INTERNATIONAL JOURNAL OF NATURAL AND APPLIED SCIENCES (IJNAS), VOL. 3, NOS.1& 2 (2008); P. 89 – 97, 8 FIGS.

The use of formal specifications in risk management as modeled in a stock tracker

I. E. Eteng*[1], B. E. Mbipom[1], I. I. Arikpo[1]

ABSTRACT

In the course of software development, people either assume that everything will go as planned or due to the nature of software development, they make sketchy plans. These two views can lead to situations which threaten the development of software projects.

Risk management is being accepted as the best practice for risk reduction and it can be used to minimize the impact of these potential problems (risks). One of the ways by which these risks in software development can be managed is through the use of formal methods. Formal methods provide more precision in thinking and documenting the early stages of the software creation process thus helping to reduce specification errors. In this papar, a stock tracker which helps users to keep track of their investment is modeled. Formal methods were applied to some aspects of the tracker which reduced risk during the modeling phase and the Java programming language was used to develop some part of the stock tracking software. The developed program provides facilities for users to check the return on their investment. The ability for the Stock Tracker to compute the return on investment for a user shows that it is quite an effective tool which users can apply not only to keep track of investments, but also to prevent losses that could ruin their investment.

INTRODUCTION

When planning software projects, it is often assumed that everything will go exactly as planned. Or, the other extreme position is taken: the creative nature of software development means one can never accurately predict what is going to happen, so what is the point of making detailed plans? These two positions can lead to software surprises, when unexpected things happen that throw the project off track (Wiegers, 2003).

Risk management is becoming recognized as the best practice in the software industry for reducing these surprises. While it may not be possible to predict the future with certainty, structured risk management practices can be applied to quickly look over the future at the traps that might be looming, and take actions to minimize the likelihood or impact of these potential problems. Risk management means dealing with a concern before it becomes a crisis.

A formal risk management process provides a number of benefits to a software project team. It gives a structured mechanism to provide visibility into threats to project success. By considering the potential impact of each risk item, one can focus on controlling the most severe risks first. Risk assessment can be combined with project estimation to measure possible loss of time if certain risks materialize into problems. Without a formal approach, risk management actions may not be initiated in a timely fashion, completed as planned, and effective as desired. The net result of these activities is to help avoid preventable surprises late in the project, and therefore improve the chances of keeping to schedules. It also helps to avoid repeating the mistakes of the past.

Risks in software development can be managed in various ways. One of such ways is through the use of formal methods, which are a fault avoidance technique that help in the reduction of errors introduced into a system, particularly at the earlier stages of design. They complement fault removal techniques such as testing and validation.

Managing risks greatly improves the likelihood of successful project completion, and reduces the potential negative consequences of those risks that cannot be avoided.

The basis of formal methods is formed by the notation of sets and constructive specification such as set operators, logic operators, and sequences. In formal methods, the data invariant, states and operations for a system function are defined by translating informal requirements for the problem into a more formal representation.

When formal methods are applied, a specification represented in a formal language such as object constraint language (OCL) is produced. The use of formal methods is quite good because logic proofs can be applied to each system function to show that the specification is correct since it uses discrete mathematics as the specification mechanism. On the other hand, even if logic proofs are not used, the structure of a formal specification will lead to improved software quality.

The failure of some systems (like safety-critical ones) could cost a lot. Lives may be lost or severe economic consequences can arise because of the failure of computer software. In such cases, it is necessary to uncover errors before software is put into operation. Formal methods greatly reduce specification errors and thus serve as the basis for software which has very few errors as soon as it is in use.

*Corresponding author
Manuscript received by the Editor May 3, 2007; revised manuscript accepted July 24, 2007.
[1]Department of Mathematics/Statistics & Computer Science, University of Calabar, Calabar, Nigeria

Objectives of the research

The objectives of this papar include:

(i) to provide formal specifications for the discovery of bugs and misunderstandings at the early stages in the lifecycle;

(ii) to translate informal requirements for the problem into a more formal representation;

(iii) to make use of discrete mathematics as the specification principle;

(iv) to model a stock tracker whose specifications are formalized;

(v) to show risk reduction using the stock tracking software, and

(vi) to aid users to keep track of their investment and also prevent losses.

Research justification

As software projects are planned and executed, future problems that could cause some loss or threaten the success of such projects could occur. There is therefore a need to apply some risk management techniques during software development. One of such techniques is formal methods.

A stock tracker which aids users to keep track of their investment was designed in this research. Some aspects of its specifications were formalized and it was used to illustrate the reduction of risks which could have resulted in problems during its implementation.

The unique ability of this stock tracker is that of computing the return of investment for any user when the current price of such an investment is supplied. This shows that the tracker is quite an effective tool which users can apply not only to keep track of investment, but also to prevent losses that could ruin their investment.

BASIC CONCEPTS

The concept of formalism in formal methods is borrowed from certain trends in 19th and 20th century mathematics. The development of consistent non-Euclidean geometries, in which supposedly parallel lines may intersect, led mathematicians to question their methods of proof and to search for more rigorous foundations. Eventually, these foundations came to be seen as describing numbers, sets, and logic. By the turn of the century, a foundation seemed to be in place, a mechanical method of manipulating symbols was thus invented to investigate these questions. Due to some fundamental discoveries, the results of using this method were ambiguous. However, the axiomatic method became widely used in advanced mathematics (Kline 1980). Formal methods are merely an adoption of the axiomatic method, as developed by these trends in mathematics, for software engineering. Even if we knew how to fit formal methods with current development techniques, there is still the problem that much software is currently produced by ‘chaotic’ processes (Humphrey et al., 1989)

Risks

The Microsoft operations framework (MOF) broadly defines risk as any event or condition that can have an impact on the outcome of an activity. It further states that within the context of information technology (IT) operations, risks are the probability, not the certainty, of suffering a loss, and the vulnerability or likelihood that the threat will occur. The loss could be anything from diminished quality of a service to increased cost, missed deadlines, or complete service failure. It also states that risks arise from uncertainty surrounding operational decisions and outcomes. Most individuals associate the concept of risk with the potential for loss in value, control, functionality, quality, or timeliness of completion of an activity. However, outcomes may also result in failure to maximize gain in an opportunity; the uncertainties in decision making leading up to this outcome can also be said to involve elements of risk.

Risk management for software projects is aimed at minimizing the chances of unexpected events, and particularly to keep all possible outcomes under firm management control. Risk management is also concerned with making judgments about how risk events are to be treated, valued, compared and combined. Pressman states that risk management is a discipline for living with the possibility that future events may cause adverse effects. Risk management can be used to continuously assess what the risks in a project are and determine which of these risks are most important, and implement strategies to deal with these risks.

The use of mathematics in the development of software systems is very effective because it can be applied to exactly describe a physical situation, an object, or the outcome of an action. Furthermore, mathematics is very useful in the software process because it provides a smooth transition between software engineering activities. System designs, functional specifications and the program code (which is a mathematical notation) can all be expressed in mathematics.The support for abstraction is a major property which mathematics possesses and it is an excellent medium for modeling. Since mathematics is an accurate method, there is very little room for ambiguity. Hence, specifications can be validated for contradictions and incompleteness using mathematics, thereby removing any trace of vagueness. Also, mathematics can be used to represent various levels of abstraction in a system specification in an organized way (Pressman, 2001).

When mathematics is used in software development, it offers a high level of validation. Its proofs can be used to show that a design matches a specification and a given program code properly reflects the design. The use of mathematics for validation of software in the early stages of development is much better than having all the checks at the testing phase.

A variety of advantages have been attributed to the use of formal software specifications. These advantages include enhanced insight into the understanding of specifications (Sommerville, 1992), (Wing, 1990) and (Zave, 1982) , help in verification of the specifications and

their programming implementations (Hall, 1990), (Jones, 1990) and possible assistance in moving from requirements specification to their programming implementation (Fraser et al., 1991).

RESEARCH METHODOLOGY

This research makes use of a software engineering approach known as formal method, the aim of which is to reduce risks in the course of developing the system. A stock tracking software which aids users to compute the return on their investment is modeled using entity-relationship diagrams and unified modeling language (UML).

The Java programming language which is object oriented is used in the implementation of the stock tracker. Java provides the users with an interactive and easy to use graphical user interface (GUI).

A stock is a store of goods available for sale. It is the total value of money, equipment, or buildings of a business or company. A stock is the business capital. Stock differs from shares because they are not issued in fixed amounts. A stock is made up of many shares, which means that, a share is a fraction of a stock. Stock belongs to a stockholder while shares belong to a shareholder. Shares can be converted to stock and the nominal value of a stock is normally higher than that of a share.

Requirements analysis

The stock tracker program keeps track of the investments of users. The program is required to store information about all stocks owned by a user, including the amount owned and other information that the user may want to record, like the date and price when purchased. The program is also able to find out the current price of other stocks in the portfolio. It is an interactive program which will request responses from the user and also allow the user to examine information about many stocks in a single session. The user must begin by invoking the program.

Portfolio

Two options are given here.

a) createPortfolio
b) editPortfolio

create Portfolio

The user is informed of the available stocks by clicking on **ListOfStocks.** The user can also click on **getQuotes** to know the current prices of the stocks. After this, the user can click on **SelectStock** for a choice to be made. An empty portfolio is created and it is the open portfolio. Users are allowed to define separate portfolios, each containing a group of related investments. A portfolio contains positions, each of which provides information about a particular stock. The chosen stock is referred to as the current position (**curPosition**). The curPosition is the open portfolio. New users fill the **portfolio** form and the **cscs** form and save it.

editPortfolio

This is utilized by old/existing users. If there is need to make corrections or include some information to their portfolio/cscs forms which have been saved already, the editPortfolio is used. The system will request for the user's **RefNum** (reference number is unique to all the user). When this is typed in, the program invokes a command to display the form corresponding to the typed RefNum. The user can edit this form and save it.

Selectstock

This allows the user to select the various positions and portfolios he wants to invest in. the selection is done based on the **List of Stocks** provided.

Buystock

This keeps track of the shares bought by a user. A user may wish to buy **n** shares and add to his current position. The user will click on **contractNote** and fill it. After that, the number of shares, NUM for the current position, is increased by n. NUM = NUM + n

Sellstock

When a user wants to sell **n** of his shares, the system will ensure that the number of shares $n \leq NUM$. (where NUM is the number of shares for the current position). The user will click on **contractNote** and fill it. After that, the number of shares NUM for the current position is decreased by **n.** NUM = NUM – n

Get Quotes

Two options are given to the user here

a) getPrice
b) getPrices

get Price

When there is a position P in **curPosition** and the user wishes to know the current price of particular stock, he clicks on the link to www.cscsnigeria.com and he can view the stock price.

getPrices

When there is an open portfolio and the user wish to know the current prices of all the positions in that portfolio, the user will click on the link to www.cscsnigeria.com and can view the current prices of all the various stocks. The user can also monitor how well the stocks are doing through the cscsnigeria website.

DeletePortfolio

The portfolio P to be deleted is put in the current position. (The current position is the open portfolio). The system will not allow a non-empty portfolio to be deleted. The individual positions within the portfolio have to be emptied first, before a non-empty portfolio can be deleted. When the user gives a command *Delete P*, P is removed from the current position in the open Portfolio.

ReturnOnInvestment (ROI)

It is necessary for a user to know how his investments are doing after a while. This can be determined by computing the capital gain, which is also called the Return on Investment (ROI). By this time, the principal (that is the original amount of money a user invested) should have appreciated or depreciated. We call this appreciated/depreciated principal **new Consideration**. The user supplies the information and the program computes the ROI automatically.

newConsideration = currentSharePrice * NUM (number of units owned)

StatutoryCharge = 4% of newConsideration

Principal (Original money invested)

ReturnOnInvestment (ROI) = newConsideration – StatutoryCharge – Principal

Requirement specification

The stock tracker is an interactive program.

NUM (the number of shares) must be a non – negative integer.

TICKER (these are the names used to refer to the various stocks which are available for investment in the stock market. It refers to how various stocks are called. In this program a List of ten different portfolios containing 2 or 3 stocks has been chosen.

Applying mathematical notation for formal specification

Buystock process

Num (Number of shares to be bought)

n (Number of shares to be bought)

- Data Invariant

  newConsideration = currentSharePrice * NUM

II The newConsideration is the appreciated/depreciated value of the principal. Statutory Charge = 0.04 * new Consideration.

- PostCondition

  ROI = newConsideration – statutory Charge – Principal

  ROI > O

II If this condition is not met

The ROI is < = O, an alert is given and the user is advised to see a stock broker because a negative ROI indicates that such an investment is not doing well.

Delete Portfolio process

- Precondition

  Portfolio = { }

II A non-empty portfolio cannot be deleted

- PostCondition

  Portfolio' = Portfolio \ positions

II The individual positions within the portfolio have to be emptied first. Portfolio = { } The Empty portfolio can now be deleted.

For new users,

#Num = O

- Precondition

  Num' = Num + n

- Post condition

  Num = O;

  Num' = Num + n >Num' = n

  Num' is the number of shares a user has at the end of the transaction.

Sell stock process

Num: Number of shares owned

n : number of shares to be sold

- precondition

  # n < = Num

II The precondition states that the number of shares to be sold must be less than or equal to the amount of shares a user owns.

- Post condition

  Num' = Num – n

  Num' is the number of shares at the end of the transaction.

II If the precondition is not met, the user will not be allowed to carry out such a transaction. The user can either reduce the amount of n or stop the selling process.

ROI process

CurrentSharesPrice, NUM, Principal

II The principal here is the original amount of money invested.

Precondition

Newconsidaration = currentshare price x Num

The newconsideration is the appreciated/depreciated value of the principal. Statutory charge = 0-04x newconsideration

- Post condition

  ROI = newconsideration – statutory charge – principal

  ROI> 0

  If this condition is not met, The ROI is< 0 an alert is given and the user is advised to see a stock broker because a negative ROI indicates that such an investment is not doing well.

Design

UML (Unified modeling language) tools were used for the design of the stock tracker. Fig. 1 shows the use case diagram illustrating both old and new users. A class diagram (Fig. 2) has shows the various classes that were developed and the relationships amongst various classes in the stock tracker. An activity diagram (Fig. 3) was also used to illustrate the various activity carried out by both old and new users

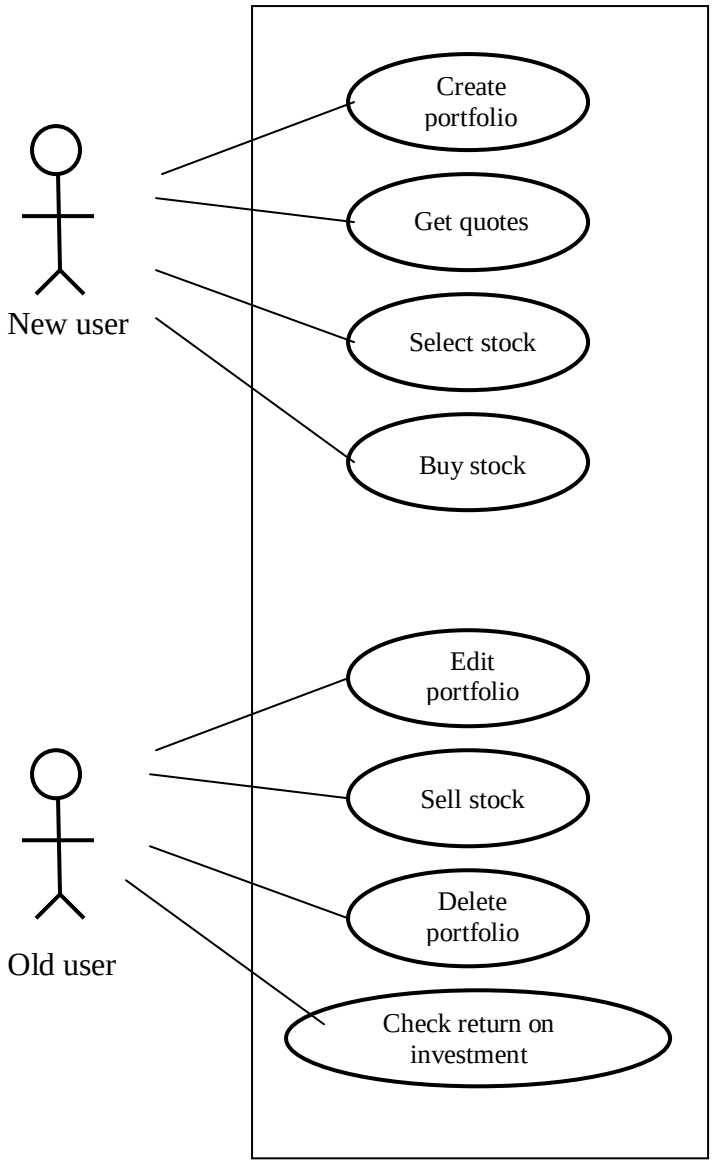

Fig.1.Use case diagram for the stock tracker

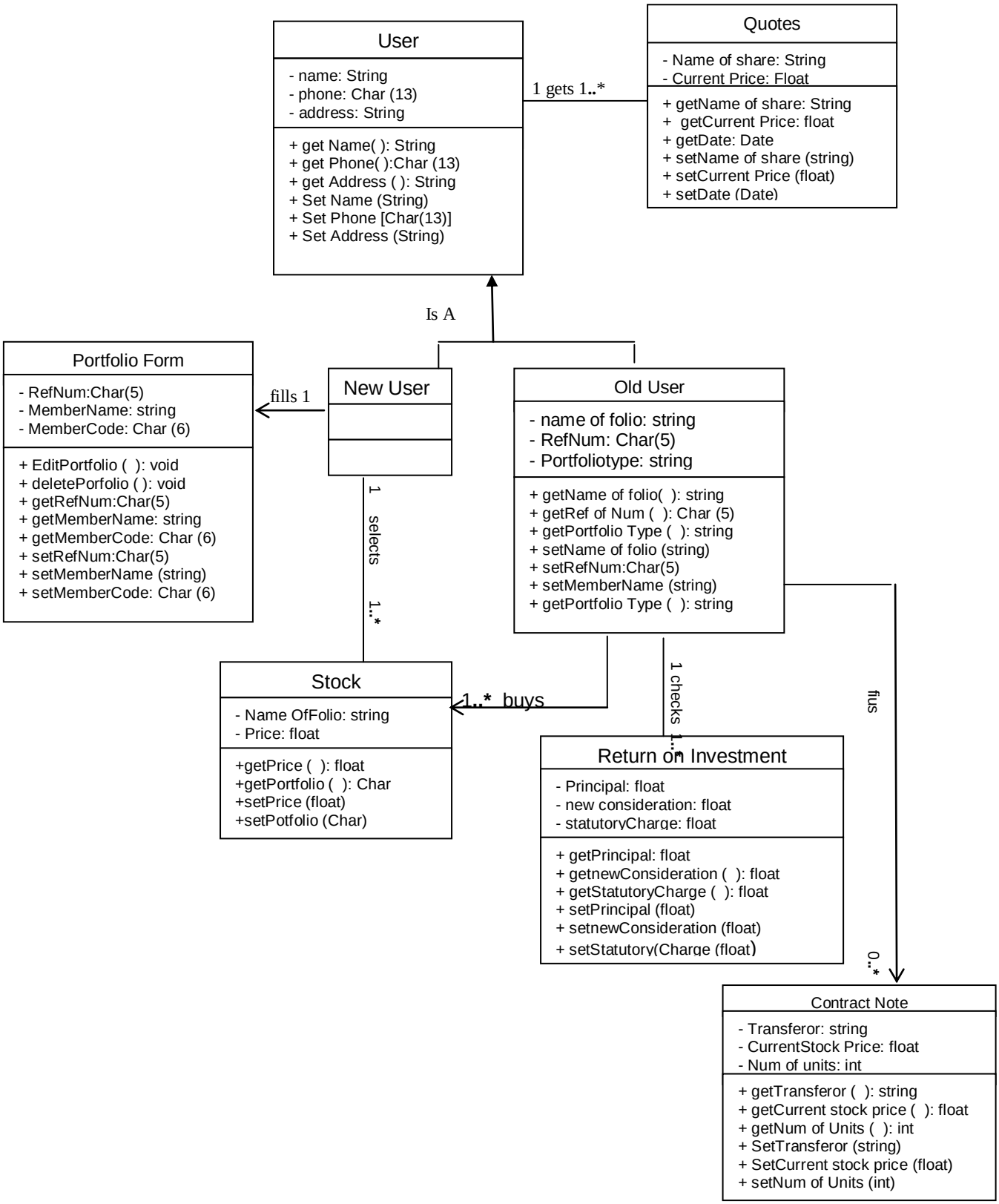

Fig. 2. Class diagram for the stock tracker

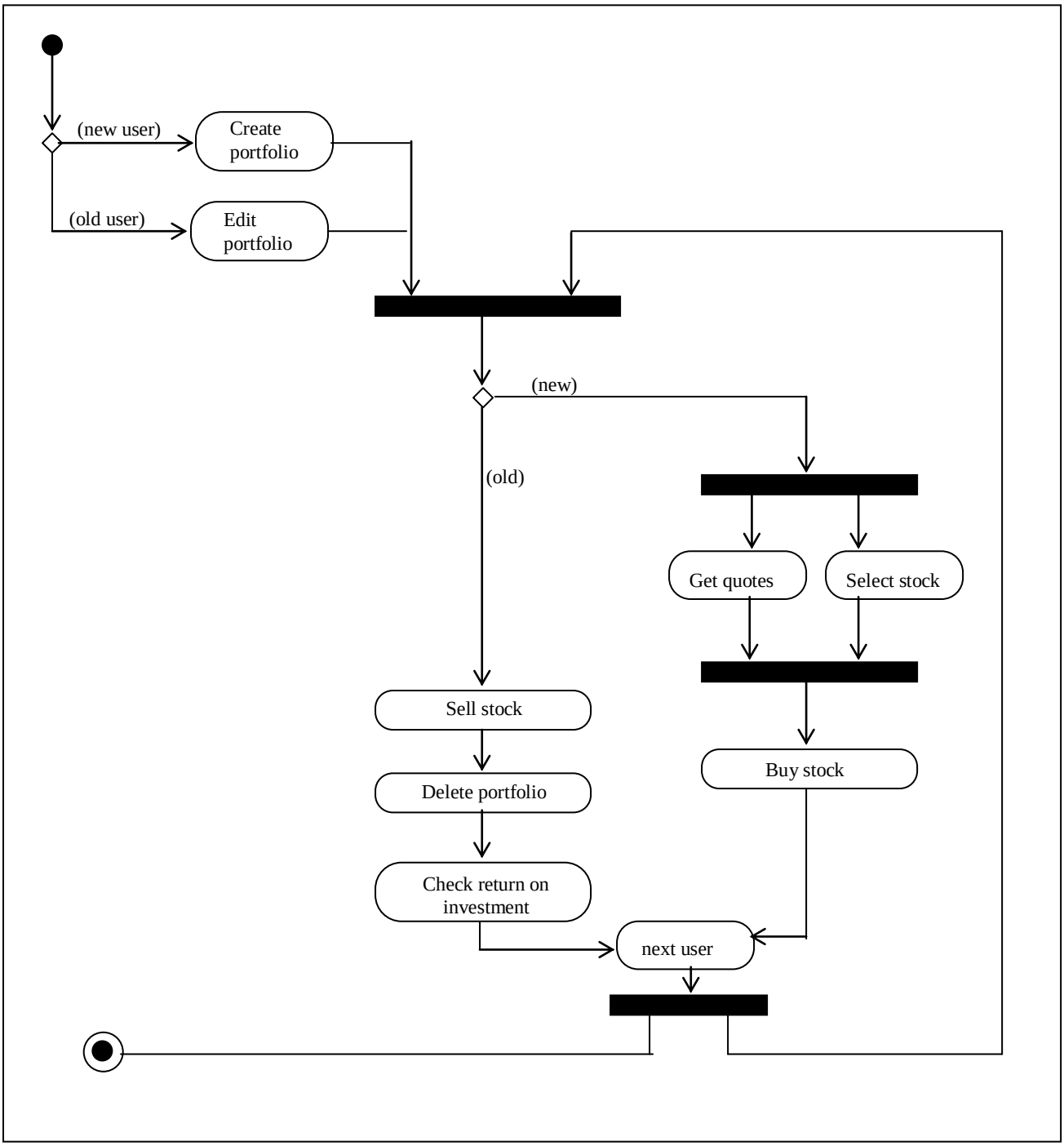

Fig. 3. Activity diagram for the stock tracker

IMPLEMENTATION

The stock Tracker which was modeled in the previous chapter is implemented here using the Java programming language which is an object oriented language. Java provides the users with an interactive and easy-to-use Graphical User Interface (GUI).

In the course of program development, various Java classes were developed for each of the functions, a user interface was designed, and error messages were included to accommodate invalid user entries. These are shown in Figs. 4-8.

Implementation and testing

The codes for the java classes of the stock tracker and the codes for the implementation were not included in this papar for want of space.

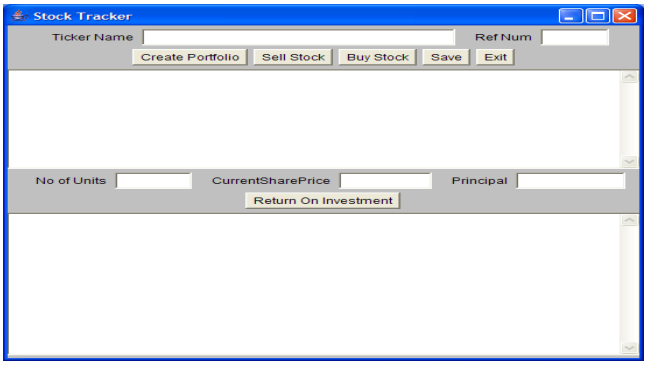

Fig. 4.An example of the stock tracker interface for users

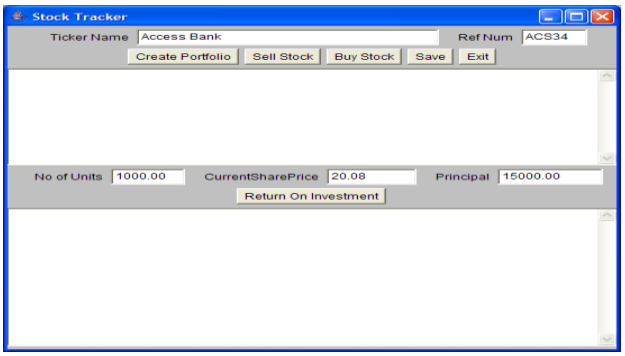

Fig. 5.The screen showing full inputs by a user

From the screen above, the user has typed in all the required data in the correct manner.

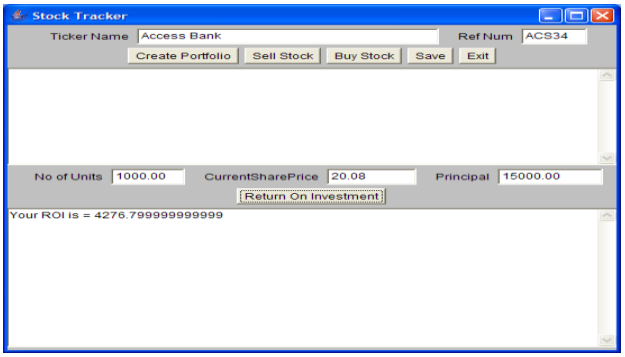

Fig. 6. The computation of the return on investment

the user's return on investment has been computed. The above result shows that the user's investment is actually doing well.

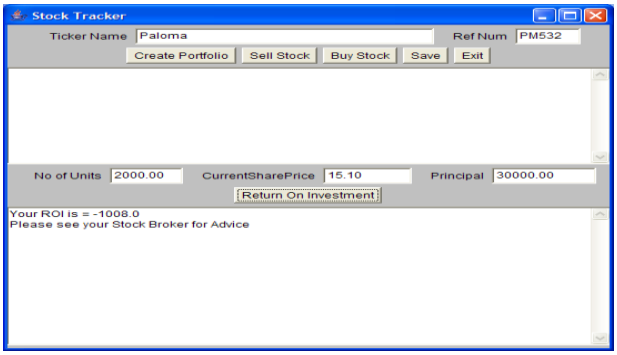

Fig. 7. A screen showing a negative Return on Investment

This screen shows a user with a negative return on investment. The user's investment is not doing well and the system has alerted the user to visit the stock broker who would advice on what to do.

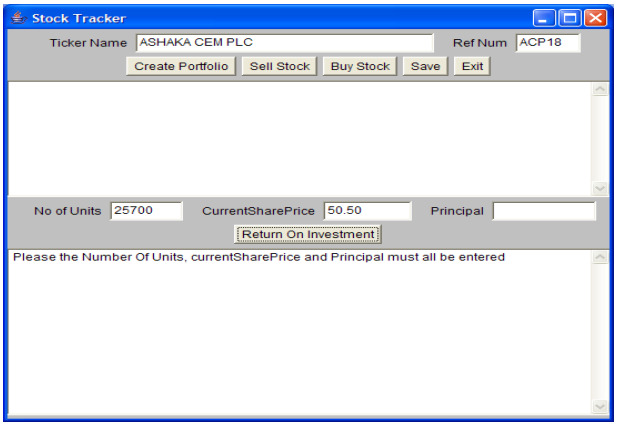

Fig. 8. A screen showing an error message due to invalid data entry

This user did not enter data in all the required fields. The “principal” field is empty so the system has prompted the user to do this.

SUMMARY AND CONCLUSION

a stock tracker which enables users to keep track of their investments was modeled. The formal method was applied to the specifications of some aspects of the Tracker and this detected and reduced some of the risks that could have resulted in problems in the course of developing the program.

The stock tracker provides facility for users to compute their return on investment based on the current prices of such investments. This feature of the stock tracker makes it quite unique and helps users to avoid losses which could be prevented.

Risk management should be integrated into decision-making just as critical factors like time; money and labour have already been integrated. It should be taken seriously and given an appropriate amount of effort and formality. The formalizing of risk management practices is a goal that can be achieved. The achievement of this goal can be enhanced by promoting a "risk management culture".

REFERENCES

Fraser, M.D., Kumar, K., and Vaishnavi, V. K. (1991). Informal and formal requirements specification languages: bridging the gap. *IEEE Trans. Softw. Eng.* 17(5) :454-466.

Hall, A. (1990). Seven myths of formal methods. IEEE *Softw.* 7(5) :11-19.

Jones, C. B. (1990*). Systematic software development using VDM*. 2ND edition. Prentice Hall, Englewood Cliffs, New Jersey

Pressman R. S. (2001). *Software engineering: a practitioner’s approach*. Mc Graw Hill, New York.

Sommerville, I. (1992). *Software engineering*. 4th edition. Addison – Wesley, Reading, Mass.

Wiegers, K.E. (2003). *Software requirement* 2nd edition, Microsoft Press.

Wing, J.M. (1990).A Specifier’s introduction to formal methods. (*IEEE Comput*. 23(9) :8 – 24.

Zave, P.(1982) An operational approach to requirements specification for embedded systems. *IEEE Trans. Softw Eng*. 18(3): 250 - 269.