System Development for Flood Hazard Management
– at a local level scenario –

Francis Simon Mwakapuja
February, 2003
System Development for Flood Hazard Management
– at a local level scenario –

by

Francis Simon Mwakapuja

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree in Master of Science in Geoinformatics.

Degree Assessment Board

Thesis advisor  
W. H. Bakker M.Sc. (First supervisor)  
M. C. Ellis M.Sc. (Second supervisor)

Thesis examiners  
Prof. Dr. A. Stein (Chairman)  
Dr. Ir. B. G. H. Gorte (External examiner)
Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation (ITC). All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.
Dedication

To the Most High God Jehovah El-Shaddai ...
“... Have I not commanded you Francis? Be strong and courageous. Do not be terrified; do not be discouraged, for the LORD your God will be with you wherever you go.”
(personalised from Joshua 1:9 NIV)

To my three ladies ...
my loving wife...my heartthrob...Juliana,
my daughters Atuganile (Caroline) and Diku (Calister).
Thank you for being so understanding.
Abstract

Specific application systems are an essential part in the whole process of geographical information system (GIS) implementation. Research on effective application system design, delivery, use and its impact to an organisation is very crucial and it is a foundation of the whole GIS investment in any domain. This research work deals with the system development for flood hazard management at local level.

Natural hazards are often recurring phenomena, the systems which meant to deal with the same must meet the domain requirements, be effectively developed and flexible to changes. The key to develop an effective natural hazard management system is to make it serve many users. An effective strategy is to make a generic design, which will incorporate reusable models, design patterns, specifications, and project plans.

The advantages of using object-oriented paradigm incorporating UML is investigated for the generic case of the flood hazard management at the local level. Different methodologies for the system development on the object-oriented platform are investigated, and the Unified Methods techniques are used in this research to build packages and components with priority in generic scenarios.

The basic aim of using UML diagrams in modelling is to convey important information, thus they were used very selectively in this research. This was done in order to avoid chaotic and overcrowding of details in a single diagram, yet this was done with important features maintained.

While object-orientation modelling and flood hazard management system development are distinct processes, the two were not discrete in this research; modelling using UML impact both and join them in flood hazard management system realisation. Moreover UML was used as a tool to join the different stages of development process seamlessly.

Keywords

GIS, natural hazards, flood hazard management system, object-oriented paradigm, UML, Unified Methods, development process.
Abstract
# Contents

Abstract iii  
List of Figures ix  
List of Tables xi  
Acknowledgements xiii  

1 Introduction 1  
1.1 Background 1  
1.2 Research Problems and Motivation 1  
1.3 Research Objectives 2  
1.4 Research Questions 2  
1.5 Thesis Structure 3  

2 Background 5  
2.1 Natural Hazards Management 5  
2.2 Related Works 6  
2.3 Flood Hazard 9  
2.3.1 Flood Hazard Management at Local Level 9  
2.3.2 Data and Relevant Information 10  
2.4 Conclusion 10  

3 Resources 11  
3.1 Concepts 11  
3.2 Object–Orientation 12  
3.3 Unified Modelling Language (UML) 13  
3.4 Discussion 14  
3.5 Conclusion 14  

4 Methodologies for System Development 15  
4.1 Object–Oriented Methodologies 15  
4.1.1 The OOSE Method 15  
4.1.2 Grady Booch Method 17  
4.1.3 The OMT Method 17  
4.1.4 The Unified Methods 17  
4.2 GRAPPLE Method 19  
4.3 Unified Process 22
## Contents

### A System Development
- A.1 Actors Generalisation with their Details .......................... 75
- A.2 Use Cases with Details ............................................ 76
- A.3 Activity Diagram ................................................... 77
- A.4 Class Diagrams ..................................................... 77

### B UML Diagrams
- B.1 Class, Objects and Models ......................................... 83
- B.2 Analysis Class Stereotypes ......................................... 84
- B.3 Component Stereotypes ............................................. 84

### C Glossary

Bibliography ........................................................................ 89

Index .................................................................................. 93
List of Figures

3.1 Models in database design ................................. 12
4.1 Object orientation generic development cycle .............. 16
4.2 The OOSE development cycle ............................. 17
4.3 Process proposed by Grady Booch ....................... 18
4.4 The OMT original modelling technique ................... 18
5.1 The UML use-case model for layer updating using Update use case ................................. 28
5.2 The UML activity diagram for layer updating action in the flood hazard management system ....................... 29
6.1 The analysis model and its packages containing analysis classes and use-cases realisations ................................. 32
6.2 Standard class stereotypes as used in flood hazard management system analysis ....................... 32
6.3 Finding analysis packages from use cases ................ 34
6.4 Finding analysis packages that share the same analysis classes ................................. 35
6.5 The service packages, Update and Analyse, each encapsulating functionality related classes ................................. 35
6.6 Dependencies and layers of analysis packages ........... 36
6.7 A class diagram for realisation of Maintenance use case ................................. 36
6.8 A class diagram for realisation of Update use case ......... 37
6.9 A collaboration diagram for a realisation of the Update use case ........................................ 38
6.10 An association instances between analysis classes ....... 40
6.11 The Confirmation class generalises Analyse and Update Confirmations classes ................................. 41
7.1 UML Deployment diagram for the Updating use case in flood hazard scenario ................................. 45
7.2 The client/server configuration in the spatial database for flood hazard system ................................. 46
7.3 The UML diagrams showing subsystem packages ........ 47
7.4 Loosely coupled architecture systems .................... 50
7.5 The UML diagrams; design classes tracing to analysis classes in their respective models ................................. 51
List of Figures

7.6 The UML diagrams showing classes in Update use case and their associations. .................................................. 52
7.7 The UML sequence diagram for the generic design objects in Update use-case realisation. .......................... 53
7.8 The UML sequence diagram for the specific design objects in Update use-case realisation. ......................... 54
7.9 The UML models for; Display's LayerWindow subtype, and LayerWindow screen shot package with components location and use cases connections. ......................................................... 57
7.10 The flood hazard user interface models for the Layer Update use case window. ............................................. 58
7.11 UML modelling of association design classes for Updating using Clip process in flood hazard scenario. .......... 60
7.12 UML Statechart diagram for the updating process from Update class which is supertype of Layer class. ........ 61

8.1 UML diagrams; (A). Active class identifies executable component, (B). Active object allocated to node with corresponding deployment component. ...................................................... 64
8.2 Design classes for a build are implemented by components and an interface provided by an implementation subsystem. ................................................................. 66
8.3 The input and result of the design test. .............................................................................................................. 68
8.4 The input and result of the integration or system test ...................................................................................... 69
8.5 The input and result of test evaluation. ............................................................................................................. 70
8.6 First part sequence diagram for the Updating use case realisation. .............................................................. 70

A.1 Part one of UML activity diagram for generic flood hazard management scenario. ......................................... 78
A.2 Part two of UML activity diagram for generic flood hazard management scenario. ....................................... 79
A.3 Part three of UML activity diagram for generic flood hazard management scenario. ..................................... 80
A.4 A class diagram for realisation of Display use case. .......................................................................................... 81
A.5 A class diagram for realisation of StartUp use case. .......................................................................................... 81
A.6 A class diagram for realisation of Analyse use case. .......................................................................................... 81
B.1 The UML class, object and package representations. .......................................................................................... 83
B.2 Standard class stereotypes used in analysis. ..................................................................................................... 84
List of Tables

7.1 Visibility options on UML class diagrams. . . . . . . . . . . . . . 59
7.2 UML multiplicity indicators. . . . . . . . . . . . . . . . . . . . . . . 60
Acknowledgements

I wish to express my sincere gratitude and appreciations to all individuals, who in one way or another have assisted me in my studies at ITC.

Above all, to God Almighty, for the grace that has made me make it throughout my studies.

My gratitude goes to:
The Dutch Government, for granting me the fellowship through the Netherlands Fellowship Programme (NFP) and ITC for all arrangements for my study.
My organisation, UCLAS in general and Land Surveying Department in particular, for giving me a study leave.

My appreciations and thanks to:
My first supervisor, Mr. Wim Harry Bakker, for his constructive criticisms, suggestions and valuable advices ... Wim, I have learned how to swim!
Mr. Martin Ellis, my second supervisor, for his productive comments and suggestions during the research period.

My special thanks to:
My course mates, GFM2 2001 students, for their friendship and company ... hopefully this is not the end of our get-together-parties!
The fellowship of the believers, at Pniel “De Rots” Church and ITC–Evangelical Fellowship, for the sharing of the daily manna.
The Dutch families; Adrie and Ginny van Dorst, W. J. F. Grol and A. C. Grol–Overling (Baba na Mama Furahisha), Pastor Henk Bentveld, Gerrit and Marjatta Kersten who made my stay in The Netherlands worthwhile.
Brother Levi Shimi Mutambo for our reversible role of Barnabas towards each other ... especially during the last six months!
My extended family of Tanzanian students at ITC and University of Twente ... dank u wel for your ushirikiano mkubwa!
Last, but in no way least, my wife Juliana and my daughters Atuganile (Caroline) and Diku (Calister) for their intercessing, encouragement and understanding during my eighteen months of career pursuit.

May God bless you all...

THANK YOU!
Chapter 1

Introduction

1.1 Background

Geographical Information System (GIS) is becoming common in various applications, implementing GIS technology improves the effectiveness of the specific operation.

In order to utilise an information system fully, an investment in specific application is important, this can incorporate databases which keep data and information that are essential in supporting processes in real-world systems as discussed in Flavin [12].

Natural hazard management systems are the mechanism whereby organised collections of spatial and attribute data about specific hazard are stored with their specific structures for features, collection of features, attributes, relationships between attributes and relationships between features. Spatial databases are essential tools to implement, maintain and expand a GIS capability for an application.

The natural hazard spatial database within an organisation has to accommodate data from outside agencies, projects and other sources so as to create GIS data sets for analysis/processing.

Recently a lot of investment has been injected in the GIS technology worldwide, but now the challenge is in integrating the systems.

1.2 Research Problems and Motivation

Less accurate and incomplete data sets have proved to be ineffective and inefficient when there is a need of integrating with other data sets with aim of providing information for specific applications.

Research suggests, as cited in ESRI technical report [47], that about 80% of the effort of the GIS application process involves locating, gathering, reformatting, and compiling the information to be used for decision making.

Researches have confirmed that various GIS tools are being under utilised in the planning field, among the reasons being; complexity of technology, lack of trained staff, incompatibility of GIS products with the tasks and functions performed by planners as cited in URISA Journal [30].
1.3 Research Objectives

In order to overcome these problems and at the same time to implement GIS technology fully in an application, an investment in spatial databases is vital. Development of flood hazard management system in a systematic and comprehensive way is a challenge for this research work. The research will deal with diverse users, with different interest and line of profession, in flood hazard scenario at local level.

1.3 Research Objectives

The objectives of the research are:

• Generation of an effective and improved data handling system for the flood hazard management at local level.

• Development of spatial database based on planning problems, process, and context [26] in flood hazard scenario.

Development will be for a flood hazard generic case with means to manifest it into specific flood hazard scenario; means to facilitate the same will be explored.

1.4 Research Questions

The research objectives will be achieved by tackling the following questions;

1. What are the user requirements?

2. Data and information.

   • What types of data/information are to be stored/used?
   • How are data to be handled and what are the constraints of the processes?

3. How spatial database technology will be used efficiently in flood hazard management processes?

4. How Unified Modelling Language (UML) will be used in modelling the flood hazard management system?

5. How flood hazard management system assessment/validation will be realised?

These questions will be covered so as to produce stable generic flood hazard management system design. This design is to meet user requirements and to be able, with minor alterations, to fit specific cases.

(Nota Bene: In this research work Flood Hazard Management System at Local Level will be sometimes referred to as Flood Hazard Management System).
1.5 Thesis Structure

The current chapter, Chapter 1, gives the background, purpose and scope of this research work. Research problems, motivation, objectives and a list of research questions are also covered.

The following is a summary of what is in the other chapters and appendices:

Chapter 2 deals with the literature review on different natural hazard mitigation projects. The analysis of what worked elsewhere are used to define how the System for Flood Hazard Management at Local Level will be developed, and the expected products from the system.

Chapter 3 discusses the characteristics of the Object–Orientation paradigm and the Unified Modelling Language (UML), which justify the incorporation of those concepts in the research work.

Chapter 4 explains different methodologies with their strengths and/or weaknesses. It states why Unified Methods in general, GRAPPLE Method and Unified Process in particular were chosen for this research work.

Chapters 5–8 relates to the phases of the flood hazard management system development, namely; requirements, analysis, design and development. It covers the methodologies used and findings in each phases.

Conclusions is covered in Chapter 9, it contains conclusion in line of the concepts and methods used in this research work.

Appendix A contains certain outputs from flood hazard management system development phases in details, with certain descriptions not in main text.

Appendix B is about UML Diagrams.

Appendix C Glossary.
1.5. Thesis Structure
Chapter 2

Background

2.1 Natural Hazards Management

The Organisation of American States (OAS) report on natural hazards management [35] lists the tools for natural hazard assessment as Geographic Information System (GIS), Remote Sensing and Special Mapping Techniques. The report state that the volume of information needed for hazards management exceeds the capacity to deal with them manually; this compels the use of computerised techniques.

GIS can play a major role in collecting, organising, analysing and presenting data. The information to be assembled for hazards management will be determined by the level of application, which are:

National level  Provision of a general familiarity with the study area, giving planners a reference to the overall hazard situation.

Regional level  It can be used for resource analysis and project identification.

Local level  It can be used to formulate investment projects and specific mitigation strategies.

Also the use of data/information gathered, whether;

- natural hazards assessments,
- vulnerability assessments,
- disaster preparedness and response or
- post-disaster relief and reconstruction activities.

There are three types of information which are essential for the hazards management tasks;

Natural hazard information  shows the presence and effects of natural phenomena. This includes location, severity, frequency and probability of occurrence of a hazardous event.
2.2. Related Works

**Natural ecosystems information** provides the basis for estimating the effect and factors/conditions that create, modify, accelerates and/or retard the hazard occurrence. This includes slopes, river flow capacity, vegetation cover.

**Population and infrastructure information** is for quantifying the impact of hazardous events on existing and planned development activities.

For this research work, the focus is on the local level and all the above types of information are covered. This is because at local level the mitigation measures are dealt with, and usually that is where different development planning and projects are formulated.

In order to have full understanding of the hazard management environment certain related works were studied.

2.2 Related Works

**Integrating Risk Assessment in the Pipeline Industry**

Risk assessment, in the pipeline operations, is a decision support process which is affected by a number of factors; one of these factors is data quality [47].

The validity of the risk assessment depends on the data quality, which is input to the system, and the quality /execution of the model used in result generation.

Over the life cycle of a pipeline facility, vast quantities of information are acquired from different sources in various formats. The *Enterprise Information Management Strategy (EIMS)*, in ESRI technical report [47], provides framework for well managed and easily accessible collection of all information regarding pipeline, the operating environment and operational history; it also encompass statistical packages/GIS and document management.

**Disaster Management System to Monitor El Niño in Peru**

GeoSolutions Consulting provided expertise in the systems designed to aid the monitoring and management of the effects of natural disasters. The 1997/98 El Niño weather event caused record levels of precipitation in Peru, this led to flooding along rivers, which resulted in the displacement of thousands of citizens. As a secondary consequence, the saturated hillsides became susceptible to landslides, which resulted in further loss of property and life.

Risk areas mapping enabled emergency response planning for El Niño related disasters in two areas of Peru. The success of this project emphasises the essential role of GIS technology in disaster relief efforts.

Data of flood extents and *Digital Elevation Models (DEMs)* were derived from RADARSAT scenes, collected during peak and post-flood periods. Classification of multispectral LANDSAT imagery and an IRS MSS scene provided land cover data for the study areas.
Chapter 2. Background

A GIS database was created by using data from various sources; also images for study areas were classified, interpreted and integrated into the GIS as explained in the Case Study report [14].

GIS techniques, based on geology, vegetation, bio-climactic, slope and aspect data produced a Landslide Hazard Prediction Map for study areas.

A flood extent map for February 1998 was defined by using the LANDSAT image of October 1997, overlayed with February and March 1998 RADARSAT images on the DEM.

Various risk maps were modelled and presented to demonstrate probable impacts of flood-related hazards; flood related landslides to transportation networks, services and structures; risk by landuse classification and population density.

Applications of IT in Risk Management

Applications of GIS and the use of Information Technology (IT) in risk assessment in the Bandung Urban Earthquake Mitigation Project, within the framework of the Asian Urban Disaster Management Project is discussed in paper by Pribadi et al [39].

The GIS role is in the management of all spatial information related to hazard, exposure and the vulnerability of the city. Information on topological, geological conditions, and also infrastructures are stored in the GIS database.

Various risk parameters are combined, with GIS support, to produce risk maps of the city at sub-district level. ArcInfo and ArcView software are used in the data analysis. The risk maps are then used for the review of the city’s spatial development and land use planning, and for raising awareness among the stakeholders (actors) in the city development process.

The role of IT is in providing the media of communication and dissemination of information to the actors and those interested in the project. A national network on urban disaster management and mitigation was established by the project.

Information Systems for Earthquake-resistant Cities

MSc thesis by Özge Yalçınier [48] defines and finds an interrelation between an urban information system and earthquake management concepts. The major aims of this research are;

- to build a body of knowledge and awareness about how to tackle the earthquake disaster,
- to create an urban inventory for pre- and post-disaster planning and
- to propose an emergency administration system.

Earthquake hazards affect a vast area, and pose great risk to human life and infrastructure. Earthquake risk depends on settlement distribution and densities, building materials and engineering standards, et cetera. Policies and
decisions must be designed in order to act effectively against earthquakes, to manage rescue and response operations, to organise and deliver relief.

A case study area for this research was an earthquake-prone district of Pendik in Istanbul, Turkey.

**Information Infrastructure for Disaster Management**

The Australian Geological Survey Organisation [15] report is on information needed for disaster management in the Pacific Island Countries (PICs), and the nature of the information infrastructure needed to ensure the delivery of that information.

Basically the report provides a guide to follow for disaster management and research projects.

Furthermore, they make some observations on a range of technical and organisational issues, such as data formats, transfer standards and custodianship arrangements, that need to be considered in establishing and operating any modern information infrastructure.

**Related Works Influence on the Research Work**

From the above related works the following were observed:

- Different technologies are used with a common platform and there is a possibility to integrate datasets and various tools.
  
  This compels the use of an object–oriented paradigm for this research. The object–oriented approach manipulates objects in their properties, operations, relationships, conversions, generalisation and specialisation in order to show their behaviours. This will make objects to be modelled for integration with other objects [7].

- IT usage in providing the media of communication and dissemination of information to the actors and other stakeholders.
  
  In order this to be effective, an open protocol which is recommended by Open GIS Consortium (OGC) is to be implemented for the flood hazard management system [10, 9, 34, 40].

- Data quality and accessibility are key factors for the hazard management process.

- Users in the hazard management scenarios are from different disciplines.
  
  These users have different professional background, thus it is essential to have a flood hazard management system which will cater for urban planners [48], engineers, insurers and decision makers belonging to the local authorities [35, 36].

These findings will be used to find the user requirements, and type of data to be accommodated in flood hazard scenario in the research work. Furthermore
the flood and earthquake scenarios will be compared in modelling the spatial database at local level.

2.3 **Flood Hazard**

Prime concern of flood hazard management are data and information on flood hazard, demographic data, and location of population [35, 36].

When these information are combined with changes in land use data, an understanding will be created of population distribution and kind of activities being undertaken by masses. With these information disaster prevention and preparedness actions (e.g. identification of vulnerable areas, construction control codes) can be initiated [48].

Integration of the floodplane information into a planning study requires the definition of floodplane and flood-prone areas together with the probability of a flooding event.

Floodplane are areas which are relatively flat and lies adjacent either to a river, channel, or stream that is susceptible to flooding; while flood-prone areas are those floodplanes which are mostly subject to recurring floods and are hazardous to development activities [35, 36].

Flood hazard information will be used in making decisions about the level of risk and mitigation measures to be used to minimise the impact of flooding. Such measures include:

- adopting land-use classification and zoning systems,
- building codes,
- taxation,
- insurance programs,
- and public awareness.

2.3.1 **Flood Hazard Management at Local Level**

The flood hazard management processes, at local level, consists of activities carried out before, during, and after a flood event in order to reduce loss of life and destruction of property. This activities are different and sometimes depend on the location of the flood-prone area [35, 36].

These processes can be grouped into; pre-event measures, measures during/immediately after natural disasters and post-disaster measures.

Planners, by using GIS tools, can formulate projects at the feasibility level to locate vulnerable network elements for the implementation of emergency preparedness and response activities. The flood hazard presence will effect the site selection, engineering design and economic feasibility of investment projects. Lifeline networks (made of important installations necessary for the public health and safety) are the critical elements of a location. The area occupied by the lifeline network should be made as little vulnerable to damage
2.4. Conclusion

as possible (by implementing mitigation measures) or be recognised as priority elements for rehabilitation following a flood disaster.

Lifeline network includes locations of potable water and sanitation systems, police and fire stations, military posts, emergency management facilities, emergency shelters, and medical services [35, 36].

GIS has to be used in preparation of lifeline network maps, these maps can be combined with flood hazard information for determination of the most vulnerable segments and the identification of mitigation measures and disaster preparedness activities.

2.3.2 Data and Relevant Information

Firsthand sources of information for flood hazard includes maps, documents, field observations and remote sensors.

Data may be available in the form of satellite images, weather satellite data, aerial photographs, topographic/soil maps, and population distribution maps. These are initial data, there is a possibility of adding new items once the flood hazard management system is developed.

The data/information will be essential in:

- Developing data models.
- Developing spatial database prototypes.

Analysis and design based on true data structure will produce stable information systems and data storage designs for the present and future user requirements [28].

2.4 Conclusion

There are different ways of obtaining requirements for the flood hazard management system to be developed (e.g. studying business domain models, conducting interviews), for this research work literature review covered in this chapter will be used for the same.

Various literature covering different hazard scenarios at different levels were covered, the next chapter is about the concepts and methods to be used to develop the flood hazard management system for local level (referred to as flood hazard management system in the thesis) based on the literature review covered.
Chapter 3

Resources

This chapter is about the resources, to be used in developing a flood hazard management system; these resources are in the form of the concepts and methods. In order to realise a flood hazard management system's analysis and design, it is essential to rely on formal methods; these methods are based on specific concepts and processes. This chapter deals with object-orientation concepts and the Unified Modelling Language (UML), which will be used as a tool to achieve the research objectives.

3.1 Concepts

In spatial database design, four stages are to be dealt with in order to achieve research objectives; requirement analysis, conceptual design, logical design, and physical design.

**Requirement analysis** deals with analysis of the organisation needs; it is not concerned with any constraints about those needs. The only constraint to be observed at this stage is the way the organisation operates.

**Conceptual design** is primarily concerned with semantics of the application prior to logical design, this levels takes-off after obtaining the results of organisational-specific analysis.

**Logical design** corresponds to spatial database development in specific structure, for this case it will be on the object–oriented paradigm.

**Physical design** is concerned with specifications on how data are to be physically stored and the methods to be used for accessing them.

The design process practically starts at requirement analysis, then goes to the conceptual stage, then to the logical, and finally to the physical, with each stage primarily independent of each other. This procedure is based on *top-down structured design* (Figure 3.1), and if it happens that certain refinements has to be done at particular stage then iterations has to be carried out to make proper adjustments.
This research is based on the object-orientation paradigm which provides better concepts and tools to model and represent the real world; it allows a direct representation of the real world model in the code. The result is that the normal radical transformation from flood hazard management system requirements (defined in user’s terms) to flood hazard management system specification (in computer terms) is greatly reduced [38]. The following section covers the object-orientation concepts.

### 3.2 Object–Orientation

The three fundamental aspects of the object-orientation paradigm are encapsulation, inheritance, and object identity [38, 23].

**Encapsulation** is any mechanism which separates a variable from code which changes its state. It implies a form of access protection, all data are shut-off from the outside world, and only special procedures called methods or member functions are allowed to access the encapsulated data [29].

**Inheritance** is the process of obtaining properties through a relation such as parent-child or general-specific, it is the act of reusing properties of superclasses.
Object Identity is a handle which distinguishes one entity from another, it is internal to an object, it provides a way to represent the individuality of an object independently of how it is accessed.

Advantages of object-orientation;

- The code is more general and flexible.
- Encapsulation means greater opportunity for re-use.
- Maintenance is easier because all message-passing is centralised and need not to be changed. Polymorphism means different methods can be written for different objects, but the calling routine's message remain the same.
- Objects, may come and go but the basic structure remains static. This opens up opportunities for the reuse of design and code.

For this research, the advantages of re-using the design/code will be utilised in obtaining a spatial database model for specific flood scenario from a generic one.

3.3 Unified Modelling Language (UML)

The Unified Modelling Language (UML) is used in specifying, visualisation, and documenting models. UML-based tools can be used to analyse application's requirements and design a solution, the results can be represented using UML's diagrams.

UML uses twelve types of diagrams, which are in three categories [17]:

**Structural Diagrams** These represent static application structure, they include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

**Behaviour Diagrams** Different facets of dynamic behaviour are represented by these diagrams, they include the Use Case Diagram, Sequence Diagram, Activity Diagram, Collaboration Diagram, and Statechart Diagram.

**Model Management Diagrams** Ways of organising and managing an application modules are represented by these diagrams; they include Packages, Subsystems, and Models.

The advantages of UML includes;

- Possibility to model any type of application, running on any type of hardware, operating system, programming language, and network.
- Distributed applications that use any middleware on the market can be modelled.
- It is ideal for object-oriented languages and environments such as C++ and Java, also it can be used to model non-object oriented applications as well, for example, Fortran, VB, or COBOL.
• UML Profiles (subsets for specific purposes) can be used to model transactional, real-time, and fault-tolerant systems.

For this research work UML will be used, the main advantage of incorporating UML’s set of tools in this research is; analysis and design models can be directly converted into implementations, that is, the ability to start at user level, then directly to a systems specification and implementation using the same modelling method.

3.4 Discussion

The conceptualisation of the flood hazard management system, using the object-oriented paradigm is independent of any programming process that may follow after modelling [6].

The object-oriented modelling emphasis is on understanding the objects components, their attributes, and the relationships exhibited to each other. By using inheritance, a subclass can derive from existing superclass, which may be available as off-the-shelf; the effort of implementing the whole class is saved, only specifications of the incremental refinements (i.e. additional structure or functional details) are necessary in the subclass [6].

In object-oriented modelling, inheritance implies reusability of implementations. Inheritance becomes a vehicle for specifications reuse; by creating new subclasses, the flood hazard management system can reuse the specifications already made for superclass earlier in hierarchical or by other similar systems.

By developing the flood hazard management system in accordance to the functions/services it provides, the changes in individual services can be inhibited. This approach yields a robust flood hazard management system that is resilient to changes. This is an illustration of how stereotypes may be used and exploited within UML.

System functionality from the user perspective in flood hazard management system will be captured by UML effectively.

UML supports custom extensions through stereotyped elements; for this research work it is flood hazard generic case, which is to be made specific for Enschede Municipality.

3.5 Conclusion

The object-oriented paradigm provides a platform to be used in the flood hazard management system development, the object-oriented paradigm is a better representation of the real world model. The development and design process have different stages; requirements, conceptual, logical and finally the physical stage. These stages are modelled individually using Unified Modelling Language (UML), which also will be used to glue the different stages seamlessly for realisation of flood hazard management system.

The next chapter covers methodologies to be used in implementing the resources discussed in this chapter in the research work, these are the tools to assist in achieving the research objectives.
Chapter 4

Methodologies for System Development

A methodology is defined as a collection of procedures, techniques, tools, and documentation aids, which helps the system developers to implement information systems [3].

It is summarised by Yourdon [49] as a component of the system for building an information system and interfaces between those components. Thus, it can be termed as a specification of a system for building systems.

The methodology consists of phases, which gives guidelines in choosing appropriate techniques at each stage of the development of the information system.

Methodologies can be distinguished by the modelling representations used, the techniques used to convert from a model at one development phase to a model at next stage, and the design processes [19].

4.1 Object–Oriented Methodologies

The integration of object–oriented design with the development process have four major phases as shown on Figure 4.1, similarity to other methodologies lies within concept formation phase.

The object modelling methods start at system requirement phase, here the methods are used to create the requirements and analysis models; the former is the collection of use cases and the latter is composed of object classes. A design model is then realised at the system design phase, it is made up of object classes, which are to be implemented using object–oriented languages.

The next sub-sections describe four methodologies using object–oriented paradigm.

4.1.1 The OOSE Method

The Object Oriented Software Engineering (OOSE) method was developed by Jacobson et al [19]. This is a use case driven method and it uses the five models as shown in Figure 4.2.
4.1. Object–Oriented Methodologies

The five models are:

**The domain object model;** which defines the standard subject terms used in an application.

**The analysis model;** which comprises of the entity, the interface and control objects.

**The design model;** which adapts the object model to the implementation environment.

**The implementation model;** for the system implementation.

**The test model;** for the system verification.

These models are used at different development phases, and there are conversion rules from one phase to the next. The conversion from one phase to another disqualify the OOSE method for this research work.
4.1.2 Grady Booch Method

This method, by Grady Booch [19], describes representation that deals with development stages, but the strength is in the design stage of the development process. This method is more open than OOSE and it does not propose steps, rather it define various notations and models and gives the relationships between them. One of the suggested process is shown in Figure 4.3.

The major models in this methodology are; the class diagram, object diagram, interaction diagrams, state transition diagrams and the module diagram.

Since this method gives a general defined development process that starts with conceptualisation then the development of an analysis model using mostly class diagrams; this generalisation removes the use of this method in this research work.

4.1.3 The OMT Method

The Object Modelling Technique (OMT) method was revised by Rumbaugh [42] from the original version which was proposed by Rumbaugh and others [43], the original version is shown in Figure 4.4.

The method concentrates only on developing an analysis model, it includes; the object diagram, the dynamic model and the functional model.

Since this method concentrates only on analysis part of the development process it was disqualified for this research work.

4.1.4 The Unified Methods
4.1. Object-Oriented Methodologies

These are developed methods that unify different existing techniques into a process that supports all the development process phases. They combine the strengths of other methodologies, to form a standard methodology that supports the whole development cycle.

Although these methods have been created intentionally for software deve-
lopment support and object–oriented programming support, they can be used profitably for flood hazard management system development [7].

**Fusion Method** Fusion uses the best aspects of different methods, including OMT, Booch, Class Responsibility Collaborator (CRC) and Formal Methods (based on the application of discrete mathematics; set theory, logic, *et cetera*) [8].

**GRAPPLE Method** The Guidelines for Rapid APPlication Engineering (GRAPPLE) incorporates ideas from different methodologies including Rational Unified Process created by Grady Booch, James Rumbaugh, and Ivar Jacobson [44].

**Unified Process** Unified Process, it is the end product of three methods; Objectory Process (released in 1987), Rational Objectory Process (released in 1997) and the Rational Unified Process (released in 1998). This method is a generic one (which can be used for different applications), it is use-case driven, architectural centric, iterative and incremental. Also it is a component-based, which means that it is made up of components inter-connected through interfaces. UML is used throughout the process and it is an integral part of the methodology [21].

**Why Unified Process and GRAPPLE?**

Unified Process and GRAPPLE methodologies will be used in research work, the combination will be used in such a way that only the strengths from each method will be incorporated at each stage.

These methodologies does not require techniques to convert models from one development stage to another, as object orientation is used into the entire development process (and not in one or two process phases as with other methods). This integration offers the usage of object orientation approach throughout the development process.

UML's diagrams and structure are used as tools throughout the development process.

These methods are generic while Fusion Method in section 4.1.4 on page 17 is based on specific application.

The following sections will deal with these methodologies in detail.

**4.2 GRAPPLE Method**

This section will describes some representations and techniques to be used by this methodology at different stages of development cycle, most of the materials used are from Schmuller [44].
4.2. GRAPPLE Method

Structure

GRAPPLE consists of five stages, each stages consists of a number of actions. Each action, under specific player, gives out a product.

The stages are; requirements, analysis, design, development and deployment.

After requirements, analysis and design stages the products are combined into a design document ready for development and deployment stages. When all the stages are complete, the stage products sum up to form a document that defines the system.

Requirements

This stage is about understanding the essential of the user's domain and the problems to be solved.

Business Process and Domain Analysis These actions are about concepts (not the system to be built), the products at this level are activity and class diagrams.

Cooperating Systems and System Requirements This is about the systems that will depend on each other with their requirements, the products are deployment and package diagrams.

Analysis

The results from requirements stage are used for more understanding of the problem.

System Usage This is a high level use case analysis, the product is a set of use case diagrams that shows actors and dependencies between use cases.

Class Diagrams Refinement The following are filled in; names of associations, abstract classes, multiplicities, generalisation, and aggregations. The product at this level is a refined class diagram.

Analyse Changes of State in Objects Further refinement of the model by showing changes of state, the product is state diagram.

Define the Interaction among Objects Object interaction is dealt with at this level. The product is a set of sequence and collaboration diagrams which depicts interaction.

Analysis Integration with Cooperating Systems Uncovering of specific details of the integration with the cooperating systems is the concern at this stage, the product is detailed deployment diagram and data models.
Design

At this stage, the results of the analysis stage are used to design the solution.

**Develop and Refine Object Diagrams** Using class diagrams each operation is examined and corresponding activity diagrams is developed, the product are the object and activity diagrams.

**Develop Component Diagrams** To visualise the components that will result from the next stage and show the dependencies among them, the product is component diagram.

**Plan for Deployment** The concern at this level is planning for deployment and for integration with cooperating systems; deployment diagram that shows where component will reside is created. The product is a diagram that is part of the deployment diagram developed earlier.

**Design and Prototype User Interface** This starts when all use case are completed, the work products are screen snapshots of the screen prototypes.

**Design Tests** Use case diagrams used to develop test scripts for automated test tools, the product is the test script.

Development

Activity diagrams serve as the basis for much of the coding at this stage, also products from analysis and design stages.

**Construct Code** With class diagrams, object diagrams, activity diagrams, and component diagrams the code for the system is constructed; the product is a code.

**Test Code** Test script is run to assess the code; the product is the test results.

**Constructs User Interfaces, Connect the Code, and Test** Prototype user interfaces are constructed and connected to the code, the product is the functioning system complete with user interfaces.

Deployment

The system is deployed on the appropriate hardware and integrated with the cooperating systems.

**Plan for backup and recovery** Plan for steps to follow in case the system crashes.

**Install the finished system on appropriate hardware** The system is deployed on the appropriate computer, the product here is the fully deployed system.
4.3 Unified Process

Structure

Requirements
The effort is to develop a system model, use cases are employed to create the model of the system to be built.

Actors and Use Cases Identifying actors and use cases is the aim of this stage, this is most crucial activity for getting the requirements right.

Prioritise Use Cases The determination of use cases to be developed in early iterations and which can be developed in later iterations.

Detail a Use Case Describing the flow of events of each use case in detail; how it starts, ends, and interacts with actors.

Prototype User Interface Building a prototype of the user interface by designing.

Structure the Use Case Model The use-case model is structured to extract descriptions of functionality.

Analysis
The requirements are analysed by refining and structuring them.

Architecture Analysis This outline the analysis model and the architecture by identifying analysis packages, obvious analysis classes, and common special requirements.

Use Case Analyse This is done in order to; identify analysis classes whose objects are needed for use case’s flow of events and to capture special requirements on the realisation of use case.

Analyse a Class The purpose of analyse a class are to; Identify responsibilities of an analysis class; Identify attributes of an analysis class; Capture special requirements on the realisation of the analysis class.

Analyse a Package A package is analysed in order to; ensure that the analysis package is independent of other packages, and that it fulfils its purpose of realising domain classes or use cases. To describe dependencies so that future changes can be estimated.
Chapter 4. Methodologies for System Development

**Design**

The system is shaped and its form (including its architecture) are realised that lives up to all requirements made on it.

**Architecture Design** This outline the design and development models and their architecture by identifying the following:

- Nodes and network configurations.
- Subsystem and interfaces.
- Architecturally significant design classes.
- Generic design mechanisms that handle common requirements.

**Design a Use Case** The purpose of design a use-case is to:

- Identify the design classes and/or subsystems.
- Distribute the behaviour of the use case.
- Define requirements on the operations of design class and/or subsystems.
- Capture implementation requirements for the use case.

**Design a Class** The purpose of design a class is to create a design class that fulfils its role in use-case realisation.

**Design a Subsystem** The purposes of this stage are to:

- Ensure that the subsystem is independent of other subsystems and it provides the right interfaces.
- Ensure that the subsystem offers a correct realisation of the operations as defined by the interfaces it provides.

**Implementation**

**Architectural Implementation** The purpose is to outline the implementation model and its architecture by: identifying architecturally significantly components; Mapping components to nodes in the network configuration.

**Integrate System** In this stage the following are done: creation of an integration build plan Integration each build before it is subject to integration tests.

**Implement a Subsystem** This is to ensure that a subsystem fulfils its role in each build, as stated in integration build plan.

**Implement a Class** The purposes of implementing a class is to implement a design class in a file component. This also includes handling maintenance aspects of the implemented class, such as fixing defects when the class has been tested.

**Perform Unit Test** The purpose is to test the implemented components as individual units.
4.4 Conclusion

Test

Test Plan Test The purpose of test planning is to plan the testing efforts in iteration by:

- Describing a testing strategy.
- Estimating the requirements for the testing efforts.
- Scheduling the testing effort.

Design Test The purpose of designing tests are to:

- Identify and describe test cases for each build.
- Identify and structure test procedures specifying how to perform the test cases.

Implement Test The purpose of implementing tests is to automate test procedures by creating test components.

Perform Integration Test The integration tests required for each build are performed.

Perform System Test The purpose of system testing is to perform the system tests required in each iteration.

Evaluation Test The purposes of test evaluation is to evaluate the testing efforts within iteration.

4.4 Conclusion

The integration of the object-oriented design, with the development process to be used in the flood hazard management system will be covered in four phases; requirements, analysis, design and development.

The Unified Methods will be used as vehicle to carry out the development processes, the Unified Process and GRAPPLE methodologies in particular will be implemented.

Both Unified process and GRAPPLE uses UML as tools throughout the development processes.

The next chapter will cover the first phase of the flood hazard management system development, namely requirement phase. The requirement phase will describe how to capture the requirements in the user and system categories on the conceptual basis.
Chapter 5

Requirement Phase

5.1 Introduction

The purpose of the requirement phase is to direct the development process towards the realisation of flood hazard management system. This is realised by describing the flood hazard management system requirements well enough so that an agreement can be reached between user and system requirements.

This chapter will describe how to capture the requirements on a flood hazard management system as a domain model to set the contents of the system based on chapter 2 (especially section 2.2 and section 2.3.1).

In South Pacific Applied Geoscience Commission (SOPAC) development of disaster management information infrastructure in Pacific Cities Project. The requirements for the project was first based on external scientific perspective and lateron it was integrated with local community knowledge, experience and expectations; Ken Granger [15] termed it outside-looking-in approach.

For Pendik district in Istanbul [48], the information for the system requirements were obtained from conducted interviews and filled-in questionnaires for specific case.

The actions at this phase are about concepts and not the flood hazard management system to be built that comes in design phase.

At this stage the products will be actors, use cases, use-case model, and activity diagram with their details. The actors, use cases, and use-case model are realised from the Unified Process [21], while the activity diagram is obtained from GRAPPLE method [44].

The results from requirement phase is the base of conceptualisation of the flood hazard system using the object-orientation paradigm, which is independent of any programming process that may follow after modelling. This is advantageous than having a direct approach to physical modelling, say, using visual basic (VB) to solve the user needs.

The results from requirement phase is the backbone of the whole system, since most of the design and analysis will be based on this phase of development.
5.2 Finding the Actors and Use Cases

Finding the Actors

The actors in flood hazard scenario are in user and system categories of flood hazard management system.

- Actors in user category are proposed to be development and investment users and community users [36]:

  **Development and investment users**: Development assistance agencies, insurers and private consulting firms.

  **Community users**: Building, engineering, planning, and safety departments; Disaster preparedness agencies; Governing bodies; Offices of emergency services; Police and fire departments; Provincial and district councils; Public works and highway departments.

  Development and investment users are those involved with development projects in local area, this are private owned companies or firms; while community users are those working in the development projects and rendering services to the community, they are under the local authority.

- Actors in system category are:

  **GIS package** A system capable of combining layers from spatial database to create maps and data sources. This package can support landuse planning, risk and vulnerability assessment, disaster forecasting and hazard management [1].

  For flood hazard management system the package will have simple interface, strong analytical and graphical capabilities [36].

  **Application software** This are used in manipulation other than those covered by GIS package, examples are; Windows NT 4.0, Solaris 2.6 as system software for client operating system.

  **External hardware** These are hardware which are outside the flood hazard management system, say, printers, plotters, digitisers.

  In order to avoid the overlapping roles of the actors, generalisation is applied to the listed actors. Generalisation simplify the work, with an understanding of the use-case model when putting together complete use-cases requested by the actor or system (Jacobson *et al* [21]).

Actors Generalisation

Actor generalisation were derived by organising actors in clusters, the actors within a cluster had more or less the same functions (e.g. civil engineer and highway engineer, police officer and officer from fire department). These clusters were generalised into packages of actors.

The following is an example of the output of actors generalisation for the flood hazard management system:
Chapter 5. Requirement Phase

Emergency Officer A person from either police and fire departments or offices of emergency services. They are responsible for flood hazard preparedness activities and rescue missions.

External Package This comprises of an application software and external hardware which are outside the flood hazard management system.

Other actors in the flood hazard management system are; Disaster Analyst, Facilitator, GIS Package and Planner. Details for these actors generalisation are in Appendix A.

Use Cases

The actors are used to suggest candidates for use cases for each actor, this is done by studying actors role in the flood hazard management system. Furthermore use cases are realised from literature review of the related work (section 2.2 on page 6).

The actors need use cases to support its work to create, change or analyse a flood hazard object. Some of the candidates may not become use cases by themselves; instead, they will be parts of other use cases (e.g. Clipping in Update use case).

Examples of use cases are:

Update This use case deals with adding or editing data/information on a layer so as to update it.

The actors on this use case are DisasterAnalyst, Planner and EmergencyOfficer.

Print The user are DisasterAnalyst, Planner or EmergencyOfficer when they need a hardcopy of specific layer or information.

The use case will require printing specifications from the initiator and selection of hardware on which the hardcopy is to be printed.

Other use cases in flood hazard management system are; StartUp, Termination, Maintenance, Save, Analyse, AnalyseTable, Evaluate, Open, Select, and Display. Their detailed description are elaborated in Appendix A.

5.3 Describing the Use-Case Model as a Whole

The diagram and description are prepared to explain the use-case model as a whole, especially how the use cases relate to each other and to the actors. A glossary is compiled to be used to define important and common terms used by stake holders/actors when they describe/use the flood hazard management system, also it is useful in reducing the risk of misunderstandings in general [21].

For this research the use-case model will be organised in clusters of use cases known as use-case packages [44], and for consistency, a glossary of terms
5.3 Describing the Use-Case Model as a Whole

will be developed [21, 44]. In UML, the description is a tagged value of the model, tagged values provide a way of defining new properties of existing elements.

An example of use-case model for specific instance is Layer Updating, which deals with adding or editing data/information on a layer so as to update it. This use case is described below (Figure 5.1):

![Layer Updating Use Case Diagram](image)

**Layer updating using Update use case (Figure 5.1)** The Planner, who is an actor instance, uses the Open use case to open the specific affected layer from the flood hazard management system.

The Planner (an actor) activates the Update use case for clipping the affected layer with floodplane layer (FP) and lifeline network layer (LN). By this process, the planner will be able to have; the extent of the flood on a particular area, where will flood be the greatest and which rescue route to be used by emergency crew.

The Planner then uses the Print use case to print the new layer for flood hazard emergency activities for the particular area also for further reference and field work.

This scene is a starting block in a system which will be able to help the Planner in flood hazard scenario. Instead of using GIS system, which leave much on the user side, this particular system will be able to assist a Planner (and other stakeholders) in such a way that, the specific flood hazard management needs will be realised at finger tips.

This approach will be also useful in realising specific cases for the flood hazard management scenarios. Specific cases can be dealt with by manipulation or addition of use cases during iterations, this can not be achieved if, say, using VB or any tools to physically realise a system directly without conceptualisation. For this research work, object-oriented paradigm is used to realise the
flood hazard management system in systematic approach so as to be able to carry out iterations for the betterment of the system.

### 5.4 Activity Diagram

The scenario from section 5.3 on page 27 is represented in Activity Diagram (Figure 5.2).

![Activity Diagram](image)

**Figure 5.2:** The UML activity diagram for layer updating action in the flood hazard management system.

**Updating by Clipping** The Planner use the StartUp use case to start the flood hazard management system, he then use Open to access the layers and GIS Package.
The actor display floodplane (FP) and lifeline network (LN) layers on which he uses clipping operation from Analyse use case to form an updated layer of floodplane which incorporates lifeline network layer.

The next action is saving the new layer and printing the hardcopy, the printer (from External Package) is used for printing.

This diagram shows the flow from activity to activity, it addresses the dynamic view of the flood hazard management system during flood hazard occurrence.

The detailed activity diagram for the generic scenario, which was used to acquire the requirements for the flood hazard management system is shown on Appendix A.

## 5.5 Conclusion

Since the flood hazard scenarios do have a common base, it is possible to model the user and system requirements for flood hazard management system without having the need for a specific case to rely upon. This can be supported by SOPAC approach in developing disaster management information infrastructure in Pacific Cities Project. The requirements for the project was first based on external scientific perspective and later it was integrated with local community knowledge, experience and expectations.

This generic approach in exploring user requirements for flood hazard scenario were realised from literature review, later on it will be specified for a particular case. This observation does not rule out the other possibilities of gathering the user/system requirements from conducting interviews and filling-in questionnaires for specific case; as it was done in Pendik district in Istanbul.

The advantage of starting with generic case is that, the specific cases can be modelled during iterations as the flood hazard management system is designed for particular case having the same generic platform. This will be realised by manipulation or addition of use cases during iterations, this is contrary in using, say, VB or any tool in direct realisation of a system without conceptualisation.

The output of this requirement phase chapter is a starting point for the subsequent chapters containing the analysis, design and development phases. The use cases will drive the workflows in the coming chapters by interactions. The use cases will be the seamless links between different workflows by using UML models in the object–oriented paradigm.

In flood hazard spatial database environment, analysis is the action of understanding and describing the user and system requirements, furthermore in the analysis phase (next chapter), the inputs will be UML requirements models obtained from this chapter.
Chapter 6

Analysis Phase

6.1 Introduction

In this chapter, the use cases will be formulated as objects. This is done in order to gain better understanding of the requirements for the flood hazard management system development workflow.

Most of the ideas used in this chapter comes from Jacobson et al [21] and Schmuller [44].

The requirements are analysed as described in the requirements phase by refining and structuring them. This is done so as to realise precise understanding of the requirements that are to be maintained in order to give the structure of the whole flood hazard management system.

The analysis gives the internal view of the flood hazard management system and it is used to understand how the system should be designed and implemented.

Analysis models were initiated by identifying major analysis packages, entity classes, and generic requirements. Then, each specific use case was realised by starting the behavioural requirements of each classes.

During analysis, continually new analysis packages, classes, and common (generic) requirements are found as the analysis model evolves. At the same time analysis packages were continuously refined and maintained.

This shows that, in the development process it is not easy to stick to top-down or bottom-up approach throughout. Alternatively, it is very common to move along two approaches as process goes on.

In order to give out better understanding of the analysis phase of system development for flood hazard management, much consideration is given to the updating process as a representation of the generic activities within the flood hazard management system.

6.2 Analysis Model

The analysis model is comprised of an analysis system, denoting the top-level package of the model. The model is organised into manageable pieces by
6.2. Analysis Model

analysis packages [21].

Figure 6.1: The analysis model and its packages containing analysis classes and use-cases realisations (Jacobson et al [21]).

The structure of analysis model is illustrated in Figure 6.1, the components of analysis model are:

**Analysis Class** represents an abstraction of classes and/or subsystems in the system’s design. It handles functional requirements (and not nonfunctional requirements which are dealt with in design and implementations activities).

Figure 6.2: Standard class stereotypes (with examples) as used in flood hazard management system analysis.

Analysis classes have one of three basic stereotypes (Figure 6.2):

- Boundary; which models interactions between the system and its actors (e.g. user interfaces, terminals).
- Control; they control the dynamics of the system, say coordination, transactions, sequencing and control of other objects in the system.
- Entity; this model information objects that are persistent and long lived in the flood hazard management system (e.g. layers, thematic data).
These stereotypes are used in finding and describing analysis classes and contributes the creation of an object model and architectures [32]; they are represented in UML diagrams (see Appendix B).

**Use-Case Realisation–Analysis** provides a trace to a specific use case in the use-case model by describing how a specific use is realised and performed in terms of analysis classes and their analysis objects.

A use-case realisation is composed of textural flow-of-events description, class diagrams (depicting participating analysis classes), interaction diagrams (depicting the realisation of a particular scenario of use case in terms of analysis object iterations). This process, as an example, is realised for *Maintenance* use case in section 6.4 on page 36.

**Analysis Package** this package provide means of organising the analysis model in manageable pieces, the package is formed by analysis classes, use-case realisations and other recursively analysis packages. Recursively in the sense that the development process in this research is carried out in iterative way.

Analysis packages contents should be strongly related (cohesive) and their dependencies on each other should be minimised (loosely coupled) [21]. This situation makes the package easier to maintain and manipulates, since changing some classes within a package will effect classes within the package itself and not outside.

The analysis packages should have the following characteristics:

- Representing a separation of analysis concerns.
- Should be created on functional requirements (e.g. updating, saving) and on problem/application domain basis.

**Architecture Description** is a description of the flood hazard management system’s architecture that includes the architectural views of the analysis models.

The following are included in description:

- The decomposition of the analysis model into analysis packages and their dependencies.
- Key analysis classes, such as entity classes for an important phenomena; boundary classes that encapsulate vital communication interfaces; control classes that represent important transaction; and analysis classes that are general and have many relationships with other analysis classes. Usually abstract classes are significant in this description than subclasses.
- Use-case realisations that realise important and critical functionality.
6.3 Architectural Analysis

The main aim of architectural analysis is to realise the analysis model and the architecture by identifying analysis packages, analysis classes, and special requirements for the flood hazard management system. Update and Analyse use cases specifically are detailed in this section so as to give an overview of its implementation in the flood hazard management system.

Identifying Analysis Packages

The grouping of use cases to packages was based on the use cases needed to support specific process. For this case Disaster Management and Planning packages are realised (see Figure 6.3).

Within these packages there is a commonality, such as using GisPackage use case, this is found in shared class Analyse, Save and Update. These use cases are taken outside the packages, this makes the packages to be dependent on this new general classes.

The sharing is also possible through sharing of analysis packages. An example is shown on Figure 6.4, in which analysis packages Disaster Management and Planning shares the analysis classes analysis/update rules handlers and layers. Dependencies for these packages is shown on their analysis classes relationship to each other.

Service packages are shown on Figure 6.5, these are those which comprises of analysis classes contributing to the same service.

In order to distinguish between specific and general functionality, it is ideal to have application-specific packages in top layer and general packages in a lower layer (Figure 6.6). This makes the distinction between specific and general functionality clear.
Identifying Special Requirements

Special requirements are requirements that occur during analysis, they are helpful in supporting design and implementation activities. Among the requirements are restrictions on:-

- Persistence.
- Security features.
- Distribution and concurrency.

These requirements are referred for individual use-case realisations and analysis classes.

6.4 Use-case Analysis

The following are dealt with in this analysis:
6.4. Use-case Analysis

- Identifying analysis classes,
- describing analysis object interaction,
- and capturing special requirements.

**Identifying Analysis Classes**

The control, entity, and boundary classes are identified for each use case realisation, also the outline of their names, responsibilities, attributes, and relationships.

Figure 6.7 shows realisation of analysis classes for the flood hazard management system maintenance process.
Chapter 6. Analysis Phase

**Maintenance Class Diagram (Figure 6.7)** A boundary class, flood hazard management system *Check UI*, allows the facilitator to interact with the maintenance entity class. The Facilitator checks the maintenance schedule list and then asks the system to carry out the maintenance.

The control class, *Maintenance Scheduler*, is responsible for coordination between flood hazard management system *Check UI* and entity class, *Maintenance*. *Maintenance Scheduler* accepts maintenance request and the date/time to carry the maintenance. It invokes the maintenance process then it changes the status of the particular maintenance (in the maintenance list) from “scheduled” to “performed” with date specifications.

The class diagrams for realisation of Update use case is shown on Figure 6.8, this diagram shows the participating classes and their relationships within use-case realisation.

![Figure 6.8: A class diagram for realisation of Update use case.](image)

**Update Class Diagram (Figure 6.8)** A boundary class, *Update UI*, allows the Planner (actor) to interact with the layers in *Layer* entity class. The Planner browsers and select layers to be clipped, and then asks the flood hazard management system to carry out the updating.

The control class, *Update Rules Handler*, is responsible for coordination between *Update UI* and entity class, *Update*. *Update Rules Handler* accepts update request and asks the confirmation to carry the updating from entity class *Updating Confirmation*.

If the permission is obtained, it invokes the updating process by clipping. After that it changes the status of the layer to “updated” with time/date specification.

The class diagrams for realisation of other use cases are in Appendix A. The update process is covered again in collaboration diagram (in next section) so as to show the sequence of actions which are not depicted in class diagrams.
6.4. Use-case Analysis

Describing Analysis Object Interaction

The sequence of actions for Update use-case realisation are depicted by collaboration diagram (Figure 6.9), the interaction is represented by links between the objects. The links have message attachment, the name of the message denotes the intend of the invoking object when interacting with invoked object [21].

A glossary is used to define important and common terms (section 5.3 on page 27). For example in the following Update use-case realisation for flood hazard scenario, a user interface is denoted by Update UI. This term has to be entered in the glossary with explanation so as to avoid ambiguity in interpretation as “Process of updating the user interface” or “A user interface for updating process”.

![Collaboration Diagram](image)

**Figure 6.9: A collaboration diagram for a realisation of the Update use case.**

**Updating process (Figure 6.9)** The Planner browses the layers to be updated (1, 2) through Update UI. The Update UI uses Update Rules Handler to check the layers against their update rules set by the authority through Facilitator (3, 4, 5), then it shows the list of layers to the planner.
Any read-only layer is flagged for the planner by Update UI with comments that the layer is read-only and no editing is allowed.

The Planner select layers through Update UI and control the clipping (6). The Update UI asks the Clip Control to schedule the clipping of the layers (7). The Clip Control then creates an updated layer (8). The Update UI then changes the status of the layer to Updated (9).

6.5 Class Analysis

The purposes of analysing a class are:

- Identifying and maintaining the responsibilities of an analysis class.

- Identifying and maintaining the attributes and relationships of the analysis class.

- Capturing special requirements of the analysis class.

The class analysis involves the following activities; identifying the responsibilities, attributes, associations/aggregations, generalisation and capturing of special requirements.

Identifying Responsibilities

The class responsibilities are realised by combining the roles that the class plays in use-case realisations, these are obtained from studying the class and interaction diagrams.

The following are class responsibilities for:

(i). Clip Control in Update scenario:

- Create updated layer from clip operation.

- Send a note when the update is effected or aborted.

- Control the layer to be clipped and the clipper as per Planner selection.

(ii). Analysis Scheduler in Analyse scenario:

- Create an analysed layer.

- Send a notification when analysis has been effected or aborted.

- Initiate the selected package for the analysis process.
6.5. Class Analysis

Identifying Associations and Aggregations

The links used in collaboration diagram are studied and the needed associations are determined. The aggregations among objects are realised together with definitions of association multiplicities and role names.

The instances of interactions between analysis objects in Analysis and Update scenarios are shown in Figure 6.10.

In Analysis; a layer is analysed by the Analyse class, the role name is analyse a layer and it is an association with (1..* ) multiplicity.

In Update; a layer is clipped by another layer by Clip Control, the association is formed by (2..* ) multiplicity and the role name is clip a layer.

Identifying Generalisation

This is used to extract shared and common behaviour among different analysis classes. The aim of this stage is to make the analysis model easier to understand.

In Analysis and Update scenarios, the Conformation class generalises Analyse Confirmation and Updating Confirmation classes (Figure 6.11).

Capturing Special Requirements

The nonfunctional requirements identified on a use-case realisation (refer to section 6.3 on page 35) are captured, these are to be handled in the design and implementation phases.
Chapter 6. Analysis Phase

Figure 6.11: The Confirmation class generalises Analyse and Update Confirmations classes.

An example of special requirements for Update use case, specifically for layer clipping, are as follows:

**Update Rules Handler** class must specify the Actors who allowed to update the layers; security features should be covered.

**Layer** class for clipping update instance must be able to handle two layers at a time, one layer to be a clipper and the other one to be clipped.

6.6 Package Analysis

This deals with defining and maintaining the dependencies of the package or other packages whose classes are associated with it.

The purpose of analysing a package is to ensure the analysis package fulfills its purpose of realising domain classes/use cases; also to describe dependencies so that future changes can be accommodated and estimated.

6.7 Discussion

When components depend on their own interfaces, instead of generic interface, the remodelling is a must when a replacement is desired [2]. The same situation is true for packages and use cases.
For example, Update use case for flood hazard scenario of Enschede Municipality (Figure 6.8) is to be replaced by Update use case for earthquake hazard for Pendik District of Istanbul [48]; the layer analysis class about Enschede is to be replaced by 3D model of geological structure and man-made features of Pendik. Since the interface was developed specifically for the flood hazard management for Enschede, the swapping of Enschede 2D layers with Pendik 3D model will require remodelling of the user interface.

There is a possibility for a package to be substituted by a human being, say, an Update package can be replaced by a human operator. In Cape Town mining database project [46], an aerial photo interpretation was carried out by an operator and the technique improved the accuracy of previously captured minerals resource data.

This situations can be contained by utilising UML advantages by modelling the generic user interfaces instead of specific ones.

The same case may apply for other analysis cases within the same domain, by grounding the analysis on generalisation platform. For example, in generalisation of analyse and updating confirmation in flood hazard scenario Figure 6.11 (section 6.5 on page 40).

In order to maintain these packages effectively, it is important to reduce the number of relationships between classes in different packages so as to reduce packages dependencies (section 6.2 on page 31).

With advancement in IT technology, there is plenty of data and information about natural hazards on the Web (world wide web) from experts in various fields, the only needed is the facility to obtain those data and to filter them in proper way. Filtering may involve temporal aspect, accuracy and other needed specific parameters.

The collaboration of UML diagrams and object-orientation modelling is exhibited in step-wise approach of exhausting the details of classes. This was achieved on section 6.4 with update realisation with class diagram (page 36) and collaboration diagram (page 38) on Figures 6.8 and 6.9 respectively.

6.8 Conclusion

The product of the analysis phase is an analysis model with its components; architecture description, use-case realisation–analysis, analysis class and analysis package.

When the structure of the analysis model is preserved during design stage, a flood hazard management system will be realised that is maintainable. It will be flexible to changes in requirements, and it will include components that can be reused when the flood hazard management system is built. Effective components will be essential during maintenance and upgrading. There is no need to upgrade the entire flood hazard management system; instead, only components get replaced or added as the need arise.

The analysis models in UML obtained from this chapter represent how reality is organised and what users need to be implemented in flood hazard spatial database. These models will be inputs in design phase, in the next chapter.
Chapter 7

Design Phase

7.1 Introduction

In this phase, the flood hazard management system is shaped so as to preserve a structure as per analysis model. All requirements (functional, nonfunctional and other constraints) are considered in order to produce a blueprint of the implementation.

The purposes of design for flood hazard scenario are realisation of:

**Architectural design** The nodes are outlined in the deployment model, major subsystems and their interfaces, design classes and in design model.

**Use Case design** Each use case is realised in terms of design classes and/or subsystems and their interfaces.

**Class design** The requirements for each class or subsystem in use-case realisations are specified and integrated into each class by creation of consistent operations, attributes, and relationships on each class.

The following sections explain how the above are realised in the flood hazard management system. The Update use case is used as a representation of other use cases as it covers a generic scenario on flood hazard management system.

7.2 Architectural Design

The aim of architectural design is to outline the design and deployment models and their architecture [21].

In this research various reuse possibilities were considered, such as reusing parts of similar systems. Example of the components which can be used by the flood hazard system are those found in Urban Information Systems [20, 50]; these components comprise of map images, tabular data, analytic methods, different actors, spatial data, computer software and hardware, transportation maps, and shortest path for the fire fighters. The obtained subsystems and interfaces can be embedded into the design model to realise a generic design.

UML deployment diagrams are used to define and to depict the component to be deployed in the flood hazard management system. This components are
the hardware, the software on those hardware, and the middleware used for connecting the components of the flood hazard management system [2].

Simple terms were used in the UML deployment diagrams; such as layers, planner and updating. These terms have the advantage of being recognised easily by the stakeholders in the flood hazard management system being developed. Only those relevant components are shown, as these are important in understanding the flood hazard management system.

An actor instance, planner, is used in the description so as to gain an understanding of specific roles among actors. Also to utilise one of the many advantages of using UML in the object-orientation paradigm, which is shifting between instances and generalisations for various reasons and yet to maintain the consistency.

The following subsections give the detailed description of the nodes, network configurations, and subsystems which make up the flood hazard management system.

7.2.1 Nodes and Network Configurations

For the flood hazard management system a network configuration of three-tier pattern will be used [45]; the tiers are for user interaction, database functionality and for application logic. This configuration has the advantage of separating the main functions, this will make it easier in managing the components. Say, in case of system crash it will be faster to process recovery. The disadvantage will be on running costs, but these are outweighed by the advantages. The other common configuration is a two-tiers pattern, in which two tiers are grouped together, say, application logic is relocated to database tier and user interaction be on its own. The advantages of two tier are the disadvantages of three-tier configuration and vice versa [18].

Since the focus of the research is a generic one, the minute details on which software components are deployed to which hardware nodes were not dealt with. This details may be realised in usage of CASE tool for specific scenario [2].

Figure 7.1 shows a UML deployment diagram for an Update use case in flood hazard management system.

The figure depicts how the major software components, that comprise a single application are deployed into the flood hazard management system. These components are essential in the whole process of realising the spatial database system for flood hazard.

The advantages of this configuration usage in the flood hazards scenario are:

- Application server may be placed on any system within the spatial database system network.
- Application servers may be active simultaneously on the same or multiple workstations.
- Depending on the data passed with the request, the request may be routed to a particular server.
Chapter 7. Design Phase

Figure 7.1: UML Deployment diagram for the Updating use case in flood hazard scenario.

- It supports **graphical user interface** (GUI).

The nodes will be communicating using the **Transmission Control Protocol and Internet Protocol (TCP/IP)**.

TCP/IP, has extensively been used in various institutions all over the world and it has proved to be reliable, for example in the emergency management for the Winston-Salem city in the United States [33].

TCP/IP does not belong to a specific vendor; it is an open protocol also the foundation on which internet is based [5], it is recommended by **Open GIS Consortium (OGC)** in that application [40]. **The Wide Area network (WAN)** and **The Local Area Network (LAN)** will be used for connections.

The selection of these protocols and connections for the flood hazard management system does not entails that these are the only alternative hardware for the same, other types may be used and still the flood hazard management system will function effectively.

Figure 7.2 illustrates how client/server technology will be used for flood hazard scenario.

The ease and speed with which data transfer can be carried out over the **Web**, and the possibility of **data mining**, are among the driving force in using the Web. Data mining is the process of searching and retrieving the data from the Web effectively. Actors interactions, including online data provision, gives new venues of interoperability that were not previously available or if available were very inefficient [25].

As the need of temporal data and real-time updating of data in flood hazard scenario is crucial, a connection to the Web service can be accommodated to the flood hazard management system. With this optional, **data mining** can even be extended to **object mining** [4], also charts, tables and ready reports can be obtained from the Web [24] in [48].

The **Federal Emergency Management Agency (FEMA)** has sponsored an OGC Pilot Project known as **Multi Hazard Mapping Initiative (MMI-1)** to enable the
sharing of multi-hazard mapping data between federal, state, and local governments. Through the MMI-1, emergency managers are able to locate, retrieve and exploit multi-hazard mapping data from any of the participating agencies [40]. This pilot project, which became operational (on OGC Network) in April 2002, shows the importance of the Web connection for the flood hazard spatial database system. Furthermore, the possibilities of acquiring or sharing data/information from different sources for the different applications within the flood hazard management system domain can be realised in effective way.

The detailed network configuration is not dealt in this research, it involves among other things; nodes processing power and memory size, characteristics of the connections and communication protocols (e.g. bandwidth, quality, availability). For this research work all these are encapsulated in Facilitator duties in the deployment diagrams.

### 7.2.2 Subsystems and their Interfaces

In this research there is no consideration of in-house subsystem development, the concentration is on reused products. Therefore most of the evaluation is on
the reuse opportunities in design model.

Consider the above Figure 7.3 and the following descriptions for the flood hazard management system:

**Application subsystem:** The packages found during *analysis package decomposition* (chapter 6, on 6.2 page 31) are used to identify corresponding subsystems within the design model.

**Middleware and system-software subsystem:** All flood hazard management system functionalities are based on middleware and system-software, such functionalities are operating system, database management systems, communication software and GUI design kit [21]. The subsystems in the middleware and system-software layers encapsulate software products. Since the systems will have different types of workstations at various departments, there is a need to interoperate across different platforms. The flood hazard management system will implement interoperability using Java's middleware of Abstract Windowing Toolkit (AWT), Applet and Remote Message Invocation (RMI) packages. In order for a workstation to execute Java code, a Java interpreter (Java Virtual Machine) is required. A web browser is for loading Web pages that carry applets. At the lower layer, system software such as the TCP/IP protocol for internet communication will be implemented.

For flood hazard scenario, choosing and integrating software products is of primary concern, the chosen product should fit the architecture and it should provide a cost-effective implementation of the flood hazard management system. When middleware and system software are acquired, there is no control over their evolution, thus it is important to avoid total dependency on specific
product [21]. This implies that the flood hazard management system can use any off-the-shelf component on the market, and when a new and effective component from any vendor is introduced in the market, it can be accommodated to replace the aging (or outdated one) without much effort in redesigning the flood hazard management system.

Nowadays many applications are built on purchased frameworks or by gluing together existing applications, an example of Microsoft’s OLE/COM which provides a way of joining together an entire applications [11].

In order to maintain the freedom of choice between different products, and to minimise the cost of updating the flood hazard management system; it is essential to treat software product as separate subsystem with explicit interfaces to a flood hazard management system.

Open standards middleware, of Open Database Connectivity (ODBC), OLE-DB, and Java Database Connectivity (JDBC) were analysed as candidates for application programming interface (API) for the flood hazard management system. All of them allow simultaneous secure, real time, read/write access to the database [22].

JDBC is a Java API for database access, it will be used as a middleware between application and database servers. This does not rule out other middleware, it is dealt with here just to show how the dependency on specific product can be avoided by proper analysis.

The JDBC standard defines four types of drivers;

**Type 1** is a JDBC-ODBC bridge that provides JDBC connectivity via ODBC drivers.

**Type 2** consists of an interface, written partly in Java, between Java programs and the vendor-specific database access middleware (for instance, Oracle SQL*Net or Sybase Open Client).

**Type 3** is a pure Java driver that, at the client, translates JDBC calls into a database-independent network protocol.

**Type 4** is a pure Java driver that converts JDBC requests into a database vendor’s particular network protocol.

Among these JDBC types, the suitable one for the flood hazard system is type 3 driver. The advantages of type 3 drivers are [31]:

- The ability to connect Java-based clients to whatever types of databases a separate server-side driver has been developed for.
- They require basic network connectivity at the client (a TCP/IP protocol stack), but they do not rely on the presence of vendor-specific middleware or ODBC.
- They are highly generic and can run on any Java-enabled platform with a TCP/IP connection to a database server.
- They are well-suited for use over the Internet.
Chapter 7. Design Phase

The backbone of flood hazard management system is the data and information in the database. These data can be conveyed through nodes as explained in this subsection, all this sums up to a core step of accessing the database itself. JDBC is one of the effective middleware to access the databases in open standard platform; the accessed database should also be reliable. For a database to be effective, a reliable database management system (DBMS) is essential. The following subsection covers the spatial DBMS for the flood hazard management system.

Spatial DBMS

The ideal database management system (DBMS) for flood hazard management system should allow the following performance on the data:

- Input and store.
- Retrieve and analyse.
- Display and select.

It is possible to store spatial data in files, and functions to define this data; But this does not allow data independence principle, and leads to problems in data security and concurrency control.

To overcome this situation the following alternatives exists [41];

Loosely Coupled Approach Many GIS separates descriptive data management from spatial data management, ArcInfo (ESRI) and TiGRis (Intergraph) are following this approach. In such architecture, two systems coexists (Figure 7.4):

- A relational DBMS or some components of its descriptive data.
- A specific module for spatial data management.

the shortcomings of this approach are:

- The coexistence of heterogenous data modules, which implies difficulties in modelling, use, and integration.
- A partial loss of basic DBMS functionality, such as recovery techniques and querying.

Integrated Approach To overcome some of the above problems integrated approach based on DBMS extensibility has been considered. The basic concept is the ability to add new types of operations to a relational system.

In the case of geospatial applications, it is done by extending relational DBMS as follows:

- The query language SQL is extended to manipulate spatial data as well as descriptive data.
- Other DBMS functions, such as query optimisation, are adapted in order to handle spatial data efficiently.
Example of these systems are Oracle8i Spatial and Postgres.

From the above details, the requirements for spatial DBMS for the flood hazard management system should:

- Integrate the representation and manipulation of geometric information with traditional data at the logical level.
- Provide an efficient support at the physical level to store and process this information.

Many object-oriented developers concentrates more on class diagrams, in this research use case diagrams are of high priority in order to achieve an effective flood hazard management system. The next section deals with the use case in design stage so as to realise the physical models of the flood hazard management system components.

### 7.3 Use Case design

The purpose of designing a use case are:

1. Finding instances of design classes/subsystems needed to perform the use case's flow of events.

2. Defining requirements on the operations of design classes and subsystems, and their interfaces.

3. Capturing implementation requirements for the use case.
7.3.1 Identifying the Design Classes

Design classes needed to realise the use case, are obtained from:

- Studying analysis classes that participate in use-case realisation analysis.
- Studying special requirements of the corresponding use-case realisation-analysis and then identifying the design classes that realised those special requirements.

Figure 7.5 shows the design classes in the design model tracing to analysis classes in the analysis model for the flood hazard management Update use case, only few instances are shown for clarity.

When analysis classes (section 6.2 on page 31) are designed they give refined design classes that are adapted to the implementation environment. For example, UpdateUI analysis class is designed by the following refined design classes; ListBox, Display, and BrowserButton, this represents how conceptual model
is transferred to *physical* model in object-orientation paradigm with usage of UML.

The class diagrams that shows the classes that participate in the realisation of use cases and their associates are the products at this stage.

The UML class diagrams can be used for various purposes, say, understanding user requirements or describing design. In each circumstances different styles has to be applied [2], in this research work they are used to describe the detailed design.

Figure 7.6 shows the Update use case realisation classes, active classes (e.g. Update Processing, Update UI) keep the flood hazard system running. They do this by passing layers from the spatial database to an actor instance (a Planner) for the updating of the layers using clip process.

It is useful in UML to utilise the available flexibility of using appropriate instance at particular time for clarity. This may be observed in this research by shifting between instances and generalisations as the needs arise, say, *Actor* generalisation and its instance *Planner*; *Update* generalisation and its instance *Clip*.

These classes in design phase do interact with each other to realise their use cases, the following subsection covers the design object interaction for flood hazard management scenario.

### 7.3.2 Description of Design Object Interaction

The description on how the design objects interact, is done by using sequence diagrams. The sequence diagrams contains the participating actor instances, design objects, and message (which triggers processes) transmissions between them.

The advantages of UML sequence diagrams are:

- Validating the logic of the spatial database for flood hazard scenario, by describing a way that the flood hazard management system is used. The logic of an Update use case scenario, described by the course of action in which a layer is selected then updated by the Planner (actor).

- Provide a way for visually step through operations defined by specific classes.
• To detect shortcomings within an object–oriented design. By tracing the messages sent to an object, and even looking how long it takes to run the invoked method by simulation using Computer Aided Software Engineering (CASE) tools.

• Detection of complex application classes in design, which in turn will compel the drawing of state chart diagrams for those classes.

The design description for sequence diagram for Update use case is depicted for generic and specific cases on Figure 7.7 and Figure 7.8 respectively.

![Sequence Diagram for Update Use Case](image)

Figure 7.7: The UML sequence diagram for the generic design objects in Update use-case realisation.

When these sequence diagrams are compared with a collaboration diagram on Figure 6.9 (section 6.4 on page 38) the complexity of design model can be compared to the analysis model.

Consider the sequence diagram for the specific design objects on Figure 7.8:

**The Planner** uses the flood hazard management system through UpdateUI applet and the UpdateProcessing application to browse the existing layers in database. UpdateProcessing uses the UpdateRuleHandler to check the
7.3. Use Case design

Figure 7.8: The UML sequence diagram for the specific design objects in Update use-case realisation.

layers against the related UpdatingConfirmation, before the UpdateUI shows the list of Layers to the Planner.

The Planner selects a layer via the UpdateUI and the flood hazard management system earmark (schedule) the layer for updating, whereby the UpdateUI passes their request to UpdateProcessing. UpdateProcessing asks the ClipControl to schedule the updating by clipping of the layer. The ClipControl then clips the layer. The UpdateProcessing application then asks the LayerProcessing application to change the state of the layer to “updated on...”, with the date specified.

The differences between generic and specific design for the Update use case are depicted on Figure 7.7 and Figure 7.8 by tracing (on both diagrams) the flow of the processes. The differences is on the participating classes in the process; for generic one, other objects are encapsulated inside others. Comparison between Figure 7.7 and 7.8 shows that between UpdateUI and Layers classes in Figure 7.7 there are Update Processing and Layers Processing classes in Figure 7.8. These extra classes communicates between them after the updating process in order to fulfill the changing status process for the layer updating. This process may not be available (as in generic case) and yet the main process of updating could still be implemented.

When a specific case for flood hazard on update process is to be realised, say for the City of Köln or Koblenz (a suburb of Neuendorf / Rhein) in the German along The Rhine [13, 22], a generic sequence diagram can be used in designing the specific case. This shows how other scenarios can be accommodated for the flood hazard management system which was developed purposely for the Enschede Municipality, by utilising object-orientation paradigm (with UML as
a tool) in different cases with the same generic characteristics.

When specific scenarios are realised from a generic one, specific requirements on the subsystems are a prerequisite. In order to do this, subsystems and their interfaces are to be identified, this is carried out in the next subsection.

### 7.3.3 Subsystems and Interfaces Identifications

Subsystems are used to group the design model classes so as to control them, interfaces are used to define the context of the subsystem (actors, classes, and other subsystems).

#### Subsystems

Consider the two cases:

**Case 1** During top-down development, it may be necessary to capture the requirements on the subsystems and their interfaces before their internals are designed.

**Case 2** It should be easy to substitute a subsystem and its specific internal design with another subsystem that has another internal design.

In the cases above, use case realisation-design can be described at several levels in the subsystem hierarchy.

Low-level subsystems are called *Service Subsystems*, their classes realise a service which is manageable unit of optional functionality. These are used to model groups of classes that have the same properties, it is possible to install a subsystem in an existing system in its entirety [21].

During modelling of flood hazard spatial database system, the optional functionality of flood hazard for Enschede scenario will be on this system, this will make it possible to change the service subsystem when other scenario is to be dealt with.

This also gives opportunity to find and design the classes within a separate service subsystem for other natural hazard, say, earthquake hazard scenario at Bandung Earthquake Mitigation Project [39] and to fit it in the spatial database designed for flood hazard management.

The service subsystem can be designed either as a bottom-up or top-down, for the case of bottom-up, subsystems are based on classes already obtained (existing); subsystem package the classes into clearly defined functions. For the case of top-down, high-level subsystem and their interfaces are identified before any of the classes are realised [21].

#### Interfaces

Packages can be modelled in various ways, for this research work the approach of separating an interface from classes in separate packages is used [11]. With consistent naming conventions, the interface package will make certain aspects
of the architecture very visible in the flood hazard management system development structure.

Manipulating geographical information from a GIS user interface can be through any of the following [41]:

**Customised interface (end user)** End users access the information through a graphical user interface (GUI) customised for the application needs. Any evolution in the application or changes in parameters of the end user query requires the designer to interact with the system.

**Application development (designer)** Application developer use a programming language to customise the system and the GUI to the application needs.

**Query language (end user and designer)** High-level declarative query language is used, which enable user to express his needs through, say, a query list with various alternatives for specific function. These enables the query facilities to be accessible to non-expert users also to be independent of the software and platforms.

The query language user interface is ideal for the flood hazard system as per above advantages, a *Graphical User Interfaces* (GUI) will be designed on this platform.

**Graphical User Interfaces**

*Graphical User Interfaces* (GUI) determine system usability; its deployment turns the system’s planned physical architecture into a reality. GUI designing aims to convey the proper information in an uncomplicated and intuitive visual context [44].

A UML model of a window and a composite diagram in the form of a package that corresponds to the screen snapshots for LayerWindow use case are shown in Figure 7.9.

User interfaces serve as a mean to implement the use cases, in UML modelling a particular application’s window is presented as a composite of a number of controls. This is an advantage of using UML in modelling over usage of screen snapshots of GUI; with the plain screen snapshots one cannot capture the relationship between screens and use cases [44], but with UML modelled screen snapshots can be accommodated with more information.

A diagram that shows a connections of the screen snapshots elements to the use cases is advantageous in identifying what each screen element is supposed to do. Also it will help to ensure that all use cases are implemented in the final design. Figure 7.10 shows how the use cases are ordered in Layer update window.

After covering use cases interactions in design stage, the following section explores various classes as they are depicted in design stage.
7.4 Class Design

The aim of designing a class is to create a design class that fulfills its role in use-case realisations and the nonfunctional requirements that apply to it, this includes maintaining the design class itself and its contents. The contents are; operations, attributes, relationships, methods (which realises operations), imposed states, dependencies and requirements (for implementations).

UML class diagrams can be used for various purposes; e.g. understanding user requirements, describing design, and in each circumstances different styles has to be applied [2]. In this research work, class diagrams are used to describe the detailed design.

7.4.1 Outlining the Design Class

The identified design classes are denoted by tracing dependencies to the corresponding analysis classes they design, as shown on Figure 7.5 in section 7.3.1 on page 51.

In designing process, it follows that, when analysis classes (section 6.2 on page 31 and Figure 6.2) are given as inputs the following occurs:

- Designing boundary classes are dependent on the interface technologies in use.
7.4. Class Design

Figure 7.10: The flood hazard user interface models for the Layer Update use case window (modified from Lervik [27]).

- Designing *entity classes* that represent persistent information often implies using a specific database technology.

- Designing *control classes*, the following are considered; distribution issues, performance issues, and transaction issues.

### 7.4.2 Identifying Operations

Operations provided by the design classes, are usually described by using syntax of the programming language.
The UML supports four types of visibility for each operation or attribute, Table 7.1 shows the details of visibility on UML class diagrams.

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Symbol</th>
<th>Accessible To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>+</td>
<td>All objects within the system.</td>
</tr>
<tr>
<td>Protected</td>
<td>#</td>
<td>Instances of implementing class and its subclasses.</td>
</tr>
<tr>
<td>Private</td>
<td>-</td>
<td>Instances of implementing class.</td>
</tr>
<tr>
<td>Package</td>
<td>~</td>
<td>Instances of classes within the same package.</td>
</tr>
</tbody>
</table>

Visibility are important in design models, while in conceptual models they are optional as they will be public (+) always.

The inputs to the operations identification are:

- The responsibilities of analysis class that the design class traces to.
- The specific requirements of any analysis class that the design class traces to.
- The interfaces that the design class needs to provide.
- The use-case realisations - design in which the class participates.

### 7.4.3 Identifying Attributes

The attributes are identified and their description written using the syntax of the programming language. This is a UML advantage, of being able to accommodate any programming language. Attributes specify a property of a design class are often implied and required by the operations of the class.

### Identifying Associations and Aggregations

Design objects interact in sequence diagrams (Figures 7.7 and 7.8 in section 7.3.2), these interactions often require associations and aggregations between corresponding classes/objects.

**Association names** for this case are; consists, uses and needs. They are indicated with their directions. In UML this is done with a filled triangle. This indicates that the associations should be read as “an update consists of clip” and the other one “clip uses Layer”. When it is clear in which direction the name of an association should be read, the indication is not a necessity [2].

**Multiplicity**, for each of the three classes involved in a relationship to realise Update use case there is a multiplicity. In UML class diagrams, (*) is not used on its own. This is done in order to avoid ambiguity in representing (0..*) and (1..*). Multiplicity indicators are shown on Table 7.2.
7.4. Class Design

Table 7.2: UML multiplicity indicators (from [2]).

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0..1</td>
<td>Zero or none</td>
</tr>
<tr>
<td>1</td>
<td>One only</td>
</tr>
<tr>
<td>0..*</td>
<td>Zero or more</td>
</tr>
<tr>
<td>1..*</td>
<td>One or more</td>
</tr>
<tr>
<td>n</td>
<td>Only n (where n &gt; 1)</td>
</tr>
<tr>
<td>*</td>
<td>Many</td>
</tr>
<tr>
<td>0..n</td>
<td>Zero to n (where n &gt; 1)</td>
</tr>
<tr>
<td>1..n</td>
<td>One to n (where n &gt; 1)</td>
</tr>
<tr>
<td>n..m</td>
<td>Where n and m both &gt; 1</td>
</tr>
<tr>
<td>n..*</td>
<td>n or more, where n &gt; 1</td>
</tr>
</tbody>
</table>

Class names on the UML design classes should be based on commonly accepted terminology, making them easier to be understood by the stakeholders in the flood hazard scenario. Glossary of the class names are to be made when the need arise (section 5.3 on page 27; section 6.4 on page 38).

A detailed description of the associations and aggregation are shown on Figure 7.11 specifically for Updating process using Clip process which is a subtype class of Update class. Attached notes (shown in figure) in UML are helpful in adding more information or explanation to a class or particular item inside a class [44].

![Figure 7.11: UML modelling of association design classes for Updating using Clip process in flood hazard scenario.](image)

Description of Figure 7.11 Class Update consists of a Clip class which uses
Layer class in the flood hazard management system. For update realisation, an Update class needs a layer for the same.

One Update class may have one or more clip functions from Clip class depending on the type of clipping process to be carried out. This is depicted by multiplicity 1..* (one or more) on Update–Clip association.

A Clip class needs at least two layers (a clipper layer and the one to be clipped) from Layer class. The multiplicity, which is two or more (2..*), is on the Clip–Layer association.

In order an update process to occur, at least one layer is needed. This is shown by multiplicity of one or more (1..*) on Update–Layer association.

Describing States

A state is a stage in the behaviour pattern of an entity. Some objects are state controlled, that is, their state determines their behaviour when they receive a message. Statechart diagram are used to describe this, they are valuable input to the implementation of the corresponding design class. In UML statechart diagram an initial state is modelled with a filled in circle, while a final state is modelled with a filled in circle with a border around it (Figure 7.12).

![UML Statechart diagram for the updating process from Update class which is supertype of Layer class (modified from (21, 2))](image)

Figure 7.12: UML Statechart diagram for the updating process from Update class which is supertype of Layer class (modified from (21, 2))
Consider UML Statechart diagram (Figure 7.12) A layer to be updated is obtained from database and displayed. The layer is then scheduled to be updated by clipping method and its state change to scheduled. When the layer is updated its state change to updated, then it is registered as an updated layer and saved in the database. Finally, the state is changed to closed.

### 7.5 Discussion

It was observed in this chapter that it is advantageous in development process to identify *generic* sequence diagrams that can work as patterns and be used by several use-case realisations within the design model.

### 7.6 Conclusion

In this chapter, which covers design phase of development process, the specific parameters of flood hazard management system are realised; ready for implementation.

The output at this stage is flood hazard management design model, which preserve a structure of the system as depicted by the analysis model.

The following were realised from this chapter:

- Design and service subsystems and their dependencies, interfaces, and contents.
- Design classes, including active classes, and their operations, attributes, relationships, and implementation requirements.
- Use case realisation–design, which describe how use cases are designed in terms of collaborations within the design model.
- Deployment model describing network configuration.

In the next chapter, the design and deployment models obtained in this chapter will be the inputs. Specifically, design classes will be implemented by file components that contain the source code, and design and service subsystems will be implemented by *implementation subsystem*. 
Chapter 8

Development Phase

8.1 Implementation

Introduction

The purpose of implementation stage within a development phase in the flood hazard system development is:

- To plan the flood hazard management system integration.
- To distribute the flood hazard management system by mapping the components into nodes in the deployment model.
- To implement the design classes and subsystems found during design stage.

The workflow in the implementation comprises the following activities [21]: Architectural implementations, integrating the system, implementing a class, implementing a subsystem and performing a unit test.

The workflow gives the implementation model as an output. The implementation model comprises of the following elements: Subsystems, components and the architectural view of the implementation model.

The following sections cover these components for the flood hazard

8.1.1 Architectural Implementations

The implementation model is outlined together with its architecture by:

**Identifying significant components** Architecturally significant components are identified by initial outlining so as to initiate the implementation work.

**Mapping executable components into nodes** Executable components are identified per active class (realised during design stage) denoting an essential process. This may also include identification of files (containing source code or data) components that are required to create the executable components.
8.1. Implementation

Consider the following description for a process of Update use case realization in the flood hazard management system. With reference to the UML diagrams in Figure 8.1.

Figure 8.1: UML diagrams; (A). Active class identifies executable component; (B). Active object allocated to node with corresponding deployment component (modified from [21])

Description The executable component called *Update Processing* is identified from corresponding active class in design model known as *Update Processing* (Figure 7.6 on section 7.3.1). This shows how active classes are used to identify executable components in implementation model by using UML diagrams.

An active object of the *Update Processing* is allocated to the *Planner Server* node. As update processing component implements the *Update Processing* class, it is also deployed inside the *Planner Server* node. This shows how active objects allocated to nodes imply a corresponding deployment of components.

The mapping of components into nodes are very essential in flood hazard system development, and they are depicted in the architectural view of the deployment model [21].

After realisation of executable components, the flood hazard management system is then built incrementally in manageable steps. Each step gives small integration or test problem, the result of each step is known as “build”.

Build is a specific executable part/version of the flood hazard management system, the next section discusses about build integration plan in the flood hazard management system.

8.1.2 Integrating the System

The purpose of this stage is to create the build plan which describes the required builds in the flood hazard management system and their requirements. This builds are integrated individually before being subjected to integration
Chapter 8. Development Phase

tests. The initial builds start in the lower layers (e.g., the middleware and system software layer), then other builds expand to the application-general and application-specific layers [21]. This is because it is difficult to implement components in the upper layers, before those in the lower layers are in place and functioning properly.

The advantages of this incremental approach are:

- An executable version of the flood hazard management system is created early; instead of having to wait for a more complete version.
- Because only small and manageable part are added to/refined in a build at a time; shortcomings are easier to locate during integration tests.
- Integration tests are thorough than systemwide tests; because the attention is focused on smaller and manageable parts.

For each use case to be implemented, the following steps are to be followed:

1. The design of the use case are considered by identifying its corresponding use-case realisation-design (section 7.3 on page 50).
2. Identification of the design subsystems and classes that participates in the use-case realisation-design (section 7.3.3 on page 55).
3. Implementation subsystems and components that are required to implement the use case are identified in implementation model (found in step 2 above).
4. Consider the impact to be realised, when these requirements on subsystems and on components on top of the current build. Evaluate if the impact is acceptable according to the criteria specified. If so, implement the plan in the subsequent build, otherwise leave it for a future build.

8.1.3 Implementing a Subsystem

A subsystem is implemented for the purpose of ensuring that a subsystem fulfills its role in each build. This implies ensuring that the requirements are implemented in the build, and those which relates to subsystem are implemented by components or other subsystems within the flood hazard management system.

Consider the above figure; Each class in the design subsystem which is required in the build should be implemented by components in the implementation subsystem, the same holds for the other design subsystems contained in the design subsystem. Also, each interface by the design subsystem should be provided by the implementation subsystem (Figure 8.2). This implies that, the implementation subsystem must contain a component or implementation subsystem that provides the interface.

After realising the above, implementation of what is required by the components in the flood hazard management system and unit test on them can be carried out. What is required by the components is covered in next section; “Implement a class”, and unit test is described in section 8.1.5 on page 67.
8.1. Implementation

Figure 8.2: Design classes (A, B, C and D) for a build are implemented by components and an interface X provided by an implementation subsystem (modified from [21])

8.1.4 Implementing a Class

The aim of implementing a class is to implement a design class in a file component. Implementing a class consists of the following:

**Outlining file components**  Since the source code that implements a design class resides in a file component, therefore the file component should be outlined and its scope considered. For flood hazard management system, lets say Java is used, a file component will be having a “.java” extension for each class implementation. This implies the file components should support the compilation, installation, and maintenance of the flood hazard management system [21].

**Implementing the design class operations**  The operations defined by the design class are implemented by methods, the term methods denote the implementations of operations [21]. Examples of the methods in file components are methods in Java and Visual Basic [21]. In implementing an operation, choosing a suitable algorithm and supporting data structure is a necessity, and then actions required by the algorithms are coded.

**Interfaces**  This deals with ensuring that the components and the design class provides the same interfaces. The component obtained should provide the same interfaces as the design it implements (Fig 10.5)

**Maintenance**  This is about handling maintenance aspects of the implemented class; say, fixing the shortcomings when the class has been tested.
8.1.5 Performing a Unit Test

The aim of the unit test is to test the components as individual units, the following types of unit testing are done:

**Specification testing** This checks/verify the units external behaviour, it deals with verifying the component's behaviour without considering how that behaviour is implemented within the component.

**Structure testing** This checks/verify the units internal implementations, that is, to verify that a component works internally as intended.

**Other tests** These are specifically for some units; such as, tests on performance, memory usage, load, and capacity.

Integration and flood hazard management system tests are done to ensure that components behave correctly when integrated.

8.2 Test

**Introduction**

After implementation phase, the next phase is testing. Here the verifications of the result from implementation is done by testing each build. Among the outputs of the testing are **test model** (which includes test cases) and **evaluation** of the performed tests and defects [21].

**Test Cases** this deals with specifying what to test in the flood hazard management system. It specifies what to test with which input or result, and under which conditions to test.

**Test evaluation** is an evaluation of the result of the testing, such as test-case coverage, and the status of defects.

**Defect** is a flood hazard management system anomaly, it is used to capture a symptom of a problem in the flood hazard management system that is to be tracked and resolved. The detected defects are fed back to the concerned workflow within the flood hazard management system development process, such as design and implementation, for rectification/repair of the defects.

For this research work the testing workflow will involve design test and evaluation processes. These elements of workflow are discussed in the next sections, the whole process are to be implemented on the prototype created.

**Prototype**

Prototyping is used when a flood hazard management system is to be tested, a model is used to gain experience with the type of problems that can be expected when developing the full-blown flood hazard management system [19].
For a flood hazard management system, a sample data of Enschede Municipality can be used to determine whether it deliver the desired products and if the performance is acceptable.

The questions to be answered on this section are [44];

- Does the flood hazard management system perform as its supposed to?
- Further refinement necessary?

The work product at this stage is a test result.

### 8.2.1 Design Test

Designing tests are carried out in order to:

- Identify and describes test cases for each build.
- Identify and structure test procedures which directs how test cases are to be performed (Figure 8.3).

**Figure 8.3: The input and result of the design test (modified from [21]).**

**Integration Test Cases** are used to verify that the components interact with each other in orderly way after they have been integrated into a build. Integration test cases are derived from use-case realisations-design as the *use-case realisation classes* and *object interaction*. 
A set of test cases with a minimum overlap, each of which test a specific scenario through a use-case realisation are chosen.

Integration diagrams of the use-case realisation are considered as input, then combinations of actor input, output, and flood hazard management system start state that leads to scenario that employ the classes (and thereby components) that participates in the diagrams.

When the test is performed, actual interactions of the objects with the flood hazard management system is captured. Then comparison of actual interaction with the interaction diagram is done; they should be the same, otherwise, a defect is spotted.

**System Test Cases** are for testing that the flood hazard management system functions properly as a whole. Combinations of use cases under different conditions are tested, this conditions are; different hardware configurations (e.g. processors, hard disks), various number of actors, different sizes of database.

This test cases can be realised by considering the use cases, especially their flow of events and special requirements.

### 8.2.2 Evaluate Test

The parts of the flood hazard management system that are acceptable are isolated, and those which did not meet the quality criteria are revised and tested again. The output is a document, test evaluation description, which is about testing completeness, reliability, and suggested actions (Figure 8.5).

1. The integration test for a build, updating, is done manually by clipping the layer using digitiser and an experienced operator.
2. Comparison is done between test results and expected (manually) results.
3. The defects are studied to trace the component likely to contain the fault, and to evaluate the overall results of the flood hazard management system for the particular use case realisations.

Flood hazard management system testing is analogously performed as integration testing (see section 8.2.1 and Figure 8.4).
8.3 Conclusion

Consider a sequence diagram which is part of a use case realisation-design for the update use case Figure 8.6. Which is first part of the Figure 7.8 on section 7.3.2 on page 52.

A test case from a sequence diagrams describes how to test a particular sequence through the diagram (update use case for this case). This is done by;

- capturing the required start state of the flood hazard management system,
- the input from the actor, and
- other parameters that makes the sequence to happen.

8.3 Conclusion

The flood hazard management system at implementation stage within development phase is implemented on the appropriate hardware and integration of these processes is done by using the UML diagrams on object–oriented platform. The implementation model is the primary input to the testing activities
for the development process, each build is tested individually. The next section covers the testing phase of the developed flood hazard management system.

The implementation model input are tested on test stage of development phase, this activities are vital for the flood hazard management system confirmation and/or commissioning. This testing process can be iterated so as to rectify the shortcomings within the flood hazard management system, and this compels for each build to be tested individually.

From this chapter it is realised that, a good and thorough analysis and design is essential in flood hazard systems development. The object-oriented principles modelled using UML can accommodate codes from any platform which support object-orientation, and they can be tested before being used for real production.

The advantage of starting with generic case is that, the specific cases can be modelled during iterations as the flood hazard management system is designed for particular case having the same generic platform. This will be realised by manipulation or addition of use cases during iterations.

Instead of using a much involving general GIS system. In later stages of the system growth, other specific use cases will be able to be added to fit particular cases.

At this stage, the flood hazard management system framework is realised and the essential part is to do programming and component implementation. This involves making choices of the suitable use cases for a particular scenario. The realised framework structure if maintained, will assist in exchanging or adding components in smooth way when the need arise.
8.3. Conclusion
Chapter 9

Conclusion

The object–orientation modelling was used to build flood hazard management system through distinct processes. The modelling using UML joined the processes in flood hazard management system development. UML was used as a tool to join the different stages of development processes seamlessly, these stages were requirement, analysis, design and development phases.

In this research it was realised that; conceptualisation of the flood hazard system using the object-orientation paradigm is independent of any programming process that may follow after modelling. The object–oriented modelling emphasis is on understanding the object components, their attributes, and the relationships they bear with each other. This is advantageous than having a direct approach to physical modelling, say, using visual basic to solve the user needs.

Instead of using a much involving general GIS system, the specific cases can be modelled using object–orientation platform on flood hazard management scenario to provide to the user specific flood hazard scenario needed. The advantage of starting with generic case is; the specific cases can be modelled during iterations as the flood hazard management system is designed for particular case having the same generic properties. This is realised by addition of use cases during iterations, or manipulation of existing use cases.

In this research various UML diagrams were used, in object–orientation paradigm every piece of object information is of its importance; selecting information for emphasis and clarification is an important communication activity in itself, otherwise it would have been a messy setup.

While using UML in this research it was observed that:

- The complexity of the flood hazard management system, implied good modelling technique are very important.
- Use case diagrams are to be prioritised, then collaboration diagrams and sequence diagrams.
- UML diagrams are helpful in communication from phase to phase in development process and more thorough analysis.
- UML should be used selectively, so as to be effective in conveying essential information.
• It is not beneficial even economical to use every available UML diagrams!

In the course of the research, the Enschede Municipality area was considered purely on practical considerations, and not on a detailed or scientific study as intended before. This situation left no room for use of real data and information of Enschede Municipality to realise a prototype of the research.

To realise a maintainable flood hazard management system, the structure of the analysis model obtained in (chapter six) can be preserved during design stage in (chapter 7). This will include components that can be reused in the system.

The flood hazard management system for local level was not realised in fully, the realised output for this research was a skeleton of the flood hazard management system for a local level, on which different use cases can be dealt with by programming and introducing components.

System development is very wide, and I thought that I could do everything myself; that is, assuming the role of a system analyst, architecture, and designer. I was wrong, as the work was very involving and I was time barred in that pursuit.
Appendix A

System Development

A.1 Actors Generalisation with their Details

**Disaster Analyst** Represents a person from disaster preparedness agencies, safety department and governing bodies. This person is responsible for carrying out qualitative and quantitative assessment of the flood hazard present in the study area and identify critical segments in the development activities.

**Emergency Officer** A person from either the police and fire departments or the offices of emergency services. They are responsible for flood hazard preparedness activities and rescue missions.

**External Package** This comprises of an application software and external hardware which are outside the flood hazard management system. An application software is used in manipulation other than those covered by GIS package. The external hardware are those outside the flood hazard management system, for instance a printer.

**Facilitator** A person who is responsible for maintaining the flood hazard management system and ensures that it runs smoothly [19].

**GIS Package** A system capable of combining layers from spatial database to create maps and data sources. These can support landuse planning, risk and vulnerability assessment, disaster forecasting and hazard management [1]. For flood hazard management at local level the package should have a simple interface, and strong analytical and graphical capabilities [36].

**Planner** A planner represents a person from any of these sectors: building, engineering, and planning departments; development assistance agencies, private consulting firms; provincial and district councils; public works and highway departments. This person or organisation is responsible for planning and implementing various development projects within the region.
A planner uses the flood hazard management system to acquire various layers and data in order to plan the projects inline with flood-hazard control measures.

### A.2 Use Cases with Details

**StartUp** This use case is used by DisasterAnalyst, Planner or EmergencyOfficer to start the flood hazard management system.

This is the first use case to be initiated in the flood hazard management system, it has to check all the flood hazard management system components whether they are functioning. Also the username and password of the actor are to be checked.

**Termination** This use case is used by DisasterAnalyst, Planner or EmergencyOfficer when they exit the GIS Package or the flood hazard management system.

The actor initiates the use case when the flood hazard management system or GIS Package is closed; the flood hazard management system inquires whether to save any unsaved document/layer before termination.

**Maintenance** This use case is responsible for maintaining the flood hazard management system and ensuring it runs smoothly. This use case is used by the Facilitator.

**Save** This is used by DisasterAnalyst, Planner or EmergencyOfficer when they need to save the created layer or edited information/data.

The initiator is required to specify the saving folder and to create the name of the object to be saved.

**Print** The user are DisasterAnalyst, Planner or EmergencyOfficer when they need a hardcopy of specific layer or information.

The use case will require printing specifications from the initiator and selection of hardware on which the hardcopy is to be printed.

**Analyse** This use case is for spatial analysis operations that create new spatial datasets from existing ones. The tools in this use case are buffers, clip, delineate and overlays (intersect and union).

The user of this use case are DisasterAnalyst, Planner and EmergencyOfficer.

**AnalyseTable** This deals with table manipulations, it can either join (appends attributes) or relate (does not append attribute) tables. These tables are layer attribute (spatial) tables or non-spatial ones [37].

The DisasterAnalyst, Planner and EmergencyOfficer are the actors in this use case.

**Evaluate** This use case deals with manipulations so as to realise quantitative information from the data. The use case Evaluate is used by DisasterAnalyst, Planner and EmergencyOfficer.
**Update** This use case deals with adding or editing data/information on a layer so as to update it. This actors on this use case are DisasterAnalyst, Planner and EmergencyOfficer.

**Open** The use case Open is used by DisasterAnalyst, Planner or EmergencyOfficer when they need to access to GIS Package, data or information from the flood hazard management system. When this use case is initialised it prompts the initiator to specify the type of data or information to be opened, for the case of GIS Package this dialogue is not displayed.

**Select** This use case is used by the Planner to select a particular area from a displayed layer: relationship, reference layer, distance layer. Before this use case can be initiated, the planner has already opened a layer which was delivered by another use case called Display.

1. Planner studies the display to check the area of interest.
2. Planner select the area of interest.
3. GIS package prompts a planner whether the selected layer is to be saved.

**Display** This use case deals with projecting the chosen layer, data or information on the monitor. The actors are DisasterAnalyst, Planner and EmergencyOfficer.

### A.3 Activity Diagram

In order to gain an understanding of the user’s business process and the client domain for the flood hazard management system; an activity diagram is an essential input of this development process.

The activity diagram, shown in parts, depict the following useful objects found in generic flood hazard management scenario [35, 36, 13]:

**Layers** Topographical base map, land use zones, remote sensing images, flood plains, flood prone areas, lifeline network layer, critical facilities layer, geomorphology layer and geological layer.

**Data/Information** Slope/aspect data (DEM), flood information (season, duration, severity), damage report, river flow capacity, statistical data.

**GIS Tools** GIS procedures and programs for collecting, organising, analysing and presenting the data and information.

### A.4 Class Diagrams

The following are class diagrams showing their participating classes and their relationship in various use-case realisations.
Figure A.1: Part one of UML activity diagram for generic flood hazard management scenario.
Figure A.2: Part two of UML activity diagram for generic flood hazard management scenario.
A.4. Class Diagrams

Figure A.3: Part three of UML activity diagram for generic flood hazard management scenario.
Appendix A. System Development

Figure A.4: A class diagram for realisation of Display use case.

Figure A.5: A class diagram for realisation of StartUp use case.

Figure A.6: A class diagram for realisation of Analyse use case.
A4. Class Diagrams
Appendix B

UML Diagrams

UML uses twelve types of diagrams, which are in three categories [17]:

**Structural Diagrams** These represent static application structure, they include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

**Behaviour Diagrams** Different aspects of dynamic behaviour are represented by these diagrams, they include the Use Case Diagram, Sequence Diagram, Activity Diagram, Collaboration Diagram, and Statechart Diagram.

**Model Management Diagrams** Ways of organising and managing an application modules are represented by these diagrams; they include Packages, Subsystems, and Models.

### B.1 Class, Objects and Models

![UML Diagrams](image)

A class is represented in the UML as a rectangle. The class is divided into three compartments, the top one showing the class name, the second one showing the attributes and third one showing the methods.

An object look very similar to a class, except that its name is underlined.
B.2 Analysis Class Stereotypes

UML packages are represented graphically as a folder. A package divides and organise models in much the same way that directories organise file systems. Each package correspond to a subset of a model and can contain classes, objects, relationships, components or nodes.

B.2 Analysis Class Stereotypes

The basic stereotypes of analysis class are used to distinguish concerns among different classes, Figure B.2.

The stereotypes used in this research are:

**Boundary** used to model interaction between the flood hazard management system and its actors, that is, users and external system.

**Control** it represent coordination, sequencing, transactions, and control of other objects, and that often is used to encapsulate control related to a specific use case.

**Entity** it is used to model information that is long-lived and often persistent.

B.3 Component Stereotypes

Some standard stereotypes of components are:

- **<< executable >>** is a program that may be run on node.
- **<< file >>** is a file containing source code or data.
- **<< library >>** is a static or dynamic library.
- **<< table >>** is a database table.
- **<< document >>** is a document.
Appendix C

Glossary

**abstract class** A reusable object–oriented design for a component. It defines the interface of a class and the tree of subclasses that can be derived.

**actor** A system user or an external system that collaborate with the system.

**architecture description** An architectural view of the use-case model, depicting the architecturally significant use cases.

**artifact** Any kind of description or information created, produced, changed, or used when working with the system.

**behaviour** The part of the class or object definition that describes what the class or object does.

**C++** A general purpose programming language with a bias towards systems programming that supports data abstraction, supports object–oriented programming and supports generic programming.

**class** A generic description of a group of objects that will have a state and behaviour.

**class diagram** A graphical representation of one or more classes and their relationship to one another.

**component** A package of software artifacts that can be independently developed and delivered as a unit and that can be composed, unchanged, with other components to build something larger.

**data mining** The process of finding new and potentially useful knowledge from data.
**encapsulation** The mechanism which separates a variable from code which changes its state. Encapsulation is one of the central process in object-oriented modelling and tends to simplify the modelling process by binding unnecessary details.

**entity relationship/data modelling (ERD)** A modelling technique which put emphasis on the data entities.

**extension** A type of inheritance whereby a new subclass is used to add to the functionality of an existing class.

**field** Part of a class or object that can be used for storing an element of its state, whether variable or constant.

**floodplane** Areas which are relatively flat and lies adjacent either to a river, channel, or stream that is susceptible to flooding.

**flood-prone areas** The Actor is a system user or an external system that the system interacts, thus actors represent outside parties that collaborate with the system.

**framework** A reusable object-oriented analysis and design for an application or subsystem.

**functional requirement** Is a requirement that specifies an action that a system must be able to perform without considering physical constraints.

**generalisation** A type of inheritance where common functionality from two or more classes are extracted and placed in their own superclass.

**glossary** Definition of important and common terms used in system description, it is useful in avoiding confusion regarding the definition of different concepts and notions.

**graphical user interface (GUI)** A special case of user interface whereby interaction is facilitated through the use of graphics.

**inheritance** The process of obtaining properties through a relation such as parent-child or general-specific; it is the act of reusing properties of a superclass.

**interface** The result of abstraction process whereby a class is described by what it does rather than how it does it.

**iteration** The process of repeatedly perform a task.
java database connectivity (JDBC)  Java DataBase Connectivity is open stan-
dards middleware, it allow simultaneous secure, real time, read/write ac-
access to the database.

JDBC™ technology is an application programming interface (API) that
lets you access virtually any tabular data source from the Java™ pro-
gramming language. It provides cross-DBMS connectivity to a wide range
of SQL databases, and now, with the new JDBC API, it also provides ac-
cess to other tabular data sources, such as spreadsheets or flat files.

message Data passed either to or from an object.

method A part of a class that represents a particular aspect of its beha-
viour.

methodology A collection of procedures, techniques, tools, and documen-
tation aids, which helps the system developers to implement informa-
tion system.

node The main hardware item, a generic name for any kind of computing
resource.

object A clearly defined model of something that has a state and beha-
viour; the encapsulation of data together with procedures for the data
manipulation.

object identity A handle which distinguishes one entity from another, it
is internal to an object, it provides a way to represent the individua-
lity of an object independently of how it is accessed.

object-oriented modelling A technique used to analyse and communi-
cate the behaviour of the objects found in the system.

object variable An object variable is one that is declared inside a class
but outside any method definition. This makes it scope available to
all methods within a class.

open database connectivity (ODBC)  Microsoft’s industry-standard da-
ta access interface, it provides; unified, secure, and real time access
to the relational data as part of the OLE DB specification.

openGIS Set of open specifications from the Open GIS Consortium.

open gis consortium (OGC) An international industry consortium
of companies, government agencies and universities participating in
a consensus process to develop publicly available geoprocessing spe-
cifications.

object linking and embedding database (OLE-DB) Microsoft develope-
d open middleware standard, that offers applications, compilers,
and other database components efficient access to Microsoft and third-
party databases.
**package** A collection of related classes that form a coherent group of some kind.

**requirement** A condition or capability to which a system must conform.

**reusability** The degree to which the same program code may be used in a variety of contents.

**risk** The number of human lives lost, casualties, damage to properties and effects over the economic activity due to occurrence of a hazardous event.

**snapshot** Drawing depiction of a set of objects and the values of some of their attributes at a particular point in time.

**state** The part of a class or object definition that describes what information is contained within the class or object.

**static variables** Also called a ‘class variable’, this is a field or method whose contents are associated with an entire class rather than a specific object.

**structured analysis** A functional–based modelling technique which put emphasis on the functions (behaviour) within the system.

**subclass** A class that inherits the properties (state and behaviour) of a parent class, called the superclass.

**superclass** The ‘parent’ of a subclass. All public or protected methods and variables of a superclass are available to any subclass that inherit it.

**system architecture** A structure of a system into its main sequential components that communicates through dataflow and events.

**use case** A sequence of actions that a system can perform, interacting with actors of the system.

**use-case model** A model of a system containing actors, use cases and their relationships.

**user interface** The means by which communication between the user and computer is enabled.

**visual basic (VB)** Microsoft programming language which is able to create powerful and complex application.

**vulnerability** The degree of loss to a given element at risk or a set of elements, resulting from the occurrence of a hazard.

**workflow** A sequence of ordered activities, in which one activity produces an output that is an input for the next activity.
Bibliography


## Index

Actor, 88  
Actors  
  generalisation, 26  
  system category, 26  
  user category, 26  
Analysis, 31  
  class stereotypes, 32  
  model, 31  
  products, 42  
ArcInfo, 51  
Build, 66–67  
CASE, 55  
Data  
  mining, 47  
Database, 1  
DBMS  
  integrated, 51  
  loosely coupled, 51  
Defect, 69  
Design  
  architectural, 45–52  
  class, 59–64  
  conceptual, 11  
  logical, 11  
  physical, 11  
  requirement analysis, 11  
  top-down, 11  
  use case, 52–58  
EIMS, 6  
El Niño, 6  
Encapsulation, 12  
FEMA, 47  
Flood hazard, 9–10  
  flood-prone areas, 9  
  floodplane, 9  
Information  
  natural ecosystems, 6  
  natural hazard, 5  
  population and infrastructure, 6  
Inheritance, 12  
Interface  
  application developed, 58  
  customised, 58  
  GUI, 58  
  query language, 58  
Java  
  applet, 49  
  AWT, 49  
  RMI, 49  
  virtual machine, 49  
JDBC, 50–51  
LAN, 47  
Lifeline networks, 9  
  maps, 10  
Methodology, 15  
  Fusion, 19  
  Grady Booch, 17  
  GRAPPLE, 19–22  
  OMT, 17  
  OOSE, 15  
  Unified Process, 19, 22–24  
Mitigation measures, 9  
Natural hazard  
  management systems, 1  
  spatial database, 1  
Object  
  identity, 12  
  mining, 47  
Object–Orientation, 8, 12, 12–13  
  advantages, 13
methodologies, 15–24
ODBC, 50
OGC, 47
OLE-DB, 50
Oracle8i Spatial, 52
Postgres, 52
Spatial
database, 1
DBMS, 51–52
Subsystems, 57
service, 57
bottom-up designed, 57
top-down designed, 57
TCP/IP, 47
Test
cases, 69
evaluation, 69
TiGRis, 51
UML, 13
advantages, 13
deployment diagrams, 45
statechart diagrams, 63
behaviour diagrams, 13, 85
class diagrams, 54, 59
model management diagrams, 13, 85
sequence diagrams, 54
structural diagrams, 13, 85
visibility, 60
Unit test
specification, 69
structure, 69
Use case, 27
realisation analysis, 33
WAN, 47