Chapter 16
Requirements Analysis of a Financial Risk Management System

'*Within this gigantic auction room, it is our job to select businesses with economic characteristics allowing each dollar of retained earnings to be translated into at least a dollar or market value*' Warren Buffett

Summary
This chapter was written in order to show how the techniques developed in [Sommerville97] are applied to a real-life problem. In this particular case we are interested in documenting the requirements and architecture for an application that models European options (this is a branch of what is called risk management).
This chapter can be read on two levels: first, by those readers who are interested in learning about object-oriented techniques for risk management and second by those readers who are not necessarily interested in risk management but who nevertheless wish to learn how to describe and document requirements.

16.1 Introduction and Objectives

We discuss how the techniques developed in this book are applied to analysing risk management applications. In particular, our aim is to integrate a number of technologies to produce a seamless route from initial customer needs to software prototypes that model option behaviour in the financial markets (for a mathematical discussion of options and other financial derivatives, see [Wilmott98] and [Deventer97]). We pay attention to the following issues:

- Domain modelling (European options)
- Numerical analysis (for example, Monte Carlo simulation of option behaviour)
- Requirements analysis (determining what customer needs and wants are)
- Object-oriented analysis using UML

The test case that we analyse and design in this chapter is concerned with the modelling of European-style 'vanilla' options (see [Wilmott98]) using so-called Monte Carlo methods (see [Boyle77], [Boyle97]). The test case is small enough to allow us to create a detailed analysis model. Once the prototype has been created and a stable architecture for it has been found we shall then be in a position to generalise it to more general financial instruments, numerical schemes and output media.
The structure of this chapter is based on the following activities and deliverables:

- Motivating the commercial rationale for the system
- Carrying out a feasibility study
- Discovering the most important system requirements
- Creating an initial system architecture
- Discovering UML Control, Entity and Boundary objects (see [Jacobson99])

The system to be analysed is called Portfolio Management System (or PMS for short). We shall assume that the system is to be built for some bank. The software organisation is called Horizon Systems.
The structure of the rest of this chapter is as follows:

- Commercial Requirements Specification (section 2)
- Feasibility Study (section 3)
- Requirements Documentation (section 4)
- An object-oriented analysis of PMS (section 5)
- Conclusions (section 6)
- Glossary of terms (section 7)

We do not pretend to have considered all aspects of PMS but the information contained in this chapter can be used as a source of requirements for future versions of the system.

16.2 Commercial Requirements Specifications Document (CRS)

This section is devoted to discussing why PMS is to be built and what impact it will have. Furthermore, the commercial rationale and eventual economic benefits will be discussed. In short, this section will describe the most important reasons for introducing PMS into the organisation.

16.2.1 Introduction

Description of system to be built

An investment bank with offices throughout the world and has commissioned Horizon Systems to develop next generation software solutions for its front-office. In particular, the current dealing-room applications need to be redesigned to work with new technologies (such as object-orientation, component technology and Internet). Because the undertaking has a major impact on the organisation it has been decided to focus on a small part of financial risk management, namely option pricing. The bank has decided to concentrate on European options and on variations of these derivative products (for example, barrier and lookback options). The reason for this choice is that European options are well understood.

In the past the development of software to enable traders price options was developed in an ad-hoc fashion. Different departments used their favorite software and hardware technologies. This resulted in 'islands of information' and this state of affairs has led to major interoperability and interfacing problems between the information systems in the organisation. Furthermore, it was difficult to adapt quickly enough to changing economic and technological trends precisely because of the fact that each system was not well documented and used its own proprietary data formats, user interfaces and business logic. It is to be hoped that the new object-oriented and component technologies will resolve these and other shortcomings and in time lead to a level of standardisation that will enable the bank to introduce new financial products more quickly and in a more efficient manner than in the past.

Commercial rationale

The main business concern is to have accurate and up-to-date information pertaining to the financial markets. Having this information will lead to competitive advantage and profitability. Since PMS is focused on option pricing we need to define what 'accurate' and 'up-to-date' mean in this context:

- Accurate: the option price value is known within certain confidence limits
- Up-to-date: the option price can be calculated in real-time

In other words, it is always possible to determine the fair price of an option but if this calculation takes too long then there is no competitive advantage to be gained. It is similar to predicting the weather with the current hardware technology (supercomputers) and numerical analysis techniques. It is possible to predict what the weather will be in three days time with a given accuracy but if it takes four days to achieve this goal then there is not much point to developing a system. Thus, the timeliness of the desired information is a real need and is not disputable.

Senior management at the bank is beginning to realise that the Internet is here is stay. To this end, future versions of PMS will be Web-enabled to allow its traders (and even customers in the future!) to trade in a virtual environment.

Features in System

Senior dealing-room management has a good idea of what they want to see in PMS. The question is if it is feasible given the current budget, skills levels and technology limitations. They see PMS as a decision-support tool to allow traders to determine if an option price (as quoted in the *Wall Street Journal*, for example) is fair or not. To this end, they expect to see the following features in PMS:

- F1: Accurate option information at all times
- F2: The ability to introduce new financial products
- F3: PMS should be an 'open' system

In short, PMS should produce accurate decision-support information on whether to buy or sell financial products (in this case options). The necessary supporting information that allows a trader to reach a decision should be easy to generate, use and display. Furthermore, PMS should be designed so that it can handle emerging financial products. In particular, changes and modifications to the system should have minimal impact on the stability of the system. In the current system the introduction of a new financial product demands that much of the system be redesigned (at enormous expense). Finally, PMS must use all necessary technology so that it can be easily integrated with other systems.

Feature F1 is important to traders and front-office personnel while features F2 and F3 are realised by a good analysis and design of PMS. In particular, we can satisfy F1 by the use of supercomputer technology while the flexibility and scalability demanded by F2 and F3 will be achieved by the appropriate application of object-oriented and component technologies.

Added value of features
The features F1, F2 and F3 above are non-disputable and they represent management's position on why PMS should be built. It is up to the software development group to ensure that these high-level requirements are realised.
Concerning F1, PMS ensures that all changes in the market are made known as soon as they occur. For example, the share may go up or down or its volatility may change. These changes trigger a response in PMS. Specifically, the option price needs to be calculated (or recalculated) and then displayed as soon as possible.
This means that traders are kept up-to-date on all events of interest.
The financial market is constantly developing new financial products. For this reason it is important to build a system that can handle changes and modifications. In other words, it should be possible to define new products and integrate them quickly and seamlessly into PMS.
Finally, PMS will be developed using standard hardware and software. This means that PMS will work in conjunction with other systems that adhere to the same interoperability standards.

16.3 Feasibility Report
In this section we discuss how a feasibility study was initiated. It was not clear on how to proceed. Eventually, we decided to start with the simplest of options, namely European options and we applied the methods developed in [Boyle77] to price such options using the Monte Carlo method, a well-known and accepted simulation technique. Furthermore, due to the fact that the developer carrying out the study had little experience with C++ it was decided to use C. Finally, two versions of the code were created, namely on a sequential machine and then on a network of workstations running under Linux and employing the parallel library Message Passing Interface (MPI) (see [Gropp97]).

Short description of proposed system
The objective of the study was to gain experience in writing software systems for option pricing and to use the resulting prototype as the requirements for new systems. In particular, we wished to achieve results comparable to those in [Boyle77]. We were in the 'get it working' mode; the drop-dead date for the study was when the results in [Boyle77] were reproduced.
The Monte Carlo is one of the most popular numerical methods in the financial markets. It is a simulation tool for pricing options and other instruments. It is used in situations where other approaches fail (for example, when an exact analytic solution is not available).
Before we discuss the Monte Carlo method we must say something about how the underlying asset value behaves as function of time. It is generally accepted that it behaves in a random fashion. The stochastic random walk for the underlying S is given by:

(1) $dS = rSdt + \sigma dX$

where dS represents the change in S in some (small) time interval dt. The value r is the short-term risk-free interest, σ is the volatility of the underlying S and X is a standard Wiener process. The option price V is give as the product of the discounted value by the expected value of the payoff function:

(2) $V = \exp(- r(T - t)) * E[\text{payoff}(S)]$

Part of the feasibility study entailed implementing the Monte Carlo method in a network of Linux workstations and on an IBM SP2 supercomputer (see [Stunkel95]). It was necessary to carry out experiments in these environments because of the performance requirements in the system. To this end, we needed to answer two fundamental questions pertaining to the performance of the Monte Carlo algorithm:
- What is the speed on n processors compared with that on one processor?
- Is the algorithm scalable?

The first question has to do with the so-called speedup and it is a measure of the performance of an algorithm:

$\text{speedup} = TS/TN$

where TS is the execution time for the most efficient sequential algorithm on a single processor and where TN is the execution time for a parallel version of the algorithm using several processors.

The Monte Carlo algorithm was tested on a number of configurations and the results were:

Number of processors	Speedup Factor
2	1.272
4	2.440
6	3.493
8	4.514

It would seem that the speed up is a linear function of the number of processors. In practice however, the speedup has a logarithmic form due to the increased communication overhead as the number of processors grows.
The second question refers to scalability. In this context we say that the algorithm is scalable if the level of parallelism increases at least linearly with the problem size. It allows us to solve larger problems in the same amount of time by using a parallel computer with several processors. In other words, we can measure the capability of the system to increase speedup as a function of the number of processors. In particular, we tested scalability of the Monte Carlo algorithm by increasing the number of simulations at the same rate as that of the number of processors. The running times are practically the same and the results are now given:

Number of Processors	Running Time	Number of Simulations
2	5.609	10000
4	5.758	15000
6	5.640	20000
5	5.703	25000
10	5.945	50000
14	5.945	70000
16	5.943	80000

We thus conclude that the Monte Carlo algorithm satisfies the accuracy and performance needs of the customer and our conclusion is thus that the project is feasible.

Consequences if System is not developed
Horizon Systems has decided to enter the financial software market. Horizon has in principle the necessary multi-disciplinary expertise to enter this market:
- Mathematical skills (both analytics and numerical expertise)
- Software architectures
- Object-oriented and component technology
- C++, Visual Basic and Java

The main consequence if the system is not built is that Horizon will have lost a great opportunity to integrate a number of key technologies in order to solve problems in an important market segment.

Summary of Advantages and Disadvantages
The main advantages of having a system to price European options are:
- It shows that Horizon can achieve results in this market segment
- It consolidates and strengthens software development expertise
- The knowledge can be used by other departments within the Horizon organisation

The main disadvantages in developing this system are:
- It is not sure what the added value is at this moment
- Developers need to be trained in new technologies
- PMS needs to be improved in order to demonstrate it to customers

The advantages outweigh the disadvantages according to senior management at Horizon. Thus, the decision has been made to invest in time and money to build a system that is adaptable to market changes.

Technology Issues
Limitations faced if current technology is employed
This project is strewn with many risks. First, new software technologies will be introduced and it may be difficult finding people with the skills to carry out the project. In particular, the possibilities and limitations of the following technologies as applied to PMS are:
- Object-oriented analysis and design
- Component technology

- High-performance computing (HPC)
- Mathematical, numerical and financial methods

Furthermore, there is not much information in books or in the technical literature on how these technologies can be successfully applied to risk management applications.

Finally, even though C is an efficient language we have concerns about maintaining code written in that language. Specifically, we found that the main disadvantages of using C during the feasibility study were:

a. It is difficult to map, understand and modify code that implements abstractions such as coefficients of differential, difference and matrix equations. One of the problems is that array indexing in C starts at 0 while in numerical analysis indexing can start at any integer value (including positive and negative values). In short, the C programmer has to 'jump' from high-level conceptual entities to low-level constructs in C. This leads to many sources of error and it makes the corresponding code almost unreadable.
b. The implementation of the feasibility study was in 'flat' namespace. There is no concept of information hiding or reusability (in all fairness, this was not an objective of the study). In short, the study paid no attention to architectural and maintenance issues.
c. The lack of reusable code, structures and algorithms, for example for the following 'blocks' in PMS:
   - Algorithms for generating random numbers
   - The normal (Gaussian) distribution function
   - Low-level implementation of arrays and matrices (via explicit pointers)
   - Hard-coded option attributes (e.g. strike price, expiry date)
d. Much time was wasted on code debugging. In some cases the source of the error was trivial (a plus instead of a minus in an equation).

All code that is needed to implement the above algorithms and structures was hard-wired into the application and we shall have to do some 'code mining' when the PMS prototype is constructed using C++ in order to extract potentially reusable code.

Impact on other systems in the organisation
Since PMS is a stand-alone system at this stage we are not concerned with the relationships, dependencies and interfaces between PMS and the other systems in the organisation. This is a 'closed' system and all input and data entering the system are simulated using dialog boxes or using ASCII files.

Impact in the Organisation
Critical processes that are supported
The most critical process that is supported in PMS is that of pricing European call and put options.

Critical processed that are not supported
PMS cannot price American or other options, nor does it involve itself with other financial products such as bonds and swaps. PMS has no interfaces to real-time data feeds. All data input was from a batch file (simulated).

Finance and Budget
Can the system be developed based on the available budget?
It should be possible to develop PMS with three people in one month. The skills needed are in the following areas:
- Architectural and system decomposition
- Knowledge of financial mathematics and applications
- Hands-on experience of object-oriented and component technologies

The estimate of one month is based on projects of similar size and which were in the same domain category (PMS is an example of a Management Information System).

What should we make and what should we buy?
In the interest of reusability we should develop as little software as possible in general. However, this may not be a viable option. The approach taken is to create reusable class libraries and algorithms that can be used in several applications (and not just in PMS):
- A Property Service that can model object attributes (see [OMG96])
- Abstractions for arrays and matrices
- Algorithms for random-number generation

- Classes for options, asset models and option pricing models

The rest of the system will be built using an architecture that has proved itself for MIS applications. We should be on the lookout for possible candidate classes and interfaces that can be reused in future projects.

Recommendations

What type of system to develop

Our recommendation is to develop a system based on the PAC model (see [Datasim99], [Buschmann96]). Once this model has been chosen it is relatively easy to find the objects and other entities that realise the responsibilities of each subsystem. We shall come back to this topic in a later section.

We recommend using C++ as the implementation language. In particular, the number-crunching activities should be implemented in C++.

Using C++ resolves a number of the problems that we encountered in the C implementation during the feasibility study, namely:

a. C++ is a vast improvement on C in the sense that it is possible for the developer to create her own classes that integrate data and functionality in one place. Furthermore, all low-level issues such as memory allocation and deallocation, indexing and other housekeeping chores are hidden from clients of the newly developed classes. As an example, consider the following piece of code to create a square matrix:

   Matrix<int, double> my_matrix (10, 10, 3.14);

   In this case the Matrix constructor allocates 100 memory positions and fills each one with the value 3.14. In this case the client does not know or should not want to know how the code in Matrix has implemented memory management.

b. We implement PMS as an instance of a Management Information System (MIS) (see [Datasim99]). This is a proven architecture and we shall discuss this model in more detail in section 5.1. A MIS is a system that processes basic transaction data (for example, share prices), transforms this data into some other form and then presents it in a form that allow stakeholders to make decisions.

c. We recommend spending time either buying or making robust and reusable C++ classes that can be used as building blocks in PMS. Specifically, we see a real need for the following reusable classes:
   - Arrays, matrices and vectors
   - Statistical distributions (for example, the Normal distribution)
   - Classes for random number generators
   - Classes for property services (see [OMG96])

d) C++ can be used as in combination with Microsoft's COM and the OMG96's Corba architectures. PMS will support both of these. COM will be supported in version 1 while version 2 of PMS will support a Corba implementation.

The property service classes is important in this context because PMS is a data-intensive application. In this case we wish to add properties to, and remove properties from objects at run-time. Furthermore, property values may be constrained in some way and thus property constraints must be modelled. In short, the approach of defining hard-wired member data in C++ class definitions will not be flexible enough for the requirements in PMS.

High-level requirements and constraints

The main customer requirements are that the algorithms for option pricing are accurate and efficient. We have achieved both of these objectives during the feasibility study. It now remains to make these requirements more concrete so that an object-oriented or component-based analysis can be carried out. This is the topic that we discuss in the rest of this chapter.

16.4 Requirements Documentation

We discuss the contents and layout of the requirements document that serves as input the analysis phase of the software lifecycle in PMS. The guidelines are based on [Nasa92] with particular emphasis on the approach taken in [Sommerville97].

Introduction

This section contains information on the reasons for developing PMS and how the requirements document is organised for easy access. We hasten to add that real-life projects will have more documentation than what is described here.

Purpose and background of project
Portfolio Management System (PMS) is a system that allows traders to manage their financial products. Future versions of PMS will support many kinds of financial instruments (such as options, swaps, bonds and combinations of these instruments). For this version however, Horizon Systems is contacted to create a stable architecture that allows traders to price European options. American, Asian and other exotic option types are excluded. Furthermore, since we do not have an exact solution to the problem of pricing an option in general we use numerical analysis (this is a branch of mathematics) in order to compute an approximate solution. In particular, we shall apply the well known and accepted Monte-Carlo method (see [Boyle77] for European options and [Boyle97] for American options).

Presentation and visualisation issues are important. In particular, it must be possible to present the generated results in popular spreadsheet format (for example, Excel) and across the Web (for example, using the Virtual Reality Modeling Language (VRML), OpenGL or XML specifications)

16.4.1 Overall system concept

The PMS system can be seen as a black box. In fact, it is an instance of a Management Information System (MIS) domain category. In this case the transaction records correspond to raw input data. In particular, records contain data pertaining to options (such as strike price, volatility and expiry date). The output (final deliverables) from PMS is decision-support information. This output can be used by traders to determine whether an option is fairly priced.
In order to reduce the scope of PMS and to remove unnecessary requirements we determine what PMS's context diagram is. This is shown in Figure 1. In this case there are two interactors whose behaviour we wish to model:
Market: This is the source of all input data. It exchanges the following types of messages with PMS:
- A new option is announced to PMS
- One or more attribute values of an option changes

PMS does not send any messages to Market in this version. In general there are several actors that exchange (send and receive) information with PMS.

Trader: This is the person (or group of persons) who is interested in pricing an option. In particular, high-level decision-support information on whether to buy or sell an option is displayed. In some cases, the trader may be able to modify certain parameters in the system (for example, the number of simulations in a running Monte Carlo experiment)

In general, there are several stakeholders (for example, trading manager, Value at Risk systems) that are interested in PMS. Each stakeholder group will have its own set of viewpoints and each viewpoint will generate new requirements in PMS.

Expected system operating environment
We describe the hardware and software environments of version 1 of PMS. Furthermore, we summarise some of the changes that will take place in future versions of the products.

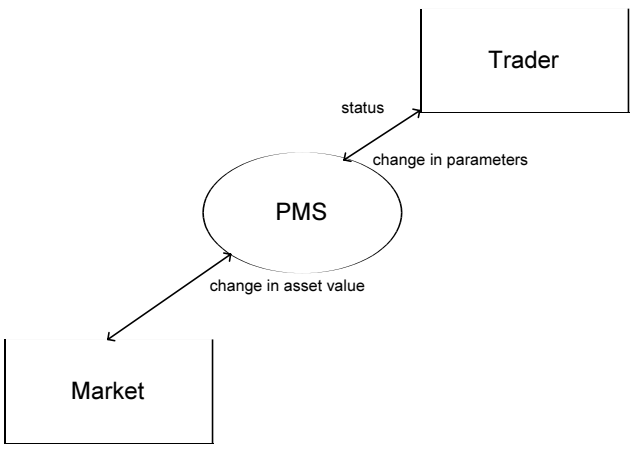

Figure 1: Context Diagram

Platform information
Version 1 of PMS will be developed under Windows and the development environment is Microsoft's Visual Studio (this contains C++ and Visual Basic). The C++ source code will make use of the Standard Template Library (STL). Furthermore, the solution will be component-based (in particular, COM/DCOM).
Horizon Systems has developed a so-called Property Services package that is based on the OMG96 Property Service and it has a number of extensions that make it suitable for modelling attributes and defining constraints on their values. Finally, Horizon has also created a flexible set of classes for a number of mathematical abstractions such as arrays, vectors and matrices.
A requirement is that the Horizon classes for properties and arrays will be used as building blocks in the development of PMS, thus alleviating some of the problems that were encountered with the implementation in C during the feasibility study.

Interface Information
As can be seen from Figure 1, PMS exchanges information with two external actors. In version 1 we shall assume that Market is modelled as a user-interface control (for example, a dialog box). All data entering PMS from the control can be encapsulated as a CSV (comma-separated variable) string. In general it is important to define and standardise this format in the early stages of software development.
Similarly, the trader can change certain 'running' parameters using a dialog box. Finally, it is important to define the decision-support information in a neutral format so that it can be displayed and presented in different packages, for example Excel, HTML and VRML.

Predictions of possible changes to operating environment
Future versions of PMS will need to run under a number of hardware and software platforms. For this reason we design the system in such a way that all sources of volatility are localised. In particular, the layered architecture that we discuss in section 5 will offer this level of flexibility.
The following changes and modifications will occur when version 1 of PMS has been completed:
- Real-time data feeds (e.g. Dow Jones) as well as dialog boxes
- Support for other types of financial instruments (e.g. bonds, hybrid and exotic options)
- Support for more accurate discrete approximations (e.g. finite differences [Wilmott98])
- Remote and local display options
- Interoperability between PMS and other external systems in the organisation

This list constitutes additions to the functionality of the system. The following list represents variations in functionality that some future customers may wish to see:
- A Corba-based solution (see [OMG96])
- Use of high-performance computing (HPC) for number-crunching
- Creating a Web-based version of PMS (for example, home trading)

We note that a HPC prototype of PMS was created using the Message Passing Interface (MPI) using a network of ten Pentium III machines running under Linux.

High-level diagrams in system
PMS is decomposed into three subsystems that co-operate to fulfil the main responsibility of PMS, which is to present a trader with decision-support data on whether to buy or sell an option. To this end, we transform the input to output in two phases in PMS (see Figure 2); first, the input data is processed by the subsystem Registration that then creates an instance of an Option entity. Then the subsystem Conversion applies a pricing model (for example, the Monte Carlo method) to the Option instance. Simulated values are produced which are displayed in a suitable medium. The display feature is the responsibility of subsystem Presentation.
To summarise, the main responsibilities of the subsystems in PMS are:

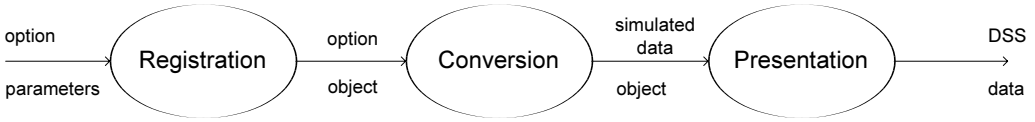

Figure 2: Information Flow in PMS

Registration
- Accept basic option data from a source (dialog box, real-time data feed)
- Create basic Option entity
- Notify Conversion that system is ready for simulating option pricing

We can conclude that Registration is responsible for the definition of the so-called continuous model associated with option pricing. This includes creating the option instance as well as defining constraints and the stochastic equations defining the behaviour of the underlying stock or share (see [Wilmott98], [Deventer97]).

Conversion
- Associate the Option entity with a discrete approximation scheme
- Calculate the price of the option using the Monte Carlo method
- Keep the trader informed of algorithm progress

We can safely conclude that Conversion is responsible for the definition of the so-called discrete model associated with option pricing. This includes defining the discrete method for simulating the price of the option and determining the desired accuracy (see [Wilmott98], [Deventer98]).

Presentation
- Accept high-level neutral data from Conversion
- Determine where and how this data is to be displayed
- Display the data on the appropriate output media

The flow of information from Conversion to Presentation is one-directional only.

Overview of high-level requirements
PMS is a prototype for pricing 'vanilla' European options (option can only be exercised at the maturity date). Early exercise is not possible and thus American-style options are excluded. Furthermore, we assume that the underlying commodity is a stock or share.
In this section we take the ISO 9126 quality characteristics as the input to finding the most important requirements in PMS. The candidates are:
- Functionality (does system satisfy the customer needs?)
- Reliability (fault tolerance, recoverability)
- Usability ('user-friendliness')
- Efficiency (computing time, resource usage)
- Maintainability (does system remain stable under change?)
- Portability (can system adapt to change?)

If you ask a customer which of the above he or she thinks is important, chances are that all of them will have the highest priority. It is your job as requirements analyst to assign a customer importance value to each one. In the case of PMS we list the following customer importance values (on a scale of 1 to 10; 1 is least important, 10 is most important)

Concern	Customer Importance
Functionality	8
Reliability	7
Usability	7
Efficiency	4
Maintainability	3
Portability	5

We note that these values are for the current system and they may change in future versions. For example, Efficiency does not play a major role now (hence the value 4) but this value will increase when a future version of the software is made that is intended to be used in a dealing room.
From these concerns we can start looking for requirements that realise then. We restrict ourselves to four major requirements and we concentrate on them.

The major requirements in PMS version 1 are:
R1: Displayed data should always reflect real operating conditions
R2: Decision-support data should be accurate
R3: Real-time data feeds
R4: Decision-support data should be easy to understand when displayed

Requirement R1 is strongly related to Functionality and Reliability, R2 is related to Functionality, R3 is related to Efficiency and finally R4 is related to Usability.
Requirement R1 refers to the fact that any changes to market data should be made known to the user of the system. In particular, each time some critical parameter changes, the algorithms in Conversion should be triggered and the results displayed. Typical examples of changes are:
- A new option enters the system
- The value of the underlying share changes
- The share volatility changes

Requirement R2 states that it must be possible to conclude that the values of all generated results are within certain confidence limits. This guarantee can be given; for example, the Monte Carlo method increases the number of sampling points in order to increase accuracy, albeit at a performance cost.
Requirement R3 is not discussed in this version of PMS but it is an important requirement in future releases. This requirement demands access to real-time data feeds.
Requirement R4 is clear. In particular, the generated data should be displayed in a form that is easy to understand for the trader. We suggest that Excel be supported, at the very least. Future versions of PMS will interface to external Value at Risk (VaR) systems and hence interoperability issues will also be important.

16.4.2 Requirements

We document requirements R1, R2 and R4. We apply the practical guidelines that have been developed in [Sommerville97] with emphasis on the following aspects:
- The identifier and name of requirement
- Description (a natural language description of what the requirement is)
- Rationale (underlying reasons for the existence of the requirement)
- Sources (which stakeholders want to see the requirement implemented)
- Risks involved (high, medium, low e.g. the requirement might be difficult to realise)
- The priority (high, medium, low)
- Possible conflicts with other requirements
- Quantitative specification (assign metrics to requirement)
- Related Concerns (from which concerns did the requirement originate?)
- Change history (audit trail of changes to requirement)

We stress the following descriptions represents the end-result of numerous discussions with stakeholders. This chapter does not discuss how the results and conclusions were arrived at.

Requirement R1 'Displayed data should always reflect real operating conditions'
Description of the requirement: The decision-support information that is displayed on a trader's screen should be up-to-date. 'Up-to-date' means different things to different stakeholders. For a trader it means that all relevant information from the real-time data feeds should be processed, calculated and displayed within seconds of its arrival in the system. For other stakeholders (e.g. market makers, homeowners who trade in shares) who have a longer time horizon these performance requirements can be relaxed.
The trader should be notified of any major bottlenecks and performance problems that arise in the system. This allows the trader to switch to another system without losing vital information on market movements.
Rationale: This requirement is needed because having decision-support information in real-time is essential in order to compete with other organisations.
Source of requirement (stakeholder): This is needed by both the trader and his management.
Risks (High): The main risks are:
- Schedule risk
- Process risk

First, the stringent performance requirements imposed on PMS may force developers to spend a lot of time tuning the system with the result that there may be less time for other activities. This can lead to budget overruns. This leads to a schedule risk. Second, the fact that we have chosen to use modern software and hardware technology means that a

new way of working needs to be introduced into the organisation and this adds to the uncertainty in the system. This is thus a process risk.
Another problem is that some developers go to extreme lengths in the early stages of software development to make the software as efficient as possible. This is to the detriment of maintainability and understandability of the code. This is a common situation.
Priorities: This is a high-priority requirement because if it is not realised the PMS system will not have any value. We say that R1 is and 'essential' or 'must-have' requirement.
Conflicts: There is a potential conflict between R1 and R2. This is because it takes more computing time to calculate a more accurate solution than a less accurate one. Some users of the system are more demanding than others: some want quick answers irrespective of the accuracy while accuracy is of paramount importance for others. PMS should be so flexible that it can accommodate both types of user.
Making data easy to understand (requirement R4) may conflict with requirement R1 because R4 involves data being transformed to various display formats.
Quantitative description of requirement: The 'passthrough' time in PMS should be at most two seconds. This includes reading real-time data feed information, carrying out a simulation (using Monte Carlo, for example) and displaying the results on the trader's screen.
Related Concerns: Functionality and Reliability
Change history: This is version 1. No changes as of yet

Requirement R2 'Accurate decision-support data and information'
Description of the requirement: This requirement states that all data in the system be known to within a given tolerance. The different types of data are: real-time (raw) data, calculated data and formatted data. Raw data is processed by sensors that connect to data feeds (e.g. Reuters), This data is then used as input to a simulation program (e.g. Monte Carlo). Most simulation programs allow critical parameter values to be set in order to increase the accuracy of the simulation method. Finally, the values that are produced from the simulation program are formatted and displayed on various media.
Source of requirement (stakeholder): Trader
Rationale: It is vital to have accurate results.
Source of requirement (stakeholder): Trader
Risks (medium): The main risk here is an external risk because it is difficult to find personnel with the necessary mathematical skills. Training IT personnel to become proficient in numerical mathematics and risk management theory is a time-consuming job.
Priorities: This is a medium to high priority. In some cases the trader may settle for less accurate results if they can be quickly calculated.
Conflicts: Requirement R2 may conflict with requirement R1 (see discussion of R1)
Quantitative description of requirement: This can be answered by knowing the workings of the numerical methods involved. For example, the accuracy of the Monte Carlo method can be increased by splitting the time horizon between now and the maturity date into more periods and introducing more scenarios (simulations). Similarly, the accuracy of finite difference methods (see [Duffy80], [Wilmott98]) can be increased by making the so-called mesh-size smaller. This is a well-developed area of mathematics and error estimates between the exact and approximate solutions are known.
Related Concerns: Functionality
Change history: This is version 1. No changes as of yet.

Requirement R4 'User-friendly data display'
Description of the requirement: This requirement states that high-level decision-support information should be displayed in a form that is easy to understand. There are two issues: first, the type of output (line chart, bar chart, 2D or 3D) and the software medium. The latter includes software packages such as Excel (a popular spreadsheet with traders), Oracle (the data needs to be stored) and XML (a markup language and platform-independent format for the exchange of data). Furthermore, some level of 'Web-enableness' must be supported in this version of the software.
Rationale: It should be easy to comprehend what the results of a calculation mean.
Source of requirement (stakeholder): Trader, Trader Manager and other stakeholders (e.g. VaR manager)
Risks (low to medium): There may be some stability risks because of the changing formats and the disparity between the different output medium types.
Priorities: Medium priority
Conflicts: Possible conflict with R1 (see description of requirement R1)
Quantitative description of requirement: Not applicable in this version
Related Concerns: Usability
Change history: This is version 1. No changes as of yet.

It seems that requirement R1 is high-risk and high-priority (at least, if we are to believe the stakeholders). In order to elicit more requirements and to show how to implement R1 we suggest defining a scenario that documents how real-time data is processed, calculated and displayed by PMS. This documentation could be used as the input for another elicitation discussion. It could also be used as the input for a prototype solution.
The steps for documenting scenarios are well documented (see [Sommerville97], [Jacobson99]). These are:

Scenario template
- Name of scenario
- A description of the system before entering the scenario (Preconditions)
- The normal flow of events and actions
- Exceptions to the normal flow of events
- Information on other activities taking place at the same time
- A description of the system after completion of scenario (Postconditions)

We now apply the above template to a particular instance in PMS.
Scenario name: Process new real-time data
Preconditions: The system is accepting data
Flow of events: There are three main events in this scenario:
- E1: New real-time data arrives
- E2: New calculated data is ready
- E3: The results are displayed on the trader's screen

As soon as event E1 occurs the data is processed, filtered and validated. Once accepted, the data is stored in an option entity. Then event E2 is triggered whence Monte Carlo simulation begins. Once finished (event E3) the calculated data is displayed on the trader's screen.
Exceptions: Exception can take place after E1 has occurred and before E3 has occurred. The main exceptions to the flow of control are:
- PMS unable to process new raw data
- Raw data has not been sensed for some time
- Problems with simulation method (timeout, overflow)
- Display medium is not responding

The reader may find it strange that we should consider hardware failures to be part of the problem domain. We introduce such features into the system at this early stage because hardware failures will trigger other emergency scenarios and these should be modelled as soon as possible.

Other concurrent activities
- Data from multiple data sources
- Other options are being modelled
- Other information is being displayed

These concurrent activities will be important when we start modelling systems with very many types of financial instruments (portfolios, for example).

Postconditions
Decision-support information has been displayed and the system is waiting for the next real-time dataset.
The next section deals with how the above requirements (and in particular the above scenario) are mapped to lower-level and more tangible entities.

16. 5 Analysis of PMS System
It is not possible (or even desirable) to include all system specifications in a single model. Instead, the requirements document should contain a number of system models that allow the system under consideration to be viewed from several different perspectives. In particular, customers, information analysts and designers will read the document. To this end, we include a number of so-called complementary models are developed and discussed in this section. They are:
I: The architecture of PMS (an instance of a PAC model), version 1
II: UML class diagrams for the current version of PMS
III: Sequence diagrams showing event flows in scenarios

IV: Model for assets, pricing algorithms and instruments
V: The relationships between the different models
VI: The architecture of PMS (future versions)

There are other complementary models possible but they are not discussed here. For example (see [Sommerville97]) we could include a data-processing model that show how data and information are processed in the system and state transition diagrams (or statecharts in UML) could be used if they add value to the understanding of the problem. These class diagrams function as input to the design phase of the software lifecycle and an experienced designer should be able to map these into design blueprints. We now discuss each of the above attention points.

16.5.1 PMS Architecture Version 1

The major requirements for PMS have now been discovered and documented. Furthermore, we know what the structural decomposition of PMS is (it is a MIS system). We are now in a position to begin with an analysis of the most important classes in PMS. In particular, we investigate the following areas:

- Control, Entity and Boundary objects in each subsystem of PMS
- Interactions between the objects and subsystems in PMS
- Integration with Property Services

These attention points are discussed in this section. Please note that we have not generalised the problem (yet). We have already stated that PMS is an instance of a MIS domain category and we have seen that it is decomposed into three well-defined subsystems Registration, Conversion and Presentation. We are now interested in defining a layered architecture using the PAC (Presentation-Abstraction-Control) model and the object categories in [Jacobson99] and [Buschmann96]. Each subsystem in PMS is layered as follows (see Figure 3):

- Level 3: Control objects
- Level 2: Entity objects
- Level 3: Boundary objects

The subsystem Registration contains the following objects:

- Level 3: Registration object (used for notification and dispatching)
- Level 2: Option object (containing all initialised option property values)
- Level 1: Acceptor object (a dialog box in version 1)

It can be difficult to find the Entity objects in general. However, we usually know what the data is in a given level. For example, in this case we know that the following attributes in level 2 are:

- Volatility

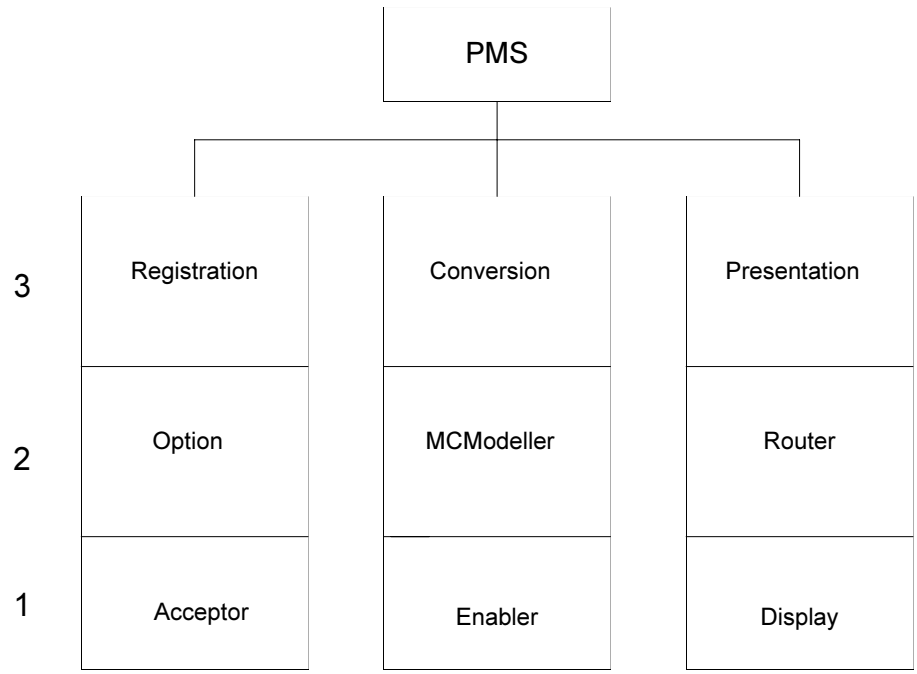

Figure 3: PAC Model for PMS

- Strike price
- Expiry date
- Risk-free interest rate
- Value of underlying stock

In this case we encapsulate these attributes in an object that we see as an instance of class Option. We have thus discovered an Entity object!

The subsystem Conversion contains the following objects:
- Level 3: Conversion object (used for notification and dispatching)
- Level 2: MCModeller object (a strategy object for a Monte Carlo algorithm)
- Level 1: Enabler object (usually an input/output GUI panel)

The subsystem Presentation contains the following objects:
- Level 3: Presentation object (used for notification and dispatching)
- Level 2: Router object (knows where to dispatch each message)
- Level 1: Display object (e.g. Excel sheet, HTML page, XML)

The PAC architecture is shown in Figure 3. The corresponding UML class diagrams is shown in Figure 4. We adopt the standard of modelling PMS and its subsystems as an aggregation while the relationships between the classes in

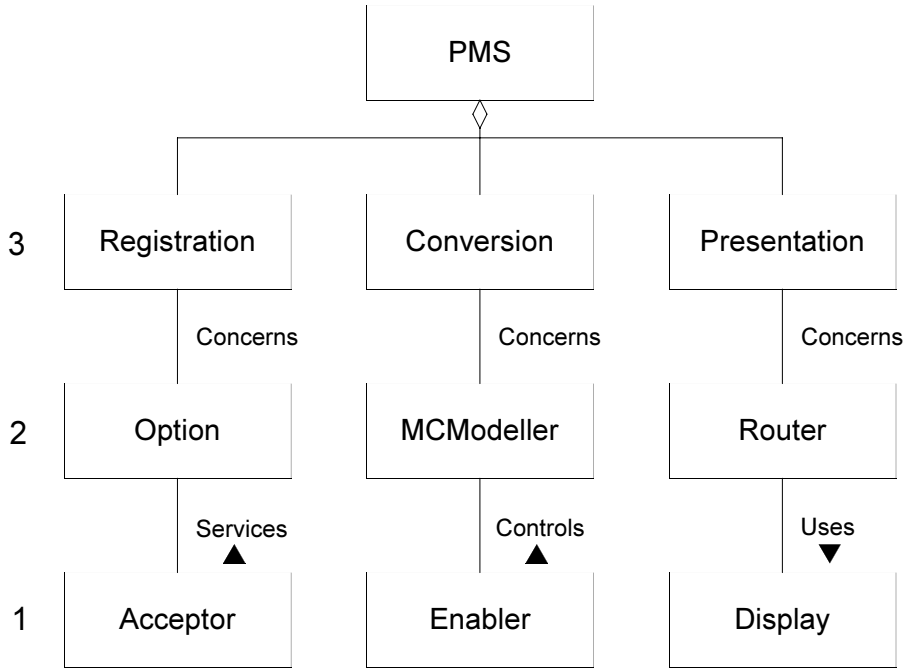

Figure 4: Class Diagram (Composition Model) in PMS

each subsystem are modelled as associations.

16.5.2 Mapping Use Cases to Sequence Diagrams

The most important requirement in PMS is to price an option and to display results in an efficient way. This requirement is decomposed into three sub-requirements:
- R1.1: create option information
- R2.1: price the option
- R3.1: display the decision-support information pertaining to the option

Since each of these sub-requirements is closely aligned to a specific subsystem in PMS it is possible to create a sequence diagram for each of the above sub-requirements. These are shown in Figures 5, 6 and 7. We explain each one in turn. First, in Figure 5 the option object (which we assume exists) is initialised with data from the Acceptor layer. Once this is done the Registration layer notifies Conversion that it can start with the actual pricing of the option. To summarise, the activities in R1.1 are:
- Acceptor senses that new market data has arrived in the system

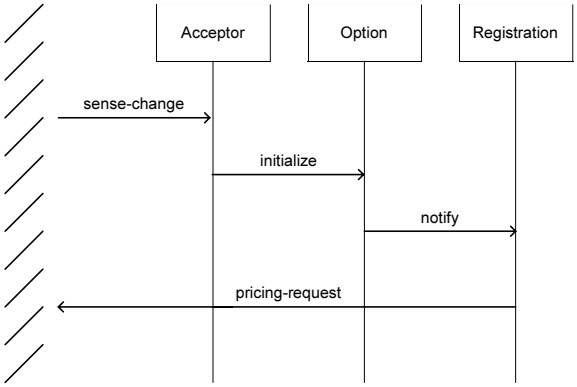

Figure 5: Initialize Option Data

- Acceptor initialises the option with market data
- Option notifies Registration
- Registration notifies Conversion

It is possible to speculate on the parameters of the messages in Figure 5 but we do not do this at this stage because it is too detailed.

Figure 6 documents the activities in R2.1 when pricing an option. They are:
- Conversion is requested to price an option
- We choose a suitable modeller (Monte Carlo MCModeller)
- The pricing algorithm is executed (message get_price)
- The Boundary object Enabler is kept informed of progress
- MCModeller informs Conversion that it is finished

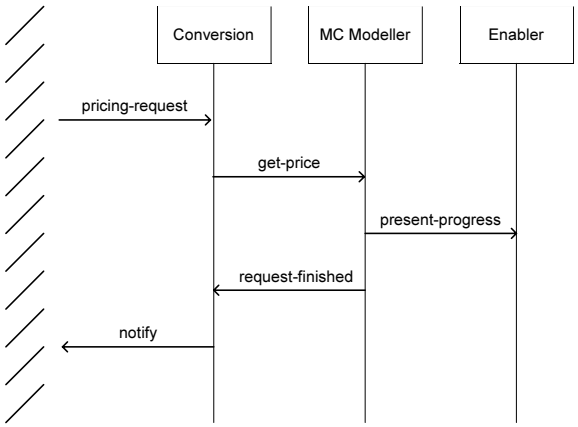

Figure 6: Pricing the Option

- Conversion notifies clients that option pricing data is ready for display processing

Figure 7 shows the activities in R3.1 when presenting the option pricing data in a given display medium. These are:
- Presentation is informed of a display request
- Presentation dispatches to Router
- Router determines where the data should be sent to
- Output on Display
- Once displayed, Display may give feedback to some external 'watchdog' system

Once the basic flow in the system is working we can then start thinking about some possible optimisations, for example:
- 'Improving' message names (e.g. making them more polymorphic and standardised)

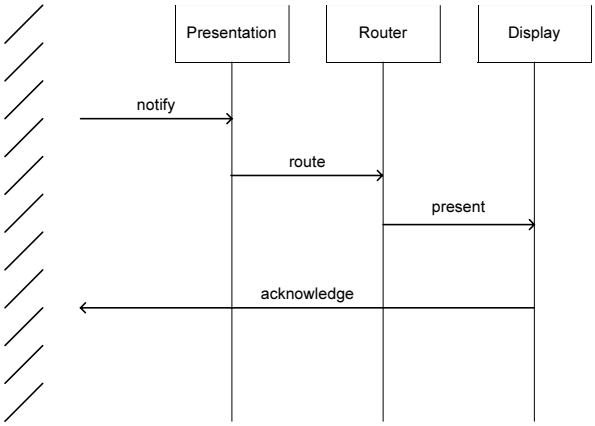

Figure 7: Presenting DSS (Decision Support) Data

- Input parameters for messages
- Determining the return types of messages in figures 5, 6, 7

However, these activities are more suitable in the design phase of the software lifecycle.

16.5.3 Classification Model and Domain Classes

In general, there are several 'forests' of class diagrams that are needed and used when analysing PMS. These correspond to the classes in the current version of the software and to those classes that can be used by different customer groups and even in different applications. These are called application classes and domain classes, respectively. We have seen an example of application classes in figure 4. The domain classes that we discuss in this section have to do with the following models:

- Asset model hierarchy (figure 8)
- Derivative model hierarchy (figure 9)
- Discrete model hierarchy (figure 10)
- Basic instrument class hierarchy (figure 11)

The AssetModel (figure 8) hierarchy contains classes that model the behaviour of the underlying asset. Most models

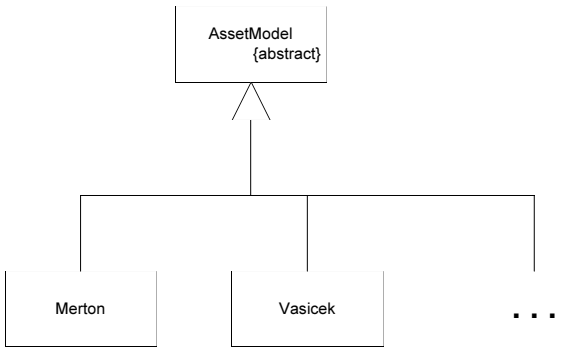

Figure 8: Asset Model Classes

are stochastic in nature and the data in the classes corresponds to those parameters that characterise the corresponding stochastic equations (see Appendix 2).

The DerivativeModel (figure 9) contains classes that model the behaviour of the derivative instrument (in this case European options). One example is Parabolic; this is a class corresponding to the parabolic partial differential equation that defines the behaviour of a European option.)

The DiscreteModel (figure 10) contains classes that approximate the classes in the AssetModel an/or the DerivativeModel. For example, the Monte Carlo method (MCModeller) simulates the Merton model while Duffy represents a discrete approximation to Parabolic.

The basic Instrument hierarchy is shown in figure 11. This contains the classes that correspond to financial derivatives. We are particularly interested in the Option class in this version of PMS.

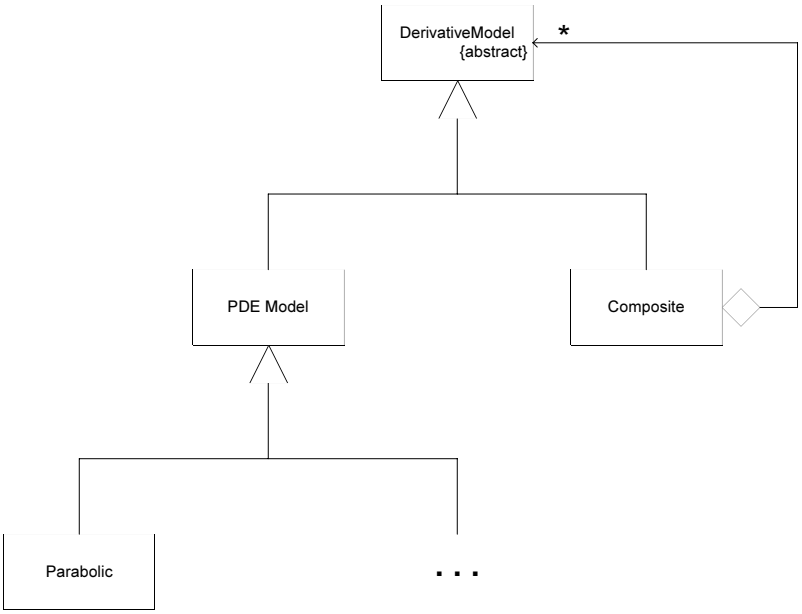

Figure 9:Models for Derivative Products

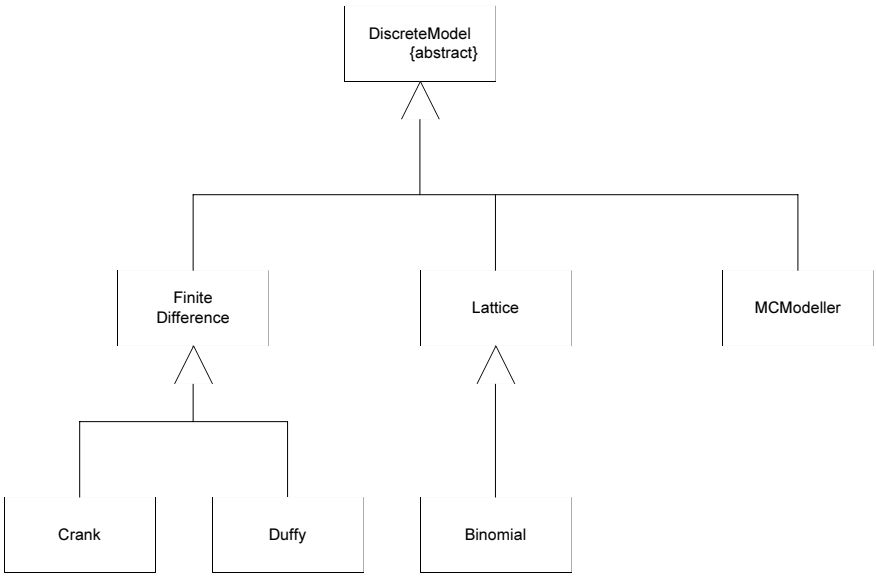

Figure 10: Numerical Models

An early set of class diagrams for this kind of problem is given in [Duffy95].

16.5.4 Relationship between the different Models

The different models in section 5.3 are related to each other. On the one hand, we must relate instruments with their corresponding asset and derivative models. Furthermore, derivatives must be approximated numerically by some discrete model. The relationships are shown as UML binary associations in figure 12:

- A1: an instrument is based on an asset model
- A2: an instrument is characterised (determined) by a derivative model
- A3: a derivative model is approximated by a discrete model

An important question is: how are these relationships maintained? In particular, we need to answer the following questions:

- Problem 1: Creating a link (a link is a relationship between two objects)
- Problem 2: Modifying a link in some way
- Problem 3: Breaking the link between two objects

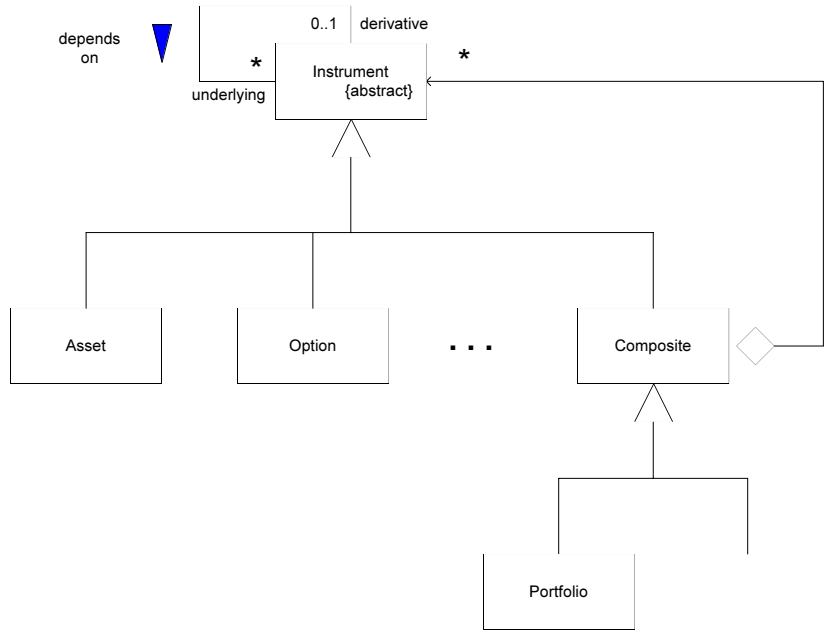

Figure 11: Classification Model (simplified) for Financial Instruments

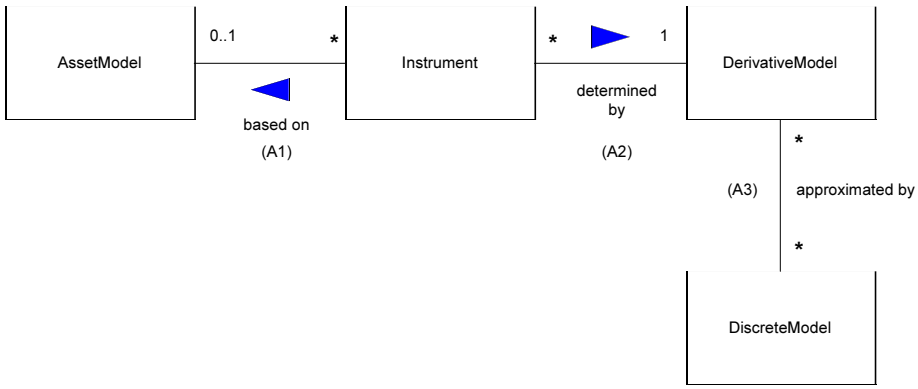

Figure 12: Relating the Different Models

In general, links and associations can be defined in configuration files or they may be established at run-time. For example, we can configure PMS so that instruments are always modelled by an instance of Parabolic.

16.5.5 Options and the Property Service

In this section we concentrate on option attributes. These will be mapped to properties.
An option is characterised by the following attributes:

- t the current time
- T the expiry date
- S the current value of the underlying stock
- E the strike price
- σ the volatility of the underlying stock
- r the risk-free interest rate

Each of these attributes is mapped to a corresponding Property object. Each property has a name, value and possibly a number of constraints that the property value must satisfy. Constraints are particularly important for certain types of options.

In general we speak of two kinds of model when pricing options:

- AssetModel: The continuous model for the underlying asset
- PDEModel: The continuous model for the derivative product based on the underlying asset

- DiscreteModel: The discrete numerical model

By the continuous model we mean those equations that model the random movements of the underlying asset or share. These so-called stochastic models allow us to simulate the behaviour of many types of asset (including shares, interest and even companies).
The second model PDEModel ('PDE' stands for Partial Differential Equation) is obtained from the AssetModel by the application of a fundamental result due to Ito (see [Deventer97]). The resulting equation is a so-called parabolic partial differential equation. Its defining parameters are:
- The coefficients of the equation
- The terminal conditions (i.e at the maturity date)
- The boundary conditions

Finally, the DiscreteModel determines how we will approximate the solution of the PDEModel. There are various options:
- Monte Carlo and quasi-Monte Carlo methods
- Binomial lattices
- Bushy trees
- Trinomial lattices
- Finite differences and finite elements (see [Duffy80])

This version of PMS supports Monte Carlo methods only and to this end we must know all the defining parameters of this method so that they can be integrated into the software system.
The Property Service is so flexible that it can handle most of the requirements that are placed on the data in PMS. First, a property can be defined as read-only, read-and-write or write-only. For example, a property that can be initialised only once could be defined as a read-only property. Second, some properties must remain within some limits (for example a share price that must remain in the range [100, 120]). If the value of the property goes outside the limits a certain message should be triggered (for example, kill the corresponding option).

16.5.7 Induction and Future Versions of PMS

Once version 1 of PMS has been realised and has been successfully released the developers at Horizon will be faced with the challenge of adding new functionality to the system. Typical questions include:
- Feature 1: Extending the functionality of PMS
- Feature 2: Integrating PMS with other systems in the organisation
- Feature 3: Building new systems based on PMS

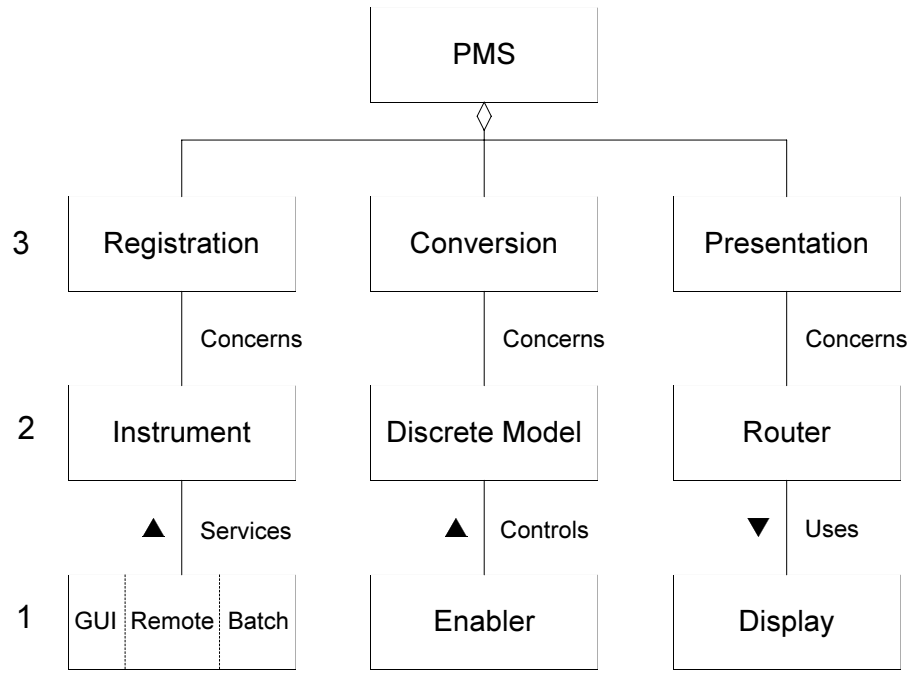

Figure 13: Composition Model, version 2

Feature 1 concerns extending the set of classes supported by PMS. Figure 13 shows how the UML class diagrams can be extended. In particular, we note that the Entity object layer in subsystems Registration and Conversion are modified to accommodate new types of financial instruments and modellers, respectively. Similarly, the Boundary object layer in Registration can be extended to include input and output devices such as graphical user interfaces, batch processing and remote input (e.g. to and from a mobile telephone).
Senior management is interested in giving each trader a copy of PMS. Each trader manages his or her own portfolios. New management information systems must be developed that allow the bank to aggregate information for Value at

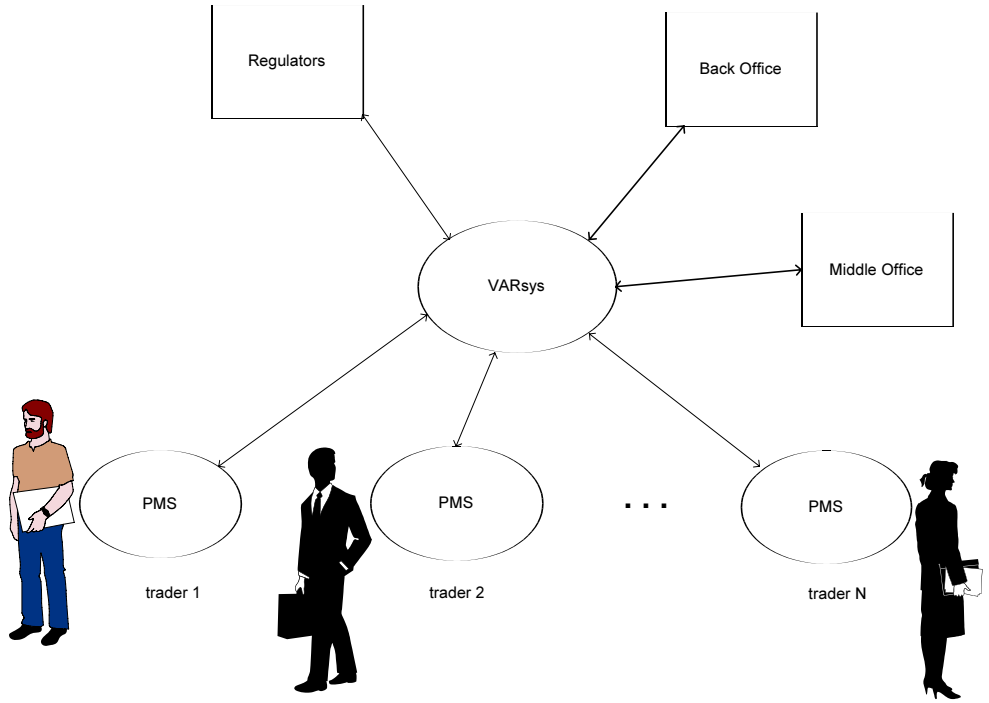

Figure 14: Value at Risk (VaR) Model

Risk (VaR) and Money at Risk applications. The configuration for such a situation is given in figure 14.

6 Conclusions

We have produced a requirements document that is based on the guidelines in [Sommerville97]. The application has to do with risk management and the approach can be adapted to other domains. Thus, this chapter can be viewed as a template that the reader can use to generate his or her own requirements documentation.

7 Glossary of Terms

We give a list of some terms and their definitions. The list is by no means exhaustive but it suffices for the purposes of version 1 of PMS.

American Option: A contract that allows the holder to exercise at any time before expiry. American options give the holder more rights then their European counterparts and hence are more valuable.
Back Office: The group of stakeholders that concerns itself with the processing of the deals that are executed by the trading operation and marking the trading books to market independently from the traders (the Front Office). This is a baseline for internal and external auditing of the trading business.
Barrier Option: A barrier option is one whose payoff is dependent on the value of the underlying price before expiration. In particular, certain aspects of the contract are triggered if the asset price becomes too high or too low. There are two many types of barrier option:

- The out option has a payoff only if a certain underlying value is not reached. If this critical value (or barrier as it is sometimes called) is reached then the option is worthless and we say that the barrier has been knocked out.
- The in option has a payoff if a certain underlying value is reached before expiry. If this barrier is reached then we say that the option is knocked in.

Furthermore, underlying prices may go up or down and hence we can characterise barrier options by comparing the

barrier with the initial value of the underlying:
- If the barrier is above the initial underlying value, we then have an up option.
- If the barrier is above the initial underlying value, we then have a down option.

Basket Option: An option with many underlying assets. The Black-Scholes formula is applicable to such options except that it is in higher dimensions.
Basket options are also known as rainbow options or options on baskets.
A special type of multi-asset option is the quanto. This is a cross-currency contract and has a payoff defined with respect to an asset or an index in one country and the payoff is converted to another currency payment.
Benchmark: A starting point in an analysis of a product, a market, or a business relative to which comparisons are made.
Bermudan Option: An option that can be exercised on specified dates or in specified periods. They can be seen as being somewhere in between European and American options in the sense that exercise is allowed on some days and not on others.
Bond: A bond is a debt instrument requiring the investor (also called the debtor or borrower) to repay to the investor (also called the lender) the amount borrowed plus interest over some specified period of time.
Book: Synonym for portfolio.
Brownian Motion Process: An example of a stochastic process that models the behaviour of many physical and non-physical entities. For example, the time-behaviour of stock prices is determined by a Brownian motion process and is determined by deterministic variables for drift and volatility and by a random 'noise' element.
Also known as Wiener process.
Call Option :A call option gives the gives the holder the right to buy a particular asset for an agreed amount at a specified time in the future.
Continuous Model: The mathematical model (for example, a stochastic equation or a partial differential equation) that gives an exact representation (in some sense) of the financial instrument whose behaviour we are interested in.
Close-of-day: The time at which trading stops at an exchange. It is at this time that prices are settled.
Delta: See the 'Greeks'
Derivative Contract: A financial contract that derives its value from some underlying asset.
Discrete Model: The model that is used to find an approximation to the continuous model. Examples of discrete models are Monte Carlo, Finite Differences and Finite Elements. In some (rare) cases an exact solution to the continuous problem can be found, in which case there is no need to resort to approximate methods.
Double Barrier Option: A special kind of barrier option with upper and lower barriers. For example, a contract becomes worthless if the underlying value reaches either of the barriers. Other combinations are possible, for example:
- The upper barrier is 'out'
- The lower barrier is 'in'

European Option: An option that can be exercised at the expiry date only and not on any dates beforehand.
Exchange: A centralised trading operation that offers standardised contracts and requiring the use of their clearing and margin account services.
Exercise: The act of executing the right conveyed by a contract. An exercise right is defined by the option contract.
Expiry Date: The date on which the option can be exercised or the date on which the option ceases to exist or give the holder any rights. Also known as expiration or maturity date.
Fair Price: The price that is to be paid for the right of exercising an option.
Front Office: The group of stakeholders that is concerned with the trading operation functions of marketing, trading and managing the trading books. In other words, the Front Office stakeholders execute the company's trading strategies.
'Greeks': A general term that relates to the derivatives of an option price with respect to its parameters. The most important ones are:
- Delta: first derivative with respect to underlying asset S
- Gamma: second partial derivative with respect to underlying asset S
- Theta: derivative with respect to time t
- Vega: derivative with respect to volatility σ.

Instrument: Synonym for Derivative Contract.
Ito's lemma: A stochastic formula that expresses the change in portfolio value over a very short time period in terms of changes in market variables and time.
Lookback Option:An option with a payoff function that depends on the realised maximum or minimum of the underlying value over some prescribed period. These options tend to be expensive. The maximum/minimum values can be measured continuously or discretely.
Market Maker: A firm that buys and sells derivative contracts.
Maturity Date: Synonymous with expiry date.
Middle Office: The group of stakeholders that concerns itself with the trading operation functions of capturing the

trading books' overall value at risk. This is a bridge between the Front and Back Office functions. The Middle Office is often the approval body for models used by the Front and Back Offices.
Monte Carlo Method: A method for simulation based on the generation of randomly distributed numbers. In this chapter we use Monte Carlo simulation to price European options.
Numerical Analysis: A branch of mathematics which is concerned with approximating continuous equations by discrete variants. The objective is to find a solution to the discrete problem (which is in theory always possible) and proving that the solution is in some sense 'close' to the solution of the continuous problem (which is not always possible to find).
Option: An option gives the holder the right to buy (a call option) or sell (a put option) a particular asset for an agreed amount at a given time in the future. We note that the holder does not have the obligation to buy or sell the asset. The amount for which the option can be bought or sold is called the exercise price (or strike price) while the date on which the option is exercised is called the expiry date (or expiration date). The stock on which the option is based is called the underlying asset.
The intrinsic value of an option is defined as the payoff that would be received if the underlying is at its current level when the option expires. The intrinsic value has three states:

- in the money: the option has positive intrinsic value. For a call option, this means that $S > K$, while for a put option this means that $S < K$, where S is the underlying price and K is the strike price.
- at the money: the strike price K is close to the current underlying value S.
- out of the money: the option has no intrinsic value. For a call option, this means that $S < K$, while for a put option this means that $S > K$.

Path Dependency: A contact is said to be strongly path-dependent if it has a payoff that depends on some property of the asset price path in addition to the value of the underlying at the present moment in time. In practical terms that means that the contract value is a function of at least three independent variables.
Payoff Function: The worth or value of an option at expiry. For a European call option the payoff is

$$\max(S - K, 0)$$

while for a European put option it is

$$\max(K - S, 0)$$

Here S is the underlying value at the expiry date while K is the strike price.

Portfolio Management: This concept is concerned with the distribution of investable liquidity across a range of available assets and liabilities with the objective of providing risks and returns that achieve performance objectives.
Premium: The amount of money paid for an option.
Put Option: This gives the right to sell a particular asset for an agreed amount at a specified time in the future.
Rainbow Option: An option that has several underlyings. Synonyms for rainbow option are: basket option, option on baskets.
Random Walk: A stochastic process in which each step taken is random and independent of the steps taken prior to that step.
Risk: Anything that cannot be predicted with complete certainty.
Scenario: An interaction session with an actor in the system. Scenarios can add to the understanding of system requirements and functionality.
Stochastic Differential Equation: An equation involving both random (stochastic) and deterministic terms.
Strike Price: The amount for which the underlying can be bought (for a call option) or sold (for a put option). Also known as the exercise price.
Underlying (Asset): The instrument on which the option value depends.
Use Case: Synonym for scenario.
Value at Risk (VaR): An estimate (with a given degree of confidence) of how much we can lose from a portfolio over a given time horizon. The portfolio can be that of a single trader or it can be a portfolio of the entire firm.
View: A stakeholder's opinion of future market behaviour.
Volatility: A measure of the amount of fluctuation in asset price. In fact, this term can refer to any entity whose value can fluctuate in some deterministic or random manner. For example, bonds and interest rates have volatility. The Black-Scholes model assumes that volatility is constant right up to maturity date while some instruments (for example, bonds) have a volatility that declines over time and reaches 0 at maturity.

Appendix 1: The Monte Carlo Method: Details and Implementation Issues

The steps in the Monte Carlo method are:

1. Simulate the risk-neutral random walk in equation (1) in section 3 starting at today's value of S. The time horizon is from today at time t to the option's expiry date which is denoted by T. This is one realisation of the underlying price path. This part of the process usually involves generating (pseudo) random numbers in order to simulate the behaviour of the Wiener process X in equation (1).
2. For this realisation of S calculate the option price by (2) in section 3.
3. Perform steps 1 and 2 a number of times over the same time horizon. This number is called the number of simulations.
4. Calculate the average payoff of all realisations based on the results in step 3.
5. Take the present value of this average using equation (2). This number is the desired option value.

We include an example to show what the results of the algorithm are. We are interested in the relationship between the values of the underlying value S and the strike price K. In particular, we simulate with a time horizon of 20 months. The following tables show the following information:

- Column 1: the current period
- Column 2: the option price for current period (raw Monte Carlo option value)
- Column 3: the standard deviation of the option value

The values obtained are similar to those in [Boyle77]. Thus, we feel confident that the code is working well.

S/K = 0.50

2	0.002446	0.001291
4	0.072702	0.011471
6	0.233232	0.025536
8	0.482631	0.042723
10	0.881517	0.067639
12	1.084691	0.082543
14	1.611326	0.096594
16	2.106002	0.121086
18	2.660823	0.145989
20	2.707084	0.157802

S/K = 1.00

2	5.250916	0.112516
4	7.271512	0.165477
6	8.597302	0.205988
8	10.403177	0.252968
10	12.152596	0.303106
12	13.090725	0.343272
14	14.361850	0.385405
16	14.903817	0.394589
18	16.500083	0.434938
20	17.171496	0.513434

S/K = 1.50

2	26.389831	0.232705
4	28.077227	0.327565
6	29.442614	0.420086
8	30.711758	0.470541
10	32.490715	0.547215
12	33.840299	0.602655
14	33.912595	0.623427
16	35.862113	0.678127
18	36.120994	0.734826
20	37.490218	0.766864

These three datasets correspond to the call option being out of the money ($S < K$), at the money ($S = K$) and in the money ($S > K$).

The Monte Carlo is one of a number of simulation techniques that are used in the financial markets. We mention the

most important ones but a discussion of these is beyond the scope of this chapter (see [Deventer97] and [Wilmott98] for more information):
- Analytical solutions
- Finite difference methods (see [Duffy80], [Cooney99])
- Binomial lattices
- Bushy trees
- Trinomial lattices

Summarising, the advantages of the Monte Carlo method are:
- It is sometimes the only alternative when there are path dependencies or when there are three or more random variables.
- It is relatively simple to implement
- It has speed advantages for problems with three or more variables.

The disadvantages of the Monte Carlo method are:
- Its convergence rate is proportional to the square root of the number of sampling points.
- It seems to be limited to European-style options.
- It does not use all scenarios for evaluation.
- It has sampling errors. In order to reduce the error we must increase the number of simulations, thus slowing down performance.
- The 'delta' (the derivative of the option price with respect to the underlying price) cannot be deduced from the data from the simulation. In this case we must do the calculation twice.

Appendix 2: Some common Stochastic Models

The Merton Model
In this case we assume that the underlying asset follows a simple random walk with zero drift:

$$dr = \sigma\, dZ$$

Here dr is the change in the interest rate r and Z represents a standard Wiener process with a mean of 0 and a standard deviation of 1.

The Extended Merton Model
This is similar to the basic Merton model but we add a time-dependent drift term a(t):

$$dr = a(t)dt + \sigma\, dZ$$

where Z is the same variable as in the Merton model.

The Vasicek Model (Ornstein-Uhlenbeck process)
Some of the limitations of the Merton and extended Merton model were resolved by assuming that the short-term interest rate r has a constant volatility σ as before and that it also exhibits a 'mean reversion' phenomenon:

$$d = \alpha(\mu - r)dt + \alpha\, dZ$$

where

r = the instantaneous short rate of interest
α = the speed of mean reversal
μ = the long-run expected value of r
α = the instantaneous standard deviation of r

References

[Bhansali98] V. Bhansali '*Pricing and Managing Exotic and Hybrid Options*' McGraw-Hill New York 1998.
[Boyle77] P. Boyle '*Options: A Monte Carlo Approach*', Journal of Financial Economics 4, pp. 323-338.
[Boyle97] P. Boyle, M. Broadie, P. Glasserman '*Monte Carlo Methods for Security Pricing*' Journal of Economic Dynamics and Control.

[Buschmann96] F. Buschmann et al '*Pattern-Oriented Software Architecture*', John Wiley & Sons Chichester 1996.
[Clements94] P. C. Clements '*From Domain Models to Architectures*' Workshop on Software Architecture, USC Center for Software Engineering, Los Angeles 1994.
[Cooney99] Michael Cooney '*Benchmarking Numerical Solutions of European Options to the Black-Scholes Partial Differential Equation*', M.Sc. Thesis Department of Mathematics, Trinity College, Dublin, Ireland 1999.
[Cox85] J. C. Cox, M. Rubinstein '*Options Markets*' Prentice Hall, Englewood Cliffs, New Jersey 1985.
[Datasim99] Datasim Education BV '*Advanced Design Patterns*' course material (see http://www.datasim.nl)
[Deventer97] D. R. van Deventer, K. Imai '*Financial Risk Analytics*' McGraw-Hill New York 1997.
[Duffy80] Daniel J. Duffy '*Uniformly Convergent Difference Schemes for Problems with a small Parameter in the highest Derivative*', Ph.D. thesis Trinity College, Dublin 1980.
[Duffy95] Daniel J. Duffy '*From Chaos to Classes*', McGraw-Hill London UK 1995.
[Gropp97] W. Gropp et al '*Using MPI, Portable Parallel Programming with the Message-Passing Interface*', MIT Press Cambridge MA 1997.
[Jacobson99] I. Jacobson et al '*The Unified Software Development Process*', Addison-Wesley Reading MA 1999.
[Nasa92] National Aeronautics and Space Administration (NASA) '*Recommended Approach to Software Development, Revision 3*', Software Engineering Laboratory Series (SEL-81-305) 1992.
[OMG96] Object Management Group (OMG96) '*Property Service Specification*', see http://www.omg.org
[Sommerville97] I. Sommerville, P. Sawyer '*Requirement Engineering: A good practice guide*' John Wiley & Sons Chichester UK 1997.
[Stunkel95] C. B. Stunkel et al '*The SP2 High-Performance Switch*', IBM Systems Journal Vol. 34, No. 2, 1995, pp. 185-204.
[Wilmott98] P. Wilmott '*Derivatives: The Theory and Practice of Financial Engineering*', John Wiley & Sons Chichester UK 1998.