (2000) *Extreme Programming Explained*, Addison-Wesley, Boston.
2. Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The Unified Modeling Language User Guide.* Reading, Massachusetts: Addison-Wesley.
3. Davis, F. (1989) Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology, *MIS Quarterly* 13, 318-339.
4. EMMSAD. (2005) CAiSE/IFIP 8.1 International Conference on the Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD). There have been ten International Conferences specifically on the evaluation of modeling methods in systems analysis and design (1996-2005), the most recent of which was held in Portugal.
5. Firesmith, D. G. and Henderson-Sellers, B. (2000) *The OPEN Process Framework. An Introduction.* Addison-Wesley.
6. Jacobson, I., Booch, G., & Rumbaugh, J. (1999) *The Unified Software Development Process.* Addison-Wesley.

7. Larman, C. (2004) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Prentice-Hall.
8. Layman, L., Williams, L., and Cunningham, L. (To appear) Motivations and Measurements in an Agile Case Study. *Journal of Systems Architecture*.
9. Layman, L., Williams, L. and Cunningham, L. (2004) Exploring Extreme Programming in Context: An Industrial Case Study. Proceedings of the Agile Development Conference (ADC '04), June 22-26, 2004, Salt Lake City, Utah, USA.
10. MDA (Model-Driven Architecture). (2006 Feb 27) [Online] Available: http://www.omg.org/mda.
11. Pikkarainen, M., Salo, O. and Still, J. (2005) Deploying Agile Practices in Organizations: A Case Study, EuroSPI, November 9-11, 2005, Budapest, HUNGARY.
12. Rumbaugh, J., Jacobson, I. and Booch, G. (1999)*The Unified Modeling Language Reference Manual*, Reading, Massachusetts: Addison-Wesley.
13. Rumpe, B. & Schröder, A. (2002) Quantitative Survey on Extreme Programming Projects, *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002)*, May 26-30, 2002, Alghero, ITALY, pp. 95-100.
14. Turk, D. France, R. and Rumpe, B. (2005) Understanding Agile Software-Development Processes, *Journal of Database Management* (JDM), 16:4 (October-December), pp. 62-87.
15. Turk, D., Vijayasarathy, L., and Clark, J. (2003a) The Value of Conceptual Modeling in Database Development: An Experimental Investigation, *8th CAiSE/IFIP 8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD)*, June 16 & 17, 2003, Velden, AUSTRIA. Unpublished proceedings.
16. Turk, D., Vijayasarathy, L., and Clark, J. (2003b) Assessing the Value of Modeling in Systems Development: A Research Agenda, *Proceedings of the 2nd Annual Symposium on Research in Systems Analysis and Design*, April 4-6, 2003, Florida International University, Miami, Florida, USA.
17. Turk, D. France, R. and Rumpe, B. (2002) Limitations of Agile Software Processes, *Conference on Extreme Programming and Agile Processes in Software Engineering*, Alghero, Sardinia, ITALY, May 26-29, 2002.
18. Vijayasarathy, L., Clark, J. and Turk, D. (2005) Use of Models in Programming: A Preliminary Investigation, *Proceedings of ISOneWorld 2005*, Las Vegas, Nevada, USA, Mar 30 – Apr 1, 2005.
19. Williams, L., Layman, L. and Krebs, W. (2004) Extreme Programming Evaluation Framework for Object-Oriented Languages Version 1.4, North Carolina State University, Department of Computer Science, Technical Report TR-2004-18.

APPENDIX A - Stock Trading System

The Stock Trading System (STS) is a software application that will enable stock brokers to manage stock portfolios for their Investor clients. The STS will offer the following functionalities:

Enter and maintain stock information (company name, ticker symbol, market price and volume).

- Add and maintain Investor information (investor ID, name, contact information).
- Manage investor portfolios (buy and sell stocks, add and delete stop orders).
- Analyze historical information through a transaction log and graphs of stock prices, portfolio item values, and portfolio values.

In order for a stock broker to use this system, they must first enter or import information about the various stocks that are available in the market and the investors for whom the broker works. Each stock's ticker symbol, name, current price, and market volume (number of shares of stock issued) are tracked. Information for an investor includes an ID and name, and contact information such as mailing and e-mail addresses and phone number.

Once the stock and investor information is entered, the broker may manage investor portfolios. A portfolio is the collection of stocks owned by an investor, and it records the amount (number of shares) of a given stock that is owned by the investor. Since the STS will hold the current market price for each stock, the value of each item in an investor's portfolio can be calculated, and subsequently, the investor's overall portfolio value can also be ascertained.

Sometimes an investor will wish to automatically buy or sell some stock based on changes in the stock's market price. Rules for doing this are known as "stop orders". The STS will allow the broker to maintain (add/delete) stop order rules for any investor. Once the stop order has been executed, the rule will be deleted from that investor's portfolio.

The STS will keep a log of all transactions including the issuance and recall of stocks, price changes of stocks, addition and deletion of investors, stock buys and sells, and creation and deletion of stop orders. This historical information can be analyzed by a stock broker to a) discover market patterns and stock trends and b) monitor the performance of his/her clients' stock portfolios. The stock broker can use the intelligence gathered from these analyses to offer investment advice to his/her clients.

APPENDIX – B – UML Class and Use Case Diagrams

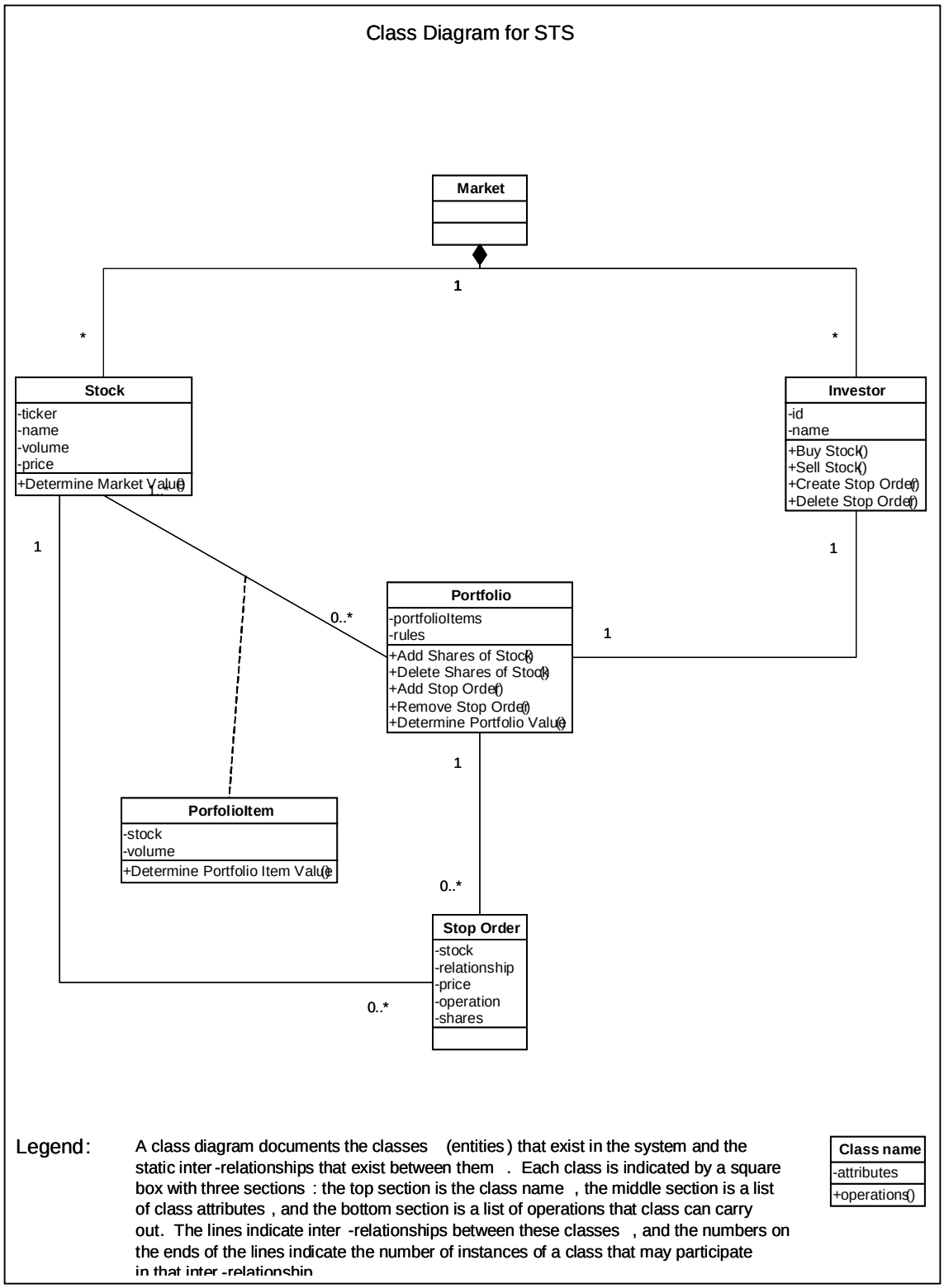

Figure 1. STS Class Diagram

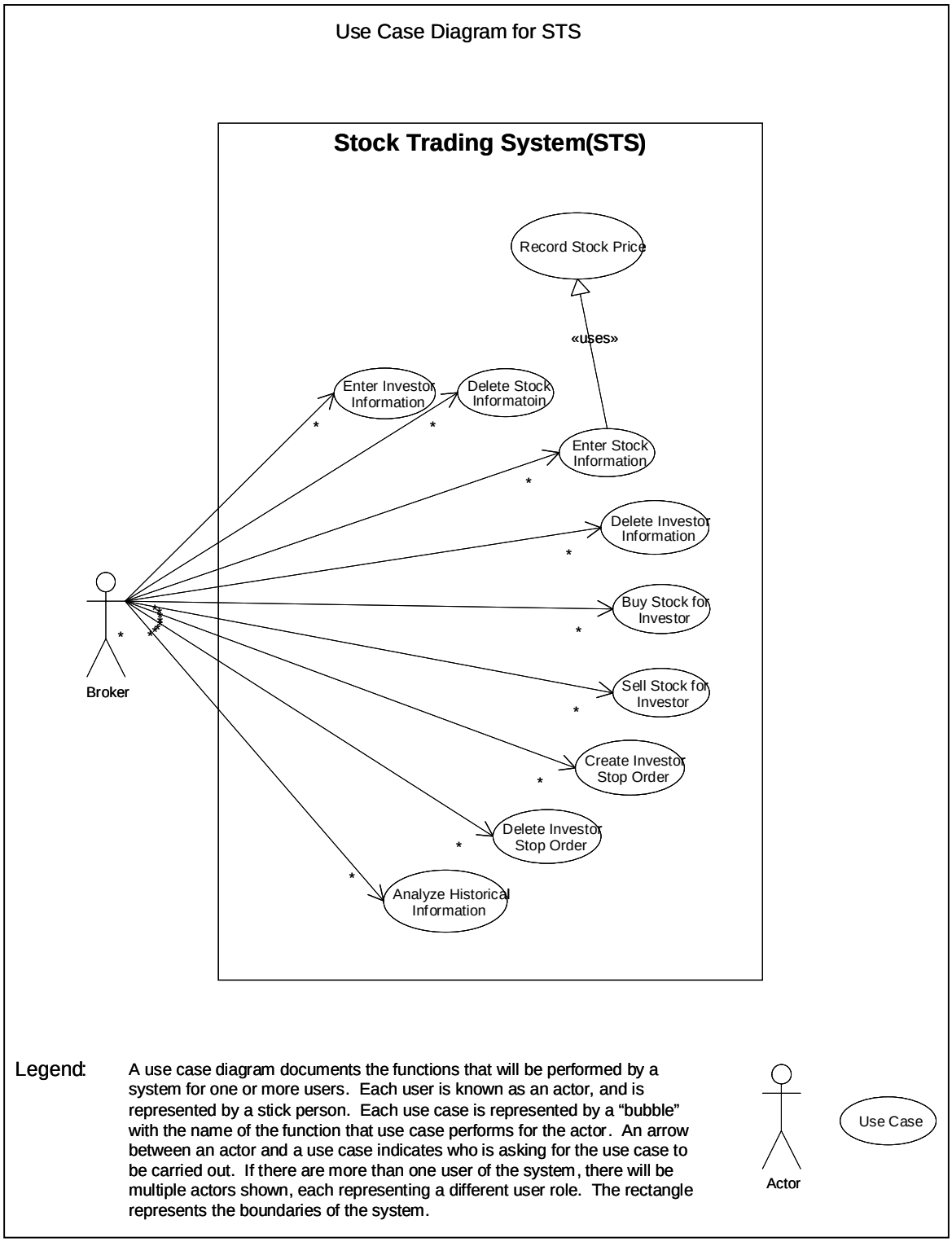

Figure 2: STS Use Case Diagram

APPENDIX – C – Stock Trading System Prototype's Sample Screen Shot

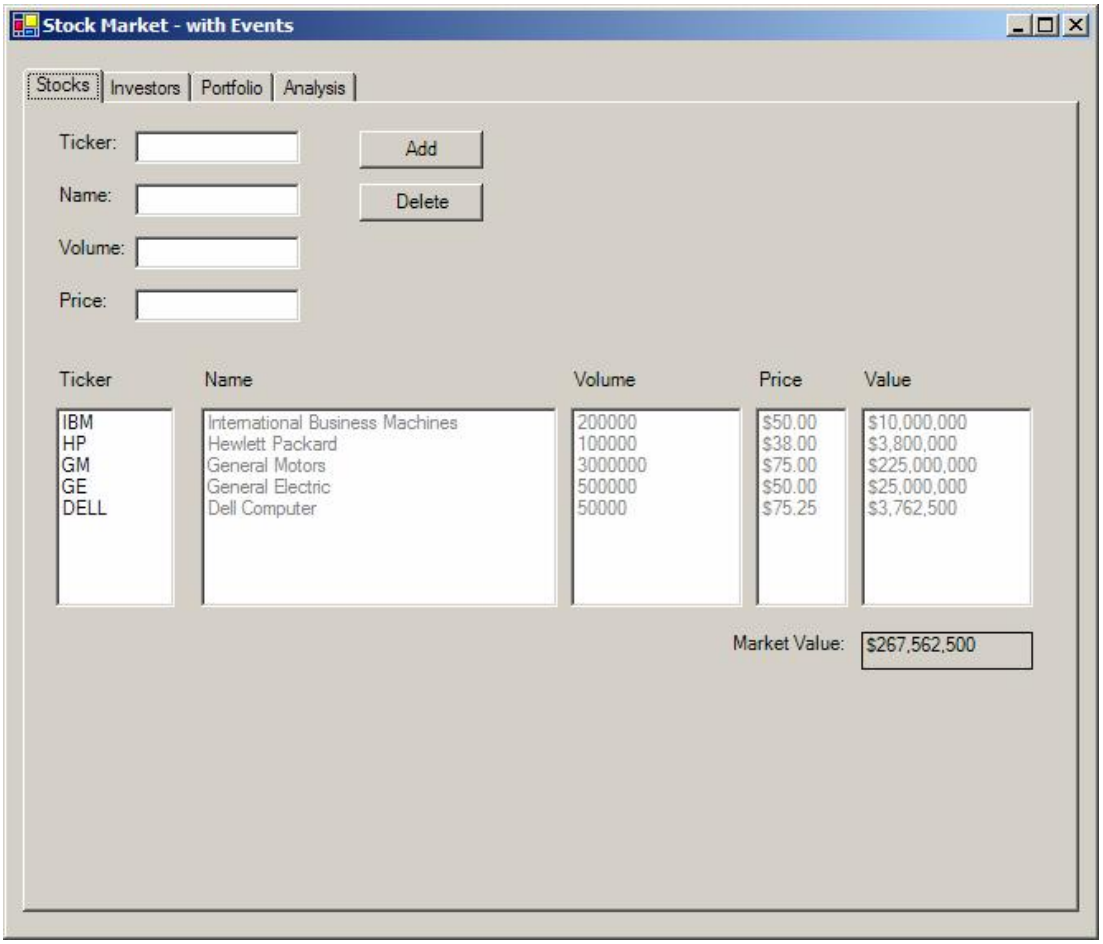

Figure 3. Stock Market Screen Shot

APPENDIX D – Questionnaire Items

Assuming that you will be a future user of STS, please indicate the extent to which you agree or disagree with the following statements. (Anchored by 1- Strongly Disagree and 7 – Strongly Agree)

a. The diagrams/prototype

- capture the functionalities of STS correctly [USE 1]
- are inadequate for representing the functionalities of STS[1] [USE 2]
- are efficient tools that I could use to identify and suggest changes to the developers of STS [USE 3]
- increase my understanding of the functionalities of STS [USE 4]
- do not represent the functionalities of STS clearly[1] [USE 5]
- are an appealing format for representing the functionalities of STS [EOU 1]
- are easy to comprehend [EOU 2]
- are complex[1] [EOU 3]
- are clear and understandable [EOU 4]

b. Overall, I find the diagrams/prototype to be useful tools for representing the functionalities of STS [USE 6]

c. Overall, I am not satisfied with the diagrams/prototype as tools for communicating the requirements of STS between users and developers[1] [USE 7]

d. Overall, I find the diagrams/prototype require a lot of mental effort to understand the functionalities of STS[1] [EOU 5]

Please indicate the extent to which you agree or disagree with the following statements. (Anchored by 1- Strongly Disagree and 7 – Strongly Agree)

a. I like working with computers-based information systems
b. Using a computer-based information system is frustrating for me[1]
c. Computer-based information systems are intimidating to me[1] [Dropped from scale after reliability analysis]
d. I am proficient at using computer-based information systems
e. I am confident about my abilities to use computer-based information systems
f. In general, how would you rate your skill level for using computer-based information systems? (Anchored by 1- Not at all skilled and 7 –Very Skilled)

[1:] Reverse Coded