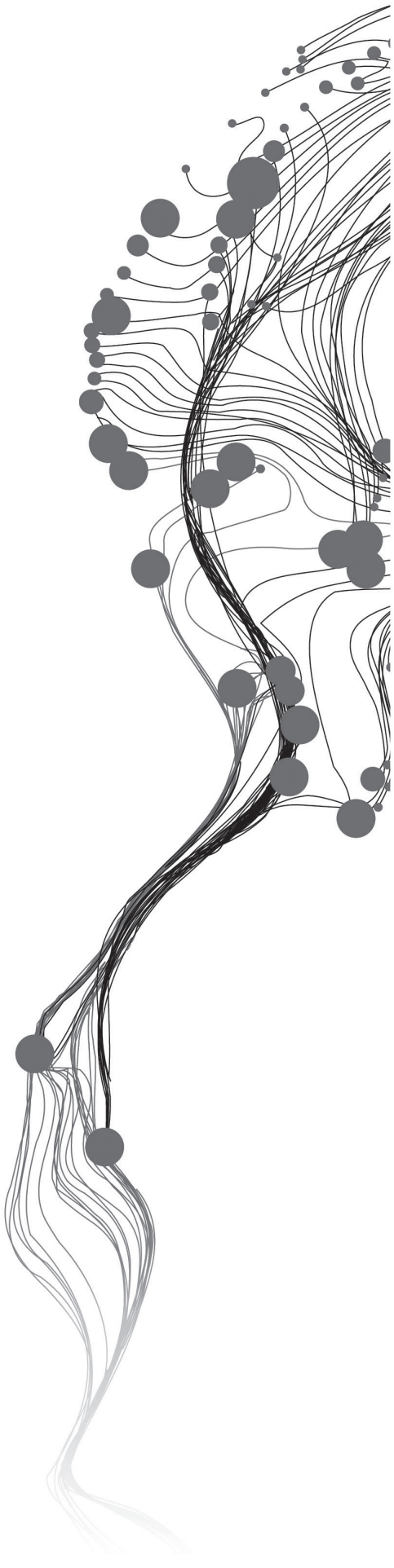
A DESIGN METHOD FOR THE DEVELOPMENT OF CYBER-APPLICATIONS

TIGIST ABRAHAM GIZAW
February, 2013

SUPERVISORS:

Dr. Javier Morales
Drs. B.J. Barend Köbben



A DESIGN METHOD FOR THE DEVELOPMENT OF CYBER-APPLICATIONS

TIGIST ABRAHAM GIZAW
Enschede, The Netherlands, February, 2013

Thesis submitted to the Faculty of Geo-information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: GFM

SUPERVISORS:

Dr. Javier Morales
Drs. B.J. Barend Köbben

THESIS ASSESSMENT BOARD:

Dr. A.A. Voinov (chair)
External examiner: J.J. Verplanke MSc

ABSTRACT

Most real world problems can be resolved by assembling existing data and components into the necessary application. In some cases, all the required components are available in the development phase of the application. However in many other cases, new components need to be incorporated into an already executing application due to the dynamic nature of the problem. These kinds of applications are called Cyber-applications. These applications are the most recent developments of web application which are data intensive and highly interactive. It is widely recognized that development of any web application requires a sound design method. This is why there exist several methodologies to guide the development process. However, these methodologies mainly focus on the design of the navigation and the presentation of data. This means, almost all of these design methods emphasize mainly on visual design and user interface aspects. In the contrary, the development of Cyber-application requires a different design method since the main focus is the processing and analyzing of new data source that are discovered at run-time. Therefore, in this research project, we present a formal design method for the development of Cyber-applications. This methodology incorporate the principles of Separation of concern (SoC), application partitioning based on the Model-View-Controller (MVC) design pattern, run-time data and component integration using the Dynamic Data Driven Application System (DDDAS) method, external resource exploitation, and communication between the partitioned application modules for the ease of the development process. Additionally, a prototype for quantitative multi-hazard analysis is built to demonstrate the basic concepts in the design method.

Keywords

SoC, MVC, DDDAS, Event-based interaction, Design method, Cyber-applications, SQL/MED, Geospatial CyberInfrastructure, WebGIS, Cloud Computing

ACKNOWLEDGEMENTS

First and foremost, I thank the almighty God for the wisdom and determination that he has been bestowed upon me during this research project. Next, I would like to pass my deepest gratitude to my first supervisor Dr. Javier Morales for his useful comments, excellent guidance and remarks all the way through the learning process. Thank you for allowing me to grow as an independent research scientist. I would also like to thank my second supervisor Drs. B.J. Barend Kobben for his kindness and valuable advices.

A special thanks to my beloved brother Netsanet Abraham. You have been a tremendous mentor for me throughout my life. Thanks for all the caring, encouragements, suggestion, and technical helps you provided me especially during this research project. Love you brother. Most importantly, I would like to deliver all the love I have in my heart to my sweetest, and most wonderful parents. Mom and Dad, you gave me birth, you gave me life, and everything else is in between. Because of you I always thrive to be better. Nani and Baby, you guys are hilarious. Love you both to bits.

Last but by no means least; I would like to express my special appreciation to Mulugeta Tadesse for cheering me up and stood by me through the good and bad times. Many thanks for always being there and for all the support :). Additionally, Thanks to the music, bases of my work, comfort and painless hits.

"*Everything should be made as simple as possible, but not simpler.*"... **Albert Einstein**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Listings

List of Acronyms

AJAX	Asynchronous Javascript and XML
API	Application Program Interface
ASP	Active Server Pages
CBD	Component-Based Development
CI	CyberInfrastructure
CIM	Computation-Independent Model
CSS	Cascading Style Sheet
DaaS	Data as a Service
DAL	Data Access Layer
DBMS	Database Management System
DDDAS	Dynamic Data Driven Application System
DFS	Data Feed Service
ESRI	Environmental System Research Institution Inc.
GCI	Geospatial CyberInfrastructure
GEOSS	Global Earth Observation System of Systems
GI	Geographic Information
GIS	Geographic Information Systems
GPS	Geographical Positioning System
GSDI	Global Spatial Data Infrastructure
HDM	Hypertext Design Model
HMBS/M	Hypermedia Model Based on State/Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IIS	Internet Information Services

IT	Information Technology
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View View Model
NASA	North American Space Agency
NDT	Navigational Development Techniques
NSF	National Science Foundation
OGC	Open Geospatial Consortium
OO	Object Oriented
OO-H	Object Oriented Hypermedia
OOHDM	Object-Oriented Hypermedia Design Method
PaaS	Platform as a Service
PIM	Platform-Independent Model
PSM	Platform-Specific Model
QGIS	Quantum GIS
RMM	Relation Management Methodology
SaaS	Software as a Service
SDI	Spatial Data Infrastructure
SOA	Service-Oriented Architecture
SoC	Separation of Concern
SQL/MED	Management of external Data
SQL	Structured Query Language
UI	User Interface
UML	unified modeling languages
UWA	Ubiquitous Web Application
UWE	UML-Based Web Engineering
WAP	Wireless Application Protocol
WebGIS	Web Geographic Information System
WebML	Web Modeling Language
WISDOM	Web Based Information System Development with a comprehensive Methodology

WFS	Web Feature Service
WML	Wireless Markup language
WMS	Web Mapping Service
WSDM	Web Site Design Method
WWW	World wide Web
XHTML	Extensible Hypertext Markup Language

Chapter 1
Introduction

1.1 MOTIVATION AND PROBLEM STATEMENT

Most interactive web applications consist of distributed components that access data, encapsulate business rules, and handle user interaction to address some specific real world problem. Identifying the kinds of components commonly found in distributed software solutions is usually conducted at the early stage in the process of application design [30]. Software components intended to be integrated into and distributed as an integral part of an application that dynamically incorporates new data and functions to their execution engine at run-time are referred as Cyber-applications. The term "Cyber" means everything that we need is provided by third-parties and it is available somewhere in the web. Therefore, Cyber-applications support CyberInfrastructure environment and provide the flexibility to manipulate external data and exploit computational resources that are available online.

Cyber-applications are data intensive and highly interactive application which runs in the web environment [39]. They provide support for the manipulation and processing of large volume of data for domains of scientific research. These data can be structured or unstructured which needs to be processed, analyzed, and linked for some sort of purpose. Extremely high storage and computational requirements, robust architecture, scalability and security are common characteristic of these applications [17]. Additionally, these applications are free-standing and highly interactive as they allow web pages to be displayed in standard web browsers to present responsive user interfaces.

There exist variety of applications which present interactive user interface in the web such as email communications, search engines, multimedia and community, encyclopedias, and basic office applications. However, we do not consider them as cyber-applications. Cyber-apps are different in the sense they enable the processing of complex tasks in the web environment and they also have the capability to dynamically integrate new components into the application at run time. Addressing the potential of these application requires available tools and assists for the composition of the application [9]. The support of recent web engineering tools such as .Net, Java, Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML)5, Asynchronous Javascript and XML (AJAX), Ruby and Cascading Style Sheet (CSS) plays vital role in the development of cyber-application.

Cyber apps are the latest advancement along the development path of web applications. They allow the exploitation of off-the-shelf online resources by using a clear separation of concerns in the assembly of the application's structure. Separation of Concern (SoC) is a design principle for dividing an application into distinct modules, such that each component or modules deals with a separate concern [38]. When concerns are well separated, individual modules can be developed and updated independently. The identified components provide certain functionality and are accessible through a well-designed interface. There are certain architectural patterns which implement this principle for partitioning the application into different modules and also useful when dynamically incorporating new data and functions into the application.

Cyber-applications can be implemented for real world problems which require dynamic and real time integration of large amount of spatially referenced data and simulation. Flood risk assessment, landslide prediction, wildfire detection, and weather forecasting are some of the few scientific research areas where the application can be adopted. Therefore, the development process of these applications requires paradigms such that components can easily be designed and integrated into the application depending on the chosen scenario. Application paradigms are design patterns or frameworks which simplify the construction of the application process. Selecting appropriate paradigm and their underlying architecture leads to effective development of cyber-applications but most importantly, it is essential to adopt a standard design methodology for the development of cyber-application.

The absence of a proper design method for the development of cyber-applications is one of the problems that need to be addressed. Having a well-established formal design method will reduce the complexity of the development process and also increase performance and quality of the application [36]. Currently, there are no standard design methods for the creation of cyber-applications since these applications are the most recent application type in the development path of web applications. Designing such a methodology will support independent exchange of data between interacting modules and facilitate the computational model of cyber-applications. It will also make the work of developers simple by distributing responsibilities over different modules of entire applications.

Limited number of application paradigms for the development of cyber-applications is another problem in the development process. Selecting the appropriate application paradigm based on the requirements and nature of cyber-application is important in reducing the complexity and satisfying the pre-requirements of the application. A typical cyber-application requires partitioning the entire application into modules with clearly defined interactions between them [39]. This indicates that the bases, which are modularity and extensions needs to be preserved by the application paradigm that is expected to be selected or developed.

Another issue that we need to consider when developing cyber-application is the extra level of complexity that is inherently generated by the approach or the methodology we are following. Most applications have certain level of complexity and we would introduce an extra level of complexity to cyber-applications by separating the concerns. This additional level of complexity is a natural consequence of the approach, arises from fulfilling the requirements of separating the application into parts. Therefore, there has to be a mechanism to identify and manage the complexity that is introduced during the development of cyber-applications.

1.2 RESEARCH IDENTIFICATION

The scope of the methodology which is proposed in this research project is defined from user's perspective. The development path of cyber-application starts with the application problem and come up with the application itself. Our target is not the application; rather it is the methodology to guide the development process. Therefore, the methodology includes identifying those problems, setting up design principles and steps, defining the underlying architecture and building a prototype for validating the concept. The proposed methodology enabled us to identify the extra level of complexity that is introduced in the development process and found a mechanism to manage it in the application in lined with the design criteria. Additionally, we assumed that there exist various dataset which are suitable for a given domain of problem and transformation process on the datasets is not included on the scope of this methodology.

The proposed methodology begins with identification of problem domain to define requirements of the application which includes data requirements, their formalization, identification of

data sources, and specification of application workflow. Followed by, the design and the development of cyber-applications. With this in place, we identified the steps that are involved in each phases of the development process and this enabled us to define a well structured and robust methodology. The concerns addressed here are described through the following research objectives and research questions.

1.2.1 Research objectives

The aim of this research project is to define a design method for the creation of cyber-applications. This general objective can be achieved by defining the following sub-objectives:

1. To select appropriate application paradigm that is capable of identifying components for the development of cyber-applications.
2. To define an underlying architecture which allows dynamic incorporation of functional and data components into the application's execution engine at run-time.
3. To validate the correctness and completeness of the designed method.

1.2.2 Research questions

1. How to define a methodology for the development of cyber-application?
2. What are the types of problems that can be addressed with the application?
3. Which application paradigm can we use to identify the components in the development of cyber-applications?
4. How to design the underlying architecture for cyber-applications based on the design pattern?
5. How to validate the correctness and completeness of the design method for cyber-application?

1.2.3 Innovation aimed at

The design method incorporates the opportunities for cyber-applications to integrated spatial data which are necessary to understand and forecast environmental and physical processes that are changing the surface of the Earth. The methodology will guide users to address the design process properly and enable them to build the application with only a small modification depending on the nature of the problem. We used some components of the system as a prototype to validate the methodology designed.

1.2.4 Related work

As discussed in the previous sections, cyber-applications are innovations of recent web application technologies. There has not been much of research done in this area. Therefore, there are no particular design methods or defined underlying architecture to follow in the development of typical cyber-applications. The challenges in the development of complex web applications, and also problems arises from the absence of disciplined approach are discussed by [40]. However, several development approaches for web applications are being presented to developers. Some of these web methodologies are discussed in the next paragraph.

Hypertext Design Model (HDM) expresses the main abstractions of a hypermedia application, their internal structure and navigation, and application-wide navigation requirements [20].

Since this is the first method proposed for hypertext, it has been used as a basis for other methodologies. The most widely accepted Object-Oriented Hypermedia Design Method (OOHDM) approach is an extension of HDM which mainly focused on navigation design, interface design, conceptual modeling and object oriented implementation of the application [46]. The Web Site Design Method (WSDM) of [10] follows "user-centered" approach composed of User model design, conceptual model design, implementation design and implementation. Relation Management Methodology (RMM) approach of [49] consists of seven parallel and iterative executed stages. UML-Based Web Engineering (UWE) approach of [29] wraps up the fundamental life-cycle of web application development based on unified modeling languages (UML). The Hypermedia Model Based on State/Model (HMBS/M) separate the content and structure using formal representation techniques based on state charts and models [51]. Object Oriented Hypermedia (OO-H) is derived from Object Oriented (OO) approaches and extends modeling and implementation process of web applications based on UML [22]. The W2000 method is based on a specific meta-model and allows the designer to model many aspects of web applications [3]. This model supports flexible definition and extension of other modeling concepts. The Web Modeling Language (WebML) method of [7] enables designers to create the basic features of a high level web application without defining architectural details. It provides high level description of a web system consisting of a data model, hypertext model, presentation model and personalization model.

The central attention of most approaches that are mentioned above is the navigation and presentation of web applications. They do not consider a clear separation of user interface, data and functionality of the application. Unlike most web applications, "cyber-applications focus mainly on the manipulation of user activities, which may result in the generation of data, or asynchronous access to a new data source" [39]. Therefore, the design method proposed here mainly focuses on the clear separation of concern and components interaction to manage the exchange of data and complexity of the application development process.

1.3 METHOD ADOPTED

In order to carry out the research objective and find possible solution for the research questions, we splited the research project into phases. The first phase is requirement analysis and specification, followed by application design. Next is defining the proper methodology, and finally undertaking the prototyping and validation phase concludes the whole research project. The overall approach adopted in this research project is shown in Figure 1.1.

Requirement analysis and specification is mostly conducted at the primary state of any software and web application development techniques. In this stage, we reviewed basic concepts of data intensive applications, separation of concern, exploitation of resources in real time and components interaction to understand the design principle and requirements of cyber-applications. In parallel, we identified a suitable problem area in which the proposed methodology is validated through. After the requirements are identified, analysis based on those requirements has been performed. The analysis includes identifying which components are needed and how the information flows between these components, what kind of resources are available and what types of problem that can be addressed with the application.

The development of cyber-applications requires design principles which enforce the partitioning of entire application into different modules and defining the interaction between them for better provision of simple user interface, content and application logic. Therefore, on the application design phase, previously specified components are distinguished, and application paradigm and underlying architecture are selected.

Since separating the internal structure of the applications results in different components, the

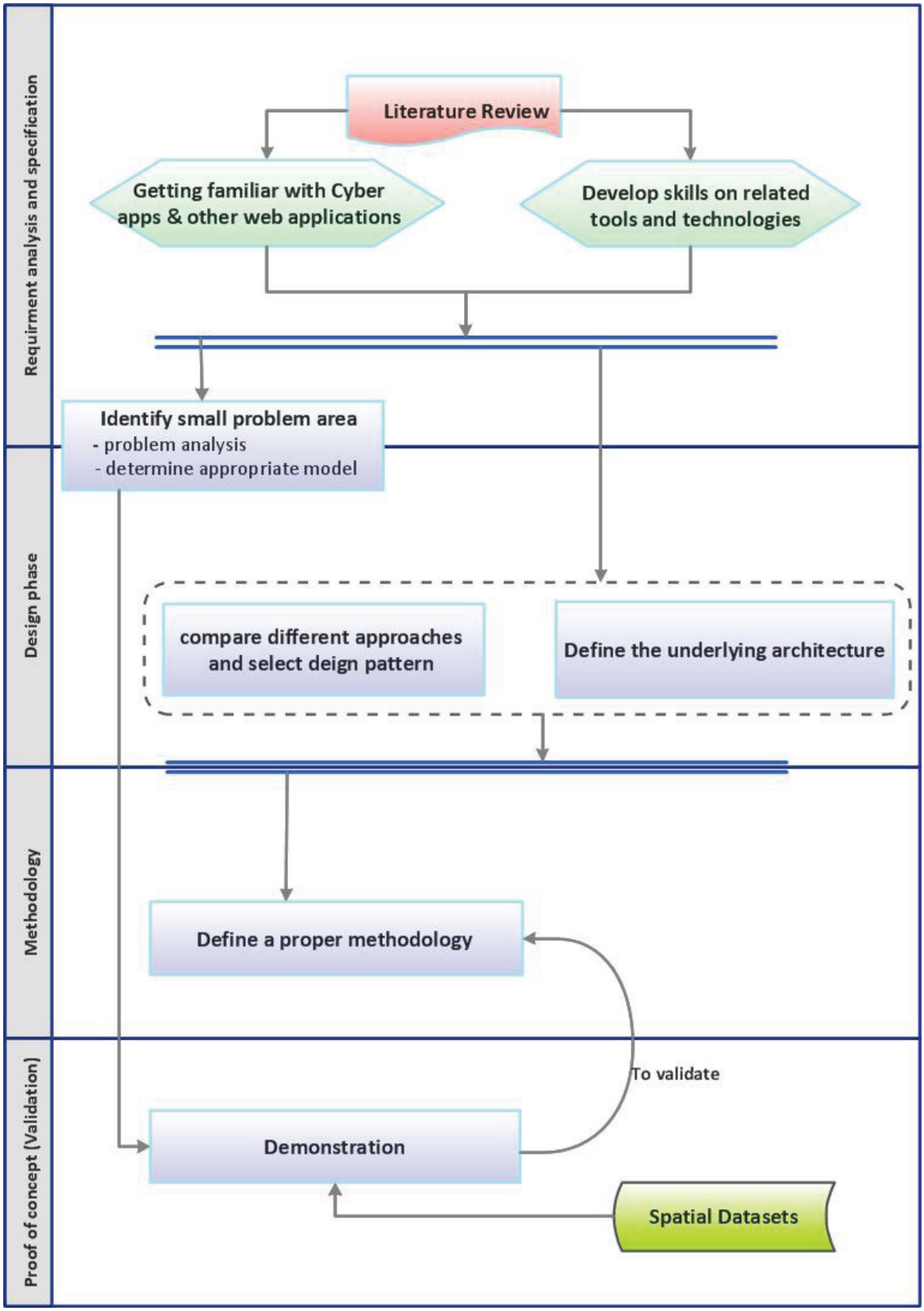


Figure 1.1: Method Adopted

use of a Model View Controller (MVC) design pattern facilitates the process of identifying these components in the design of cyber apps. The MVC design pattern of [50, 31] consists of three elements. The Model element handles the application data, the View element handles the display of the data and the Controller handles user's interaction. The structure of this design pattern can be seen in Figure 1.2. By adapting this design pattern to cyber-application, each of the components have a particular task in one of the above mentioned MVC features. MVC enables designers to separate the user interface from the data presentation and processing functions. The compatibility between technological infrastructure and appropriate partitioning of the applications will reduce the complexity of designing the application structure [31]. There exist alternative approaches to MVC such as Model View Presenter (MVP), Model View View Model (MVVM) and WebForms. However, most of these approaches focus on the presentation layer and making automatic unit testing is an impossible task. Therefore, for our particular case MVC is best option since it enforces separation of concerns, and unit testing is possible. It also structure and organizes application modules, and provides the capability to handle complex application.

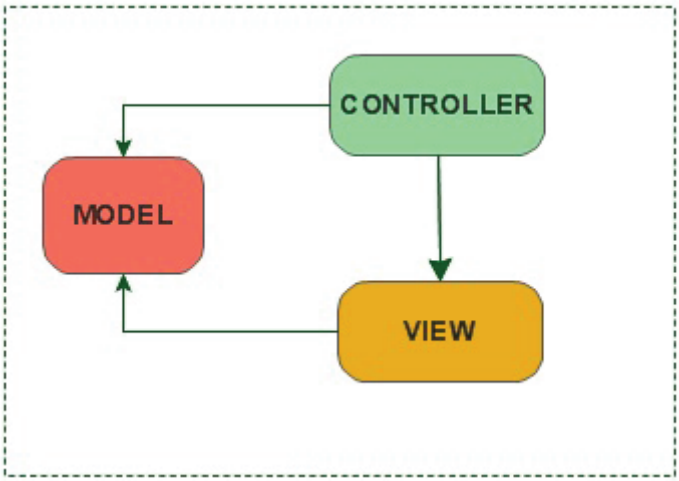


Figure 1.2: MVC Structure

The Interaction between the partitioned components needs to be defined in order to avoid losing control on the sequence. Most Web-based applications are event-driven, they display events as part of their application interface to understand interaction detail between components [30]. However, the architecture of Cyber-applications must support distributed and collaborative components to completely adopt this approach. Therefore, this research project also defines the underlying architecture for cyber-apps to distribute tasks between components.

Additionally, we put into consideration the situation where new components need to be added to the application once it is already in use due to the dynamic nature of the problem and scenario. For this case, the Dynamic Data Driven Application System (DDDAS) approach is proposed as a solution. This approach was first recommended by [8]. DDDAS is a new paradigm in which simulations, measurement data's, and models are dynamically integrated into application to create high performance capability for a wide range of advanced scientific and computational technology research areas [43]. It is also useful for predicting the characteristics of complex application in order to prevent sudden failure of system components.

Based on the approaches proposed in the requirement analysis and design phases, a proper methodology consists of set of steps is defined for the creation of typical cyber-application. During

the final phase of the research project, the suitable problem identified in the requirement analysis phase allowed us to demonstrate some components of the system as a prototype. This prototype is used to validate the correctness and completeness of the methodology; it is not for solving any application problem.

1.4 STRUCTURE OF THESIS

This thesis document is composed of Six chapters including the introductory chapter which describes the research project. The remaining chapters are organized as follows:

Chapter 2 Introduces the preliminary concepts needed to understand the concepts of cyber-application. The main concepts discussed in this chapter are geospatial Cyberinfrastructure, different data sources and cloud computing which greatly participate in the growth of geospatial science.

Chapter 3 This chapter presents the development of web application, their emerging technologies and reviews of some of the proper methodologies. Additionally, few of the modern web applications types are discussed and their similarities with cyber-application is laid out.

Chapter 4 The design method for the development of cyber-application is defined in this chapter. This is the basic chapter of the thesis document where the development perspectives, the design principles and structured set of steps are defined.

Chapter 5 This chapter discuses the identification and specification of the use case together with the result and evaluation of the prototype built based on the design steps. The prototype is developed to validate the methodology defined. Additionally, the underlying architecture for the application is defined.

Chapter 6 The final stage of the research project is summary, conclusion and recommendation. This chapter discusses the strength and reveals the limitations of the design method such that we recommend for further studies.

Chapter 2
Preliminary Concepts

2.1 INTRODUCTION

The main intention of this chapter is to introduce the preliminary concepts needed to understand the contents of the thesis project. We present concepts relevant to the development of Cyber-application and for the process of defining a design method. It is a preparatory step for the remaining chapters.

This chapter defines the concepts of geospatial CyberInfrastructure and its relation with cloud computing environment. We also reviewed some of the data sources that are available for the development of Cyber-applications and Web Geographic Information System (WebGIS). There exists an enormous list of data sources which provide different kinds of services for users and applications in general. However, we only provide brief review of these lists. In Section 2.3, cloud computing has been described in detail and how it is supporting the geospatial science. The final section of this chapter aims to finalize the concepts that are described in relation to the requirements and characteristics of Cyber-applications.

2.2 GEOSPATIAL CYBERINFRASTRUCTURE: WHAT IS IT?

A CyberInfrastructure (CI) is a combination of data resources, hardware, software, network protocols, and computational services that enable distributing computing and information management for building data intensive or other kinds applications [54]. The term CI was first introduced by the United States National Science Foundation (NSF) to describe research environments for the support of data acquisition, storage, integration, and information processing services spread over the Internet.

Geospatial CyberInfrastructure (GCI) is based on CI which is required to store, process, and deliver large amount of sensor data collected worldwide by using geospatial principles and geospatial information to change the way research, development and education are carried out in all aspects of science [54]. It is designed to deal with the distributed geospatial processing and to define the workflow and file transfer process. Geospatial Cyberinfrastructure's requirements are a good match for common capabilities of Clouds.

[54] states that the main components of GCI are community layer, functions, and computing infrastructure. The community layer represents user interaction and virtual organizations which are in specific user domain such as, environment, geography, climate and other science domains to access the available tools and function. The basic components of the GCI framework are the functions and computing infrastructure layers; the framework offers CI functions and geospatial-specific functions and a geospatial middleware to bridge the gap. These functions provide various analytical mechanisms for users. The geospatial information integration layer is supported by geospatial processing, knowledge mining, and observations to integrate geospatial data, knowledge and information. The geospatial information integration and functions layers distinguish GCI from common CIs.

2.2.1 Geospatial Data sources and Services

Geospatial data sources are the most visible part of CI which are used to organize resources, data, directories search tools and application in the web environment. In recent years, the growth of web and the acceptance of the internet support the progression of creating, assembling, and disseminating geographic information for a wide range of activities [23]. Geographic Information (GI) became very important to understand the true behavior of the earth and large number of geospatial resources are available on the World wide Web (WWW) for educational and research purpose. Keeping this in mind, the innovation of recent data sensing technologies plays vital role in accumulating enormous amounts of data in digital format. Data collection can be done by simple forms such as the human eye; and recently, in-situ sensors and satellites help us produce data in real time. The datasets are collected in different locations and obtain information on multiple phenomena's that are changing the surface of the earth. For example, the North American Space Agency (NASA) has more than six petabytes of data stored till today [26]. The collected data are increasingly offered in online data repositories. The initiatives like INSPIRE - the Infrastructure for Spatial Information in the European Community and the Global Earth Observation System of Systems (GEOSS) support this trend. Different standards on the datasets are popularly used in order to be exploited by Wide range of applications. Similarly, the accessibility of these datasets has become effortless since the establishment of Application Program Interface (API)s. Due to this, software components can directly interact with the data and it can also be used to run data portals for retrieval and updating processes on the data sources.

Large amount of spatial data sources can be found in different data repositories which are basically called Geo-portals. Geo-portals provide capabilities to query metadata records for a wide range of applications by linking them directly to online services [32]. This can facilitate the discovery and access to data so that the research environment can analyze these data. The portals can be created by individuals, organization and institutions to deliver geographic content on the web. Some of the geo-portals which have a great impact on the discovery of these spatial data on the web includes: the British geological survey which covers geosciences resources, the World Bank's data catalogue that list of available World Bank datasets, reports, pre-formatted tables, and other resources, the GEOSS which enhance the relevance of Earth observations to global issues, the Global Spatial Data Infrastructure (GSDI) portal which index quiet large number of global geographic data, EU INSPIRE that deals with national government data, US Geospatial One Stop (www.geodata.gov) that serves as a public gateway for improving access to geospatial information, the GeoNames geographical database which covers all countries and is accessible through various web services to provide place names that can downloaded for free, the Dutch national Spatial Data Infrastructure (SDI) (PDOK) that provide nationwide topographic and thematic maps of all kind. However, this is not the complete list rather it is only a small preview of the well known geoportals nowadays.

Since Cyber-application depends greatly on external data resources, some of specific data repositories which can be relevant for our vulnerability analysis use case presented in Chapter 5 are: SRTM (shuttle radar topography mission) which produce digital topographic data and provides elevation data on a near-global scale, TRMM (Tropical Rainfall measuring mission) which provides detailed information of rainfall over the tropical and so on. Therefore, the approach we are following for the development of Cyber-application provide a mechanism to dynamically create access to, and operations on the external datasets. The required data and functions can be gathered from the underlying infrastructure.

2.3 CLOUD COMPUTING

What is cloud computing? Cloud computing refers to distributed computing which relies on sharing of computing resources to handle complex application in a network rather than computing on local servers and personal hardware devices. The other definition of cloud computing is "a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [34]. It is an internet based computing in which servers, storage, and infrastructures are provided as a service to several users over the internet. Figure 2.1 shows the general overview of this technology. In conclusion, it allows users to take advantage from all of the technologies, without the need for deep knowledge.

Geospatial CyberInfrastructure has gained advantage from the recent developments of Cloud Computing technologies such as using Cloud Computing platforms, typical Information Technology (IT) resources such as storages, computing utilities, and databases that are available as services in the cloud environment [54]. For example, MapReduce is a programming model that provides a mechanism and runtime capabilities to undertake GCI tasks on the cloud. "Providing geo-processing functions in Could Computing platforms can bring scalable, on-demand, and cost-effective geo-processing services to geospatial users" [55]. Therefore, cloud computing gains the importance for providing computing infrastructure for users and developers of different applications.

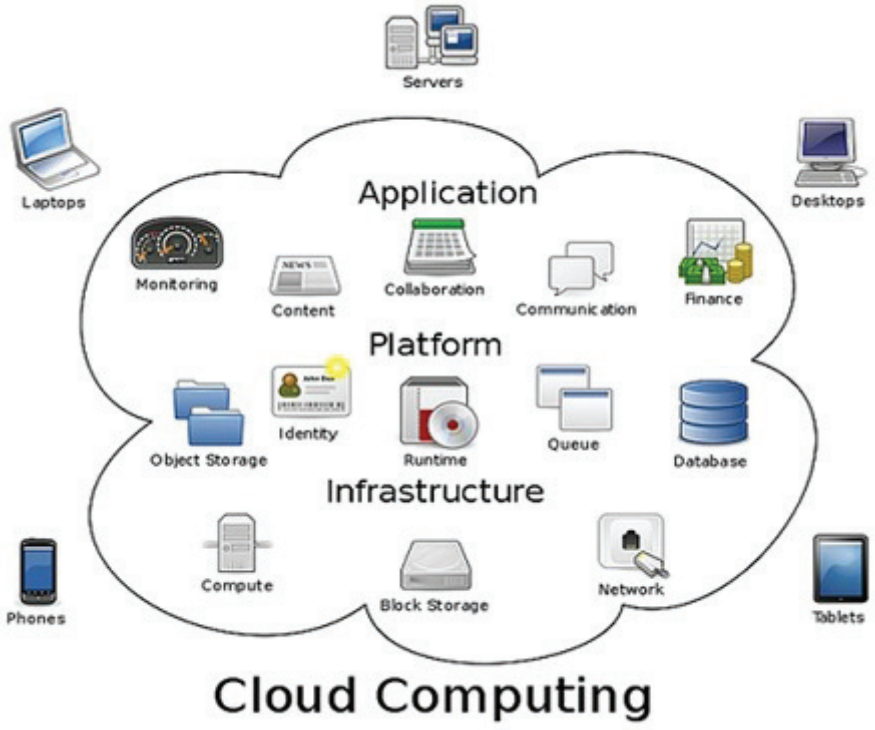


Figure 2.1: General Overview of Cloud computing [24].

2.3.1 How Cloud Computing Works?

Personal computers are no longer undertaking heavy processing since a network of large group of servers running in low cost computer with specialized connection spread data processing across them because of cloud computing technologies. However, the user computer needs to run the cloud computing systems interface software and the cloud network do the rest of the work. In [48], cloud computing system is divided into two sections: the front end and the back end. The front end is the client computer which we interact with and the back end is various computers, servers and data storage's that compose the cloud computing system. Cloud computing mainly depend on virtualization technologies which offer flexibility and platform independence; however they require certain requirements upon systems and applications [33].

Cloud computing utilizes central server which follows some kind of protocols to manage the system and use a software called middleware to allow the set of networked computers to communicate [48]. Many organizations define their own cloud computing technologies because the standard for connecting the computer that runs the software and the system itself is not defined. The technologies that are defined by different companies are based on open standards and open source software's.

2.3.2 Service Models of Cloud computing

There exist several services that are provided by cloud computing based on different models and according to the abstraction level of the capabilities and resources provided. The services can be integrated to provide better solution. Cloud computing adopt concepts from Service-Oriented Architecture (SOA) since the aim of SOA is to deal with requirements of loosely coupled, standards-based, and protocol-independent distributed computing [52]. The basic service models provided by cloud computing are:

- *Infrastructure as a Service (IaaS)*: this layer provides storage and compute resources. This means consumers can directly use the resources such as CPUs, storage, firewalls and so on. The Iaas service model focuses on the enabling technologies and use virtualization techniques to meet resource demands [45]. Virtualization is the creation of a virtual machine that acts like a real computer with an operating system and dividing the resources of one physical computer in order to execute them on virtual machines. Amazon E2C is typical example which mainly offers IaaS.

- *Platform as a Service (PaaS)*: this service model offers development platform such that applications can be built and deployed using programming language and tools. Developers of the application do not need to know what kind of computing resources that the application is using [52]. Google AppEngine is an example of Platform as a Service, it provides scalable environment for developing and deploying Web applications.

- *Software as a Service (SaaS)*: this model is the most known and sometimes called Application Clouds, it provides applications or software using cloud infrastructure. The applications provided as a service can be accessed through a simple web browser by the user; there is no need for client side installation. Salesforce.com adopts SaaS model to offer applications of business productivity. And Google Docs is an example of SaaS which offers office productivity software that can be accessed online. Application of cloud computing is everything starting from basic word processing to customized computer program. Nowadays, it is possible to run any types of application and execute many programs in a cloud computing system.

2.3.3 Types of cloud computing

Cloud computing are classified as public, private, hybrid and community clouds based on model of deployment.

- *Public clouds*: are available for the general public over a network. They reside outside of private firewall. This approach is useful for companies which want to save time and cost for deployment and maintenance purpose. Data security issue is not handled in public clouds and this where the difference lays from other types of cloud deployment models. Amazon AWS, Microsoft and Google are public cloud service providers. They own and operate the infrastructure and offer access only via Internet.

- *Private clouds*: operate operated exclusively for a single organization, thus data and accessibility are only for internal users by interacting directly to the local data center.. These clouds can be managed internally in the organization or by third-party. It is suitable for users focusing on data security and privacy. Implementation of these deployment models can be complicated and time-consuming.

- *Hybrid clouds*: combine aspects of both public and private clouds to deliver solution. This approach is useful for organizations wishing to reduce costs, while preserving data security and privacy [34]. Hybrid clouds employ cloud bursting; which is an application deployment model such that application runs in a private cloud and "bursts" to a public cloud when there is an increase in computing capacity demand.

- *Community clouds*: shared by several organization and support specific community with common concerns (e.g., security, mission, policy, etc) [34]. It can be managed by the organization or third-party and may exist internally or externally.

2.3.4 Cloud computing and the Geospatial science

The geospatial science requires computing infrastructure that can provide access to data, data processing and real time resources. Traditional computing platforms can no longer handle the increased computing demand and it is almost impossible for individual organizations or any end user to own high-performance computing infrastructure. In order to build enhanced applications, we need computing infrastructure that can enable us to access real time data and processing functions on the cloud in order for us to acquire scalable and cost effective geoprocessing services. Therefore, cloud computing platforms using strong storage service can be used store the geospatial data that are required for geoprocessing applications [53].

[53, 55] states that Cloud computing provides solution for geospatial problem by integrating main components of geospatial science which are observation data, phenomena simulation, decision support, parameter extracting and user feedback. It will become the next computing infrastructure for supporting and providing service for geospatial science community over the web. One advantage of using cloud computing is that; cloud platforms that are available in present offer built-in load balancing function to web applications when various request are running.

Public clouds provide adequate resources for both data-intensive and computing-intensive geoprocessing applications. We can mitigate geoprocessing applications into the cloud computing platform by using virtualization techniques, compiling, or building computer source code and following Open Geospatial Consortium (OGC) service standards. In addition, the programming language used for development is usually Java due to its platform-independent feature such that the application can be moved easily from one computer system to another.

Cloud computing are well known for providing service models for consumers. Data as a Service (DaaS) model is specially defined to geospatial sciences to supports data discovery, access, and utilization on the fly to end users regardless of geographic location of provider and consumer [42]. Currently, DaaS is being developed based on several cloud platforms. It is designed to maintain large amount of metadata entries to support data intensive applications.

Generally, there have been several efforts to use cloud computing for geospatial application and Microsoft, Amazon and Environmental System Research Institution Inc. (ESRI) are exploring how to operate geospatial applications on cloud computing environments and are trying to adapt to this new approach in the coming years.

2.4 CONCLUSION

In conclusion, we have available data sources and computing infrastructure that are necessary for the development of any web application. The concept of Cyber-application came from the fact that most of the things we need to solve a problem are actually available somewhere in the web environment and upgraded by third-parties. This includes the data itself, the computing platform, infrastructure specific to geospatial data, and so on. Cyber-applications is different from that of the previous applications in the sense that we do not need to start from the resources we have; rather we directly assume that there exist external sources with certain functionality and the concern is how to be able to exploit from those resources.

These applications also take advantages from the popularity and recent advancement of cloud computing technology. It is possible to use platforms provided by the clouds to build our application or create components on demand and use the computing infrastructure that can enable us to access real time data and processing functions on the cloud. Therefore, everything that has been defined in the previous sections of this chapter assists us to identify the available resources and concepts which can possibly contribute for the construction of Cyber-applications. Understanding the requirements and characteristics of cyber-application is the stepping stone for defining our proper design method.

Chapter 3

Review of Web Application Development, Their Emerging Technologies and Proper Methodologies

3.1 INTRODUCTION

The development process of web application is supported by several technologies and design methods to deliver expected application performance. This chapter presents the development perspectives of these applications, their emerging technologies and a review of proper methodologies that greatly participate in the development process.

3.2 WEB APPLICATION DEVELOPMENT

Web applications are software systems which can be accessed through the internet using a web browser. The development of these applications is a versatile activity which involves technical, organizational, social and artistic issues. The process must cope with a number of requirements such as handling of structured and unstructured data, exploratory access through navigational interfaces, graphical quality, customization of content structure, navigation primitives, and presentation styles, easy development and maintenance, interoperability of systems and data and many more [18].

The structure of web application consists of client-side scripts which run in the client and interact with the user; HTML data displayed to the user and server side scripts which perform main processing at the server end and interact with the database. They perform their task regardless of the operating system and browser running client side. The installation of these applications is rapid and can be deployed wherever at no price without any installation requirements at the user's end. These applications allow users to submit and retrieve data or information over the internet using their favorite web browser to/from a database. They dynamically generate web documents to serve users of the application (e.g., in our use case, informing users about the vulnerability level of a building). These documents are generated in a standard format in order to be supported by all browsers. One of the important aspects of web applications is to support user participation to add value to the application and cooperate with users. Currently, the web is a dominant platform which offers an enormous collection of tools and components to application developers. In general, web applications take the place of desktop applications and became one of the fastest growing technologies with the help of web internet infrastructure, protocol standards, design methods and emerging technologies [27].

3.2.1 Development Perspectives

Before defining the design principles and design steps for the development of Cyber-application, we need to understand different perspectives which are carefully considered in the development process of web applications. Figure 3.1 shows the three main perspectives: the presentation layer, the data, and the business logic. Most web applications reflect on the definition of navigation

structure and the specification of UI on the presentation layer and the storage, the requirements and formalization of data on the data design layer and essentially the specification of core business logic.

Figure 3.1: Development perspectives of web applications

The presentation layer is the part which manages the user interaction with application data. It consists of user interface and navigation structure. The user interface components construct the presentation layer to acquire and render data. Navigation structure includes links and controls which basically enables users to navigate easily through pages. It is important to separate navigation from user interface elements in the design process because it will reduce confusion and hide application complexity.

The data layer holds the data sets which will be manipulated by the application. This layer also includes data access components and service standards to provide functionality for accessing the data hosted in the application or the data exposed by external data sources through web services respectively.

The business logic is responsible for processing function of the application. It implements and encapsulates the behavior and core functionality of the application. Retrieval, transformation, processing and management of the application data are handled in this layer.

Generally, these are the basic design perspectives for the development of most web application but there are also other aspects which are equality important in the design process of Cyber-applications. However, these perspectives are the basis to define the design method for the construction of Cyber-application in Chapter 4.

3.3 TYPES OF WEB APPLICATIONS

There are many examples of web applications, and many of them come out all the time. These applications can be categorized as document centric, interactive, transactional, workflow based, collaborative, portal oriented, semantic, and social [41]. Since the popularity of Google Maps and advent of different APIs, web application developers started to illustrate their application using online maps and use the mechanism to access data directly from the data source for geospatially enabled web applications. Few of the modern web application types are discussed here.

3.3.1 Google docs

Google Docs is sometimes called Web 2.0 application which allows users to collaborate in real-time to generate and edit documents held on Google repository [12]. It is a cloud application that provides the capability of sharing of documents, presentation slides and spreadsheets on the web environment. Changes to documents are saved at a regular interval and updates the repository accordingly. It is a lightweight application and there is no need of configuration at user's computer; we only need a simple web browser to run the application. However, this application does not support off-line editing of documents and the collaborative protocol of Google docs can only be used on textual documents not on graphics or other contents.

3.3.2 CitySourced

CitySourced is a real time web application which was developed in 2009 by CitySource Company. It uses location-based technology like Geographical Positioning System (GPS)-enabled device to connect citizens to different governmental organizations [13]. The data from citizens is fed directly to Geo-databases and become accessible for geospatial analysis. This application can be used to collect and report information varying from damage report to traffic problems. Users with Smart-phone's can take the picture and submit it along with more information to report about specific problem. Then the application sends it up to a cloud-based platform where it can be routed to the responsible organization. User with no Smart-phone can still take part in reporting by using the site.

3.3.3 WebGIS

A WebGIS application is a type of web applications, which generally deals with complex geographic data and makes it accessible on a web environment to users for different purposes. These applications offer numerous services for visualization and analysis of geographic information on the web [28]. In the case of geographic information, WebGIS developers should handle the data separately from the task or functions that can be performed on the data. Therefore, far more complex functionalities are required in WebGIS for visualization and content management other than the common web application tasks which are navigation and composition of tasks. As a matter of fact, this kind of application needs bigger attention on spatial data's with different format which can be obtained from different data sources, hence it is necessary to follow a proper development process to develop such applications.

3.4 THEIR EMERGING TECHNOLOGIES

The support of software tools, methodologies, database, object-oriented programming languages and emerging technologies help the designer to master the development of web application significantly. Some of the recent emerging technologies such as JavaScript framework, .Net technologies, HTML5 and CSS have come together to redefine the way developers build web applications. These technologies provide simple interactivity without plug-ins such as Java applets and Active X controls. There exists other standard coding languages, web design software's, scripting languages, graphic and content management tools which greatly support the development process. To mention few, Web 2.0, MVC architecture, Adobe Flash, Microsoft Silverlight, Java FX, Google web kit and different mobile technologies such as Extensible Hypertext Markup Language (XHTML), Wireless Application Protocol (WAP), Wireless Markup language (WML) and so on. Web technologies seem to change rapidly with the nature of web applications and their development process.

The choice on selecting development platforms heavily depend on the application developer but for the purpose of our research project, we choose to work with Microsoft Visual Studio 2013 the web developer edition, Active Server Pages (ASP).net MVC framework. This platform is where we will write and debug our code. It includes editors, debuggers and all the short cut technologies such as C#, JavaScript, HTML and CSS to develop our application. We also use PostgreSQL DBMS and Internet Information Services (IIS) Express to manage our data and to respond to HTTP request from the browser respectively. IIS is a light weight web server that can be used during development time. It runs behind the scene and help to run and execute our application.

ASP.Net MVC design pattern is selected for building our prototype since we want to separate the responsibilities of our application components in the User Interface (UI). This design pattern does not dictate what type of data access layer we should use. We can either use services, relational database or any form of storage behind the scene. It is extensible, testable and embraces the web. Jquery is also used to add AJAX feature to our application.

3.5 REVIEW OF PROPER METHODOLOGIES FOR WEB APPLICATION DEVELOPMENT

A proper methodology to support web application development is one that clearly defines the design principles of the application, have a good architectural model, and also progress with the rapidly changing nature of the web. Web application development is unique in a way it requires the selection of an appropriate development methodology. Most of web application development methodologies being used currently are extensions of standard software engineering methodologies. They often involve requirement, analysis, design and development phases. Application development methodologies are step generation activities in different phases of a system development life cycle.

In this section of the chapter, we reviewed few of widely used methodologies for the development of web application and WebGIS. This includes Model-Driven approaches (e.g., the Navigational Development Techniques (NDT) methodology), Component-based development, User-centered approach, WISDOM Framework, and the Web engineering methods.

3.5.1 Model-Driven Approach

The approach proposed by [14] for the development of web application is a model-driven approach which is based on the Ubiquitous Web Application (UWA) of [19] design framework, the MVC of [50, 31] architectural pattern and the JavaServer Faces technology. The approach combines the MVC design pattern with a strong proper methodology for the user-centered conceptual design of web applications to improve separation of business logic and data presentation. It also helps to produce web applications which are of standard quality from users perspective and simpler in terms of development process by taking the advantage of model-driven Development and user-centered design.

Some of the concerns and development steps for web application engineering together with their specification can be found in [6]. Basically most of the design methodology representations of web applications are in three development levels: content, navigation and presentation. They try to addresses a number of aspects beginning from structure to behavior of the web application. As it is depicted in Figure 3.2, aspects are orthogonal to all construction levels. It describes how contents are structured as well as the structure of the navigation and presentation represents how pages are organized. Additionally, behavior which can be the interaction of user can be associated with content, navigation and presentation [14].

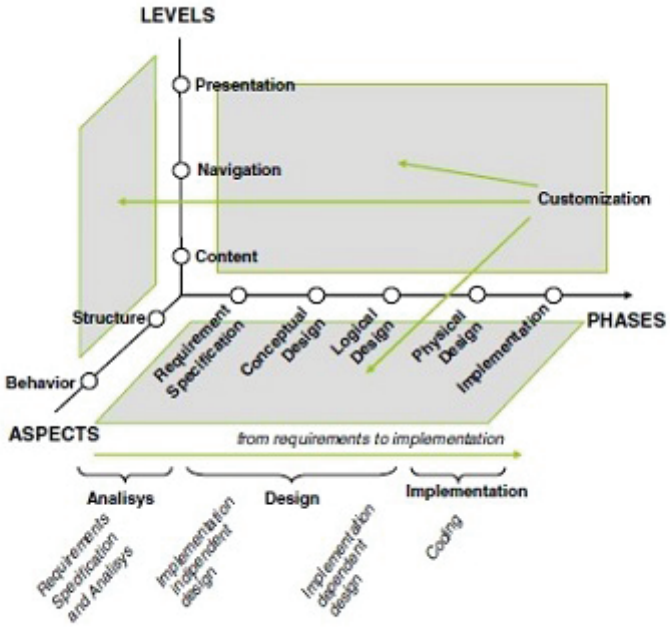


Figure 3.2: A general development framework for web application [14].

The development process starts with requirement specification and continues with a number of design steps until the implementation phase. The steps that are included in the design phase are Conceptual design, which mainly focused on describing the problem domain and what the system should do. This step is works independently of any technological detail. The second step is Logical design, which centers the attention on the operation of the system and hides implementation details. The last step is Physical design; obtain detailed specifications for implementation with the selected platform by adapting the logical model of the application.

To summarize, the proposed method defines an intermediate model based on MVC logical model, UWA conceptual design and implementation phases for the development of web application. It also includes rules to transform the logical model to a platform specific model for Java ServerFace applications. This methodology has been experimented in an example case study.

The NDT Methodology

The procedure of NDT (Navigational Development Techniques) [16] begins with requirements analysis and identifies a set of transformations processes to produce analysis models. Like any of the software development phases, this method includes viability study, requirements treatment, analysis, design, construction, implementation, and maintenance and test phases. The NDT is based on the model-driven paradigm because each of the development phases has meta-models in order to support the pieces defined in the methodology. This methodology defines clearly the requirements engineering and analysis phases of a software development process. Information storage requirement model, functional requirements model, actors model, and interaction requirements model are defined in the requirement engineering phase [44].

NDT also defines two models in the analysis phase; conceptual model which represent the structure of the system and navigational model which presents a view of the conceptual model and shows how to navigate through the information in the system. The relationship between the models is defined in all the phases and it is extended to fit the WebGIS. Geographical data storage requirement model which include the specification of the geographic data is defined in the requirement engineering phase to manage the information and display in the navigational system. Geographical conceptual model is also defined in the analysis phase to represent the structure of the system and describe the information handled by the system within the development of navigation.

This methodology targets the development of WebGIS and it covers the natural characteristics of geographic information system and also the behavior of the web and available Web platform. This methodology combine the NDT approach with some of model driven web methodologies to introduce new process for creating WebGIS applications.

However, there are some drawbacks encountered by the NDT methodology [1]:

- they do not consider a complete organization of requirements which is suitable in most web applications,
- they consider non-functional requirements separately,
- they mainly focus on design aspects of the intended Web system without considering the web requirements and last but not least,
- they do not perform the analysis of the user's needs.

3.5.2 Component Based Development

Component-Based Development (CBD) approach proposed by [30] is a software development approach where the complete phases of the development process is focused on concept of component life cycle. This methodology is specifically designed for high-level design component-based Web applications, since most methodologies lack the support for componentization. A component in Web-based applications is a software program that has been pre-compiled to present certain functionality. Therefore, web application development is definitely component-oriented and need appropriate object oriented process [2]. The CBD process has five phases: requirements, analysis, design, implementation and testing. The methodology introduces page and component classification then specifies procedures for requirements analysis and design. An example is also used to demonstrate various stages of the analysis and design process.

The two key parts of the methodology are the components requirements analysis and component specifications. The methodology starts with identifying application function both in terms of high level abstraction as well as lower levels in the requirement analysis phase. The three steps necessary for component specifications phase are: rendering specification, integration specification, and interface specification. The integration specification identifies relationships between components and the interface specification finalizes the interfaces of the components. "First two specifications are needed for the developer to build, customize, and compose components, while the third one is used to locate and call components in a distributed application environment" [30].

Since the size and complexity of web applications is increasing through time, component-based application development is an essential and effective approach. The proposed methodology introduces comprehensive framework to facilitate such development process. However, it also contains few limitations such as it did not clearly define the interaction between different component types and this is particularly important for ease exchange of data. Most spatial problems require great amount of data exchange in the application, hence this should be defined when modeling the interaction between the different kinds of components.

3.5.3 User Centered Approach

The paper presented by [21] is a design method along with process models for the development of web applications and it includes the aspect of user modeling and customization to Web engineering. This approach is user centered since it considers individual needs and preferences by adapting the information provided in the application. Designer can adjust the presentation of the application content according to types of user since the user profiles are captured and user categories are defined.

The proposed approach considers three different levels of abstraction: Conceptual, Logical and Physical. The management of user's data, content design, navigational design and interface design are separated in the development process by following SoC principle of [38].

In the conceptual level, a conceptual navigation schema is created and transformed in to a set of logical pages and links to enable the implementation of the application in the chosen environment. The proposed method is defined as a set of data models and process models. The process models symbolize design guidelines including a number of different steps such that the developer can follow and create the intended application. The data models provide the concepts which allow for describing the application structure.

The approach considers navigation session by the user as an achievement, so the user and user goal are modeled. This conceptual modeling step is to characterize the user goals in terms of navigational elements. The structured way of navigating is defined in a Navigational structure Navigational guide. This process is incremental and recursive and it also includes two steps. User Goal Reduction process which is based on the defined User Profiles and Deriving Navigation Structures and Navigational Guidelines.

3.5.4 The WISDOM Framework

WISDOM (Web Based Information System Development with a comprehensive Methodology) [4] is a structured methodology that can be adapted to a particular web-based information system application development. This technique is mainly focused on managing application data in the web environment since the web has become the standard interface or platform for publishing and exchanging data. In the context of web application, information import and information export are the two main concerns in WISDOM framework.

The two main activities identified in the methodology related to the information export concern are Web site design and workflow design. Associating web site with its content for querying, development and maintenance and separating the information content from navigation and presentation are the two basic goals of web site design. Therefore the methodology distinguishes three main sub-activities: the data design, the hypertext design, and the presentation design. These activities are independent of each other but highly coordinated. The workflow design is relevant for all the activities that are involved in building the application. This phase involves three steps: process analysis, workflow specification and workflow implementation.

On the other hand, information import in WISDOM involves series of steps to identify heterogeneous information sources and come up with unified information representation. The major activities in this phase are extraction, integration and warehousing. Identification of information sources, selection of target data model and wrapper design are three sub activities that are required in extraction process so that relevant information can be extracted.

In general, the WISDOM framework support the development of web-based information system application in which information sources are analyzed, extracted, processed and integrated to give some kind of services on the web.

3.5.5 Web Engineering

Web Engineering is an engineering principle which is the most traditional of Web-specific methodologies that proposes efficient development framework for creating web based applications and systems [41]. It specifically focuses on the methodologies, techniques and tools which help in building the foundation of web application development and support their design, development, and evaluation. Web engineering deals with all aspects of Web-based application development, it starts from conceptualization to design, implementation, and testing life-cycles.

3.6 CONCLUSION

This chapter explored the various aspects of web application development. This includes the development perspectives, the types of web applications and common tools and technologies that greatly participate in the development process. Web application types that are described in Section 3.3, share few similarities with Cyber-applications as pointed out in Table 3.1. All of these web application types require a robust design method to guide the development process.

Table 3.1 Similarities with Cyber-applications

	Shared Similarities with Cyber-applications
Google docs	- Allows users to collaborate in real-time
CitySourced	- Uses location-based technologies like GPS-enabled devices
	- Can be used to collect and report information based on problem domain
WebGIS	- Handles the data separately from the task and function that can be performed on the data

In Section 3.5, we reviewed in detail some of the proper methodologies that are used to develop web application. All methodologies that are reviewed possess qualities that have significant values for Web development. Therefore, the design method for the development of cyber-applications overlay multiple concepts from the techniques and approaches described above. However, our approach is different since it has the capability to dynamically incorporate new data and components into the application's execution engine at runtime. This is where it basically departs from the previously proposed proper methodologies. It is based on the MVC design pattern and the DDDAS method, such that it fits the purpose of cyber-application. This methodology will be discussed in the next chapter of this thesis document.

Chapter 4

Our Methodology

4.1 INTRODUCTION

This chapter presents a design method for the development of Cyber-applications. The proposed approach is used to guide developers who wish to gain high degree of control over implementation. The chapter is organized as follows. Section 4.2 provides structured set of steps for the development process which includes Problem definition, Data structure design, Data source identification, Workflow specification, Application partitioning, Interface specification and Realization. Their detailed explanation is presented. Some of these steps also include sub-steps to elaborate the design method and meet the requirements of the application. Section 4.3 summarizes the basic concepts of this chapter.

Additionally, after undertaking an analysis of the existing methodologies summarized in Section 3.5, set of elements are selected and incorporated into our methodology based on their strong aspect pointed out in Table 4.1. We overlaid the concepts of separation of business logic with the data presentation and component identification in the application partitioning step. The management of data, identification of heterogeneous information source and information representation in the data structure design and data source identification phases. Also, the description of the problem and the identification of application functions in the problem definition and workflow specification steps respectively.

Table 4.1 Basic Concepts Overlaid in Our Methodology

	Selected Features
Model-Driven approaches (e.g NDT)	- Separation of business logic with data presentation - Requirement of problem definition at early stage - Management of geographic data in the geographical data storage requirement model.
Component-based development	- Identification of application function both in terms of high level abstraction as well as lower levels. - Component identification and interaction
User centered approach	- Separation user's data, content design, navigational design and interface using SoC principle.
WISDOM Framework	- Identification of heterogeneous information sources and come up with unified information representation and selection process of data model for this information.

4.2 DESIGN STEPS

Figure 4.1 shows the design perspective for the development of Cyber-application. The inner layer is the same for any arbitrary web application development. The two outer layers describe the steps relevant for Cyber-applications. The development steps are:

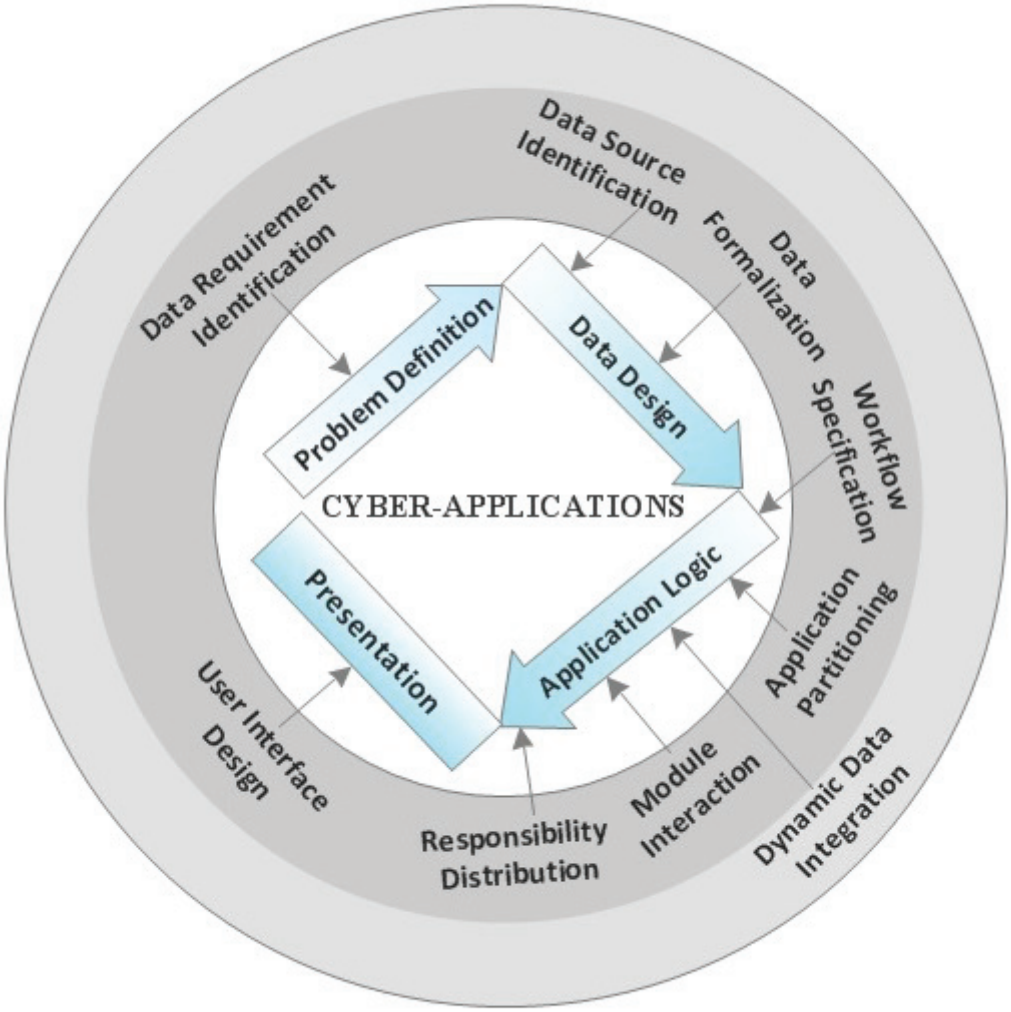


Figure 4.1: Cyber-Application Design Steps

Step 1 **Problem Definition.** This step focuses on creating the textual description of the problem domain. This means we need to identify and clearly describe the problem and how we are going to provide solution for it. This step also includes the identification of data requirements.

Step 2 **Data Structure Design.** In this step, we should create a formal representation of the data elements which will be circulated inside the application based on the data requirements. We need to define a pre-defined data structure for the datasets in a way they can be incorporated into our application.

Step 3 **Data source identification.** This step includes the identification of suitable data sources from available infrastructures and mapping them into the data structure that is created. We have to decide which datasets are fit to use on the basis of the area of interest and provide mechanisms to use them in the application. This step includes sub-steps: data accessing and data fitting process.

Step 4 **Workflow specification.** This stage of the design process focuses on identifying the workflow, identifying tasks and their ordered sequence of action to take place, and the events that are behind the instantiation of those tasks to achieve the goal of the application.

Step 5 **Application partitioning.** This step is about the definition of executable modules. After the workflow is specified and different elements of the application are identified, it is necessary to partition the application into different pieces called application modules to distribute tasks and achieve the separation of concern principle.

Step 6 **Interface specification.** Once different application modules are identified, we need to define the communication pattern to avoid losing control on sequences of interaction. Therefore, in this step we build the interaction interface between the partitioned modules to be able to realize the communication between them.

Step 7 **Realization.** The final step of our design process is the realization of the application. Since we have all the necessary components of the application at this stage, the reasonable step here is to instantiate the application.

4.2.1 Problem Definition (Step 1)

At the start of the implementation process, the developer should clearly define the problem through simple to understand descriptions. This includes identifying and describing the problem domain along with its data requirements. It is also necessary to describe the overall application behavior in relation to the problem domain. This means the developer should point out what the application is aiming to solve and identify what kind of functionality is offered by the application. At this stage of the design step, it is not necessary to use any technical terms.

Data requirement identification

Defining the problem enable us to identify data requirements that are relevant to the application. The identification of data requirement means documenting the data which is necessary to describe each item in the definition of the problem domain. Data describes real world problems and the procedure of identification is an iterative process. Data can be identified by different means. For example, from analysis of existing data structures, document preparation, interviews, and so on [37]. The outcome of this step is a documented set of data which describes the overall application structure and represents requirements of the problem domain. Section 5.4.1 presents an example of this step for the use case which is presented in Chapter 5. Therefore the next step of the design process is to organize and structure the contents so that the information that is circulated in the application can be represented formally.

4.2.2 Data Structure Design (Step 2)

In this step, we make a formal representation of the data required by the application. This step corresponds to the categorization, organization and modeling of the core data which was previously identified during the first step into a conceptual schema defined using Computation-Independent Model (CIM). We have to create a conceptual schema of the problem by documenting the basic entities of the application and their simple relationships. After creating the conceptual schema for the application, we need to map it into a logical schema using Platform-Independent Model (PIM) (For example, into a definition of table for relational Database Management System (DBMS)) but independent of a particular DBMS and other physical consideration. In the logical data schema, we should specify more detailed entities and their specific attributes with their relationships without implementation detail. This process is important since it allows the developer to build a different module if another technology comes around. Therefore, the PIM should be independent of all physical considerations. The next step in the data modeling process is the physical level; which is defined using Platform-Specific Model (PSM). It is described under the next sub-step.

Data Formalization

The PSM is basically adding physical definition to the logical model. It describes the database structure and specifies how the functionality identified in a PIM is realized for a particular platform. The data should be structured such that it can easily be maintained and be accessible by the application. In the case of Cyber-applications, the data should be formalized in a way it allows the integration of new dataset coming from different data sources. Therefore, we need to create a pre-defined structure (For example, as Model elements) to represent the information base of the application using a platform of choice. This means we need to decide what technology to use and implement the PSM into the application.

The data schema that is created has to be represented by a set of model elements in the application. The responsibility of representing Cyber-applications data falls between Model-type

application modules when following the MVC design pattern. Models are created to instantiate a class in the data schema. These Models capture the semantics of the problem domain, which the application is intended to solve. These model elements will be instantiated as classes of language and they represent the actual data which come from a database in the background by the means data access layer, or from external data sources which can be accessed by a service.

An example could be, a Model called "building" to manage the data about buildings. Listing 4.1, show that in each of building object, we are interested in the Materials, Floors, Occupancy, People, Area and Value. We have assigned a data type and a display name for the elements in the model. This model and the properties of its elements are the same as our respective data schema in our data structure such that it represents the data structure properly.

Listing 4.1: Model element for building information

```
//Class for building information
namespace Vulnerability.Models
{
        public class Building
        {
            [Display(Name = "Building ID")]
            public double Build_ID { get; set; }

            [DataType(DataType.MultilineText)]
            public string Materials { get; set; }

            public double Floors { get; set; }
            public string Occupancy { get; set; }
            public double People { get; set; }
            public double Area { get; set; }
            public double Value { get; set; }

        }
}
```

In Listing 4.2, a Model has been created to hold information about landslide vulnerability for all the buildings. The same thing works here; in our Landslide model we are interested in the Landslide_runout, Landslide_distance_max, Landslide_distance_beyond, and Building_vuln_Value. The Building ID is also required since it is a foreign key in our respective table in the back-end.

Listing 4.2: Model for landslide hazard

```
//Landslide hazard Model
namespace VulnerabilityAnalysis.Models
{
    public class LandslideModels
    {
        [Display(Name = "Building ID")]
        public double Build_ID { get; set; }

        public double Landslide_runout { get; set; }
        public double Landslide_distance_max { get; set; }
        public double Landslide_distance_beyond { get; set; }
        public double Building_Vuln_Value { get; set; }
    }
}
```

All of the models should be correct since most of the View and controller elements depend on them to manipulate the datasets. When these model elements are loaded, all components of the application can effortlessly access and manipulate the data.

4.2.3 Data Source Identification (Step 3)

This step includes the identification of data sources that fits the pre-defined data structure from available infrastructures based on the problem domain. We need to identify datasets and map their data definition into the data structure of our application. The availability of these data sources is vast. Therefore, the identification should be based on fit for purpose criteria such that appropriate datasets that fit the model of the application can be identified and accessed for the specified geographic area covered by the application [39]. This means, there will be several data sources that can be associated for the application since the application can be used in different geographical areas. One of the functionality that is offered by Cyber-application is the mechanism to access and operate on external data sources. Therefore, our design method provides different approaches to use identified datasets for the purpose of the application as described in the next sub-step.

Data accessing and fitting processes

There could be three situations to use the datasets for the application:

1. If there exists an already available dataset that fits the model,
2. If the identified dataset fits the model and,
3. If the identified dataset needs to be manipulated in order to fit the model.

Identified datasets can be accessed and used in our Cyber-application by three different approaches. For the first situation, we only need to introduce the data to the application. The datasets can be straight uploaded to the models of the application. For the second situation, our design method uses Management of external Data (SQL/MED) approach which provides foreign data wrapper to wrap the data and use it in the application. The third situation is if the identified dataset does not fit the model and needs manipulation. For this, we need another approach which is called the DDDAS method.

SQL/MED

Cyber-app implements the concepts of SQL/MED standard specification [35] to access and manipulate identified datasets which fit the pre-defined data structure of the application. SQL/MED stands for the Management of External Data. It provides a fast and easy technique to access external data sources. This specification is divided into two parts: the *foreign data wrappers* which is used to access external data and the *datalinks* which is used to treat or manage external linked files. For the purpose of our design method, we can use the foreign data wrapper to access datasets from different data sources. The data's can be stored in XML, CSV, SHP, or HTML file formats.

The foreign data wrapper provide access to third party data sources and translates them into a standardized format [47]. It also gives information about the data sources and delivers the data to the application. The wrapper uses regular Structured Query Language (SQL) to manipulate the data stored in SQL-based or NoSQL DBMSs. It does not need any other form of implementation to communicate with web services. The foreign data wrapper architecture is depicted in Figure 4.2 with the steps to install, connect and create the foreign table.

To implement the foreign data wrapper concept, we need to install a library that handles the communication with the external service. Once the library is installed, the extension has to be loaded and we need to establish a link with the foreign server which also provides a pointer to the web

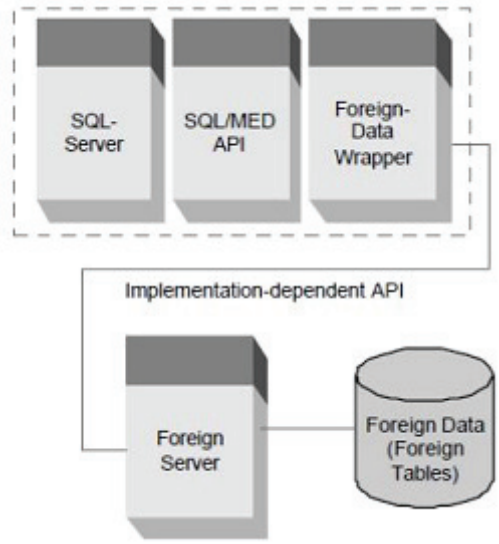


Figure 4.2: Foreign Table [35].

service. The next procedure is to create the foreign table and access the data via a simple SQL queries and the resulting data's can then be transformed/interpreted such that they can be compatible with the pre-defined data structure (or Models elements) of the application. Listing 4.3 shows a simple example on how we can exploit external data sources using the concepts of foreign data wrapper.

Listing 4.3: Creating connection to web service

```
1 create extension www_fdw;
2
3 create server google_server foreign data wrapper www_fdw
4 options (uri 'https://ajax.googleapis.com/search/web?v=1.0');
```

In Listing 4.3, we have installed the extension foreign data wrapper (www_fdw) from the library and we created the foreign server with an option setting a pointer to the appropriate web service. This will create a direct connection to the requested third party data stream.

Listing 4.4: Creating a foreign table

```
1 create foreign table google_table (
2   title text,
3   link text,
4   snippet text,
5   q text
6 ) server google_server;
```

After we set up the connection, we create a foreign table in Listing 4.4 which brings the rows from the external data source with the response fields.

Listing 4.5: Executing a query in the external web service

```
1 select * from google_table where q='cat dog' limit 1;
2
3 Result:
4 title              | snippet               | link
5 -------------------+-----------------------+-----------------
6 CatDog - Wikipedia | CatDog isan American..| http://en.wikipedia..
```

Finally, we use a regular SQL query to access the external web service or data source.

Generally, there will be different data sources if we use the application in different geographical areas, but the application data definitions remain the same. The identified dataset might easily fit into the pre-defined data structure created for the purpose of the application. Therefore, we can adopt the above wrapper approach [1] concept to access the datasets. However, this might not always be the case because of the dynamic requirements of the application and the heterogeneity of the data sources. In this case, we use the DDDAS approach.

DDDAS

Another approach to access the data is the DDDAS method by Darema [8]. This method is used for the cases where the identified dataset does not automatically fit into the data structure of the application because of the heterogeneity and we do not know what the original data is going to be. There are also conditions where new data might not fit because the application requirement can change dynamically at execution time due to the nature of the problem, and require the dynamic integration of new data into their execution engine. The DDDAS method provide the capability for dynamic adaptation of new data into the application considering the heterogeneity of computational models and data sources that are available on the web environment.

The DDDAS method is used to deal with the problem of large-scale dynamic data integration, where the data sources are unknown at design time, and are from heterogeneous sources [56]. Data from different data sources or real-time data-acquisition system can be incorporated to the application to enhance prediction capability using the DDDAS method. This method requires the integration of more data and computational modules into the application to be able to function as required. DDDAS is usually used where data is fed into an executing application either as the data is collected or from a data source. The data is integrated on the fly and the DDDAS method enables the identified dataset to be understood by the models of the application for different purpose.

How to implement the DDDAS approach in our design method?

How do we use it here? In cases where the data schema of the data source does not fit with the schema of the destination data store, the dataset needs to be parsed and formatted to an intermediate data repository structure based on its metadata. These intermediate data structure can also be defined based on the application's predefined data structure. The next step will be to map the intermediate structure to the target data schema of the application. There must also be an "integration manger" that will impose an integration strategy between the intermediate and target structure. This includes ignoring an attribute of the intermediate structure, merging with the version that already exists in target data schema, initializing a new attribute if it is not available in the target data schema and etc.[15].

The STEPS in a sample case

We have two elements:

- The data structure that has already been pre-defined for the application to be able to run; and
- The dataset that is available as input which comes with its own metadata; and unknown to the developer. This data may or may not match the pre-defined data structure of the application.

[1] https://github.com/cyga/www_fdw/wiki/Examples

Step 3.1 To retrieve that metadata and create an intermediate data structure of the datasets (e.g., as tables)

Step 3.2 This table which is created in the first step will be mapped into our application data schema. (Mapping of the existing columns from external tables into the columns that exist into the application). Now we have the mapping between the attribute we have in the original arbitrary datasets and the attribute of the data structure.

Step 3.3 This step is to populate the database so the tables of the application with the original data to be used by the application.

Following the steps from this method, we can be able to map arbitrary datasets to the predefined data structure of our application. This means the description of the identified dataset will be matched with the description of the targeted data schema of the application. Generally, the DDDAS approach provides valuable mechanisms which allow us to use heterogeneous data sources. There are also many other cases where the method comes up with a solution.

4.2.4 Workflow Specification (Step 4)

This stage of the design process, deals with the identification of different elements of the application which includes functions, functional steps and events that trigger those steps. Workflow specification is one of the important steps in application design. To specify the workflow of the application, we need to identify a series of tasks and events that occur in the application in their correct order. In workflows, the sequence of possible actions is predetermined. This means tasks need to be organized in the application structure according to their execution order to achieve the correct sequence of activities in the design process. Such workflows can be described using formal or informal diagramming techniques. Tasks can be complicated and complex in Cyber-applications because of the dynamic behavior of the application. Therefore, before starting the implementation phase, it is important to visually represent how the tasks are being completed in the design process.

For example, refer to the Figure 4.3 to understand the above concept in a sample workflow specification diagram. We explain the concept used in determining a landslide vulnerability value of a building. Note: this workflow is not a complete workflow specification for the use case. It is only for the landslide vulnerability.

To calculate the vulnerability value, we first need the building dataset and landslide runout dataset as input data. The landslide runout dataset contain the maximum runout for each landslide areas. Then we need to join the two datasets and extract the minimum landslide runout value per building from the attribute table of the building. The next step is to calculate the maximum runout distance for the landslide to which a building is exposed. Then we subtract the minimum landslide runout from the maximum runout distance to calculate how far the landslide moves further than the house. If this value is zero, it is undefined else the building is exposed. The final step is to count the number of buildings which are located in the runout zones of landslides. If the distance of the landslide beyond the building is more than 17 meter, the building is considered to be destroyed; otherwise the vulnerability value increases linearly [5]. Therefore, when a user selects a building and calculates for landslide vulnerability, the computed value for the corresponding building will be displayed to the user along with additional attributes.

Workflow expresses the behavior of the application. Each of the steps in the workflow needs to have input description, rules, algorithms and output description. Therefore, tasks should be identified as well as events that occur though the workflow steps. Based on the tasks and lists

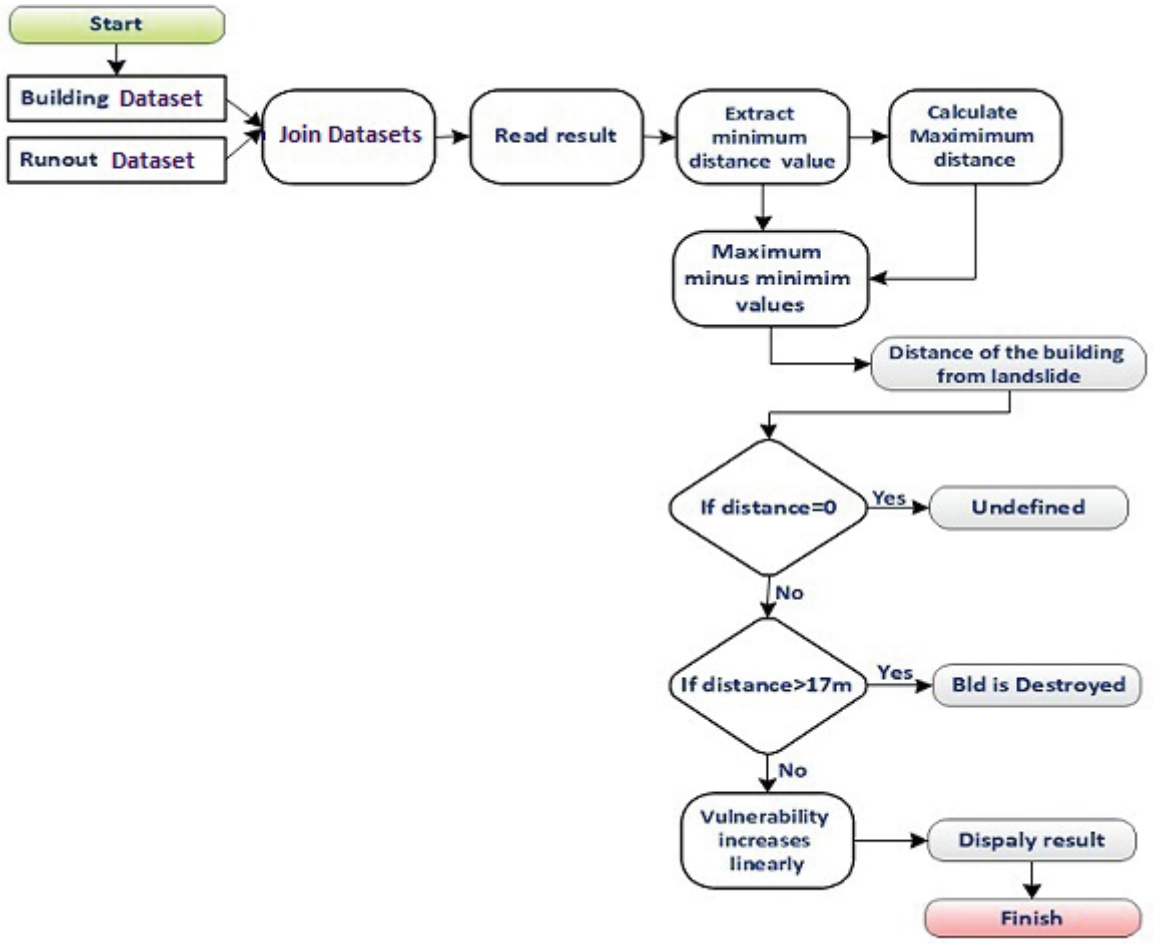


Figure 4.3: Workflow for determining landslide vulnerability value

events identified in each step, we can be able to partition the workflow of the application in to different parts.

4.2.5 Application Partitioning (Step 5)

Once the correct application workflow is defined and list of events are identified, it is a simple procedure to partition the internal structure of Cyber-application into application modules such that tasks can be assigned. Application modules are elements of the application which contains application code for manipulation of data and provides access to the code through well-specified available services. We are using application partitioning principle because it allows us to spread the load between different parts of the application and achieve separation of concern principle. We can build highly scalable and flexible systems by partitioning the application based on lists of events occur and by separating the presentation from the data and core application logic parts for ease of implementation purpose.

Based on the workflow specification, the first step of partitioning cyber-applications is to differentiate the data's from the presentation and the application logic. These major concerns can be identified and separated in the workflow. Since we are using the MVC design pattern for identification of application modules, each of these concerns will be represented by the Model, the View and the Controller elements respectively. An application module can be a Model, a View

or a Controller element. After we partition the application structure into three major parts, once again each of these elements will be partitioned into pieces to represent separate tasks and cover the whole application structure. This partitioning process must exhibit major phenomena and events in the application workflow. Given that the partitioned application modules capture the behavior of the application, they must also encapsulate the application workflow specified in the previous step.

Partitioning a Cyber-application in a simple use case:

Suppose we want to use as an example the vulnerability analysis for partitioning the application into a set of functional units (named application modules) that can be composed into a larger application. In this application, the important elements are the elements at risk (e.g., building, population information) the hazard and etc. We have available datasets for the region of Nocera, Italy. Based on the general application workflow, we partitioned the application into Model, View and Controller elements. Then each of these elements are partitioned into different pieces. First the model is partitioned into five model classes which are "building", "landslide", "flooding20", "flooding100", and "mudflow" model classes to represent the data concern in the application structure. These model classes are domain specific representation of the data models. Therefore, they are basically partitioned based on the data models of the application and data aspects identified in the workflow. The View is partitioned into four views which are three main views and one floating view. Namely: "map view", "hazard computation view", "grid view" and "results view". This partitioning is done based on the user's interaction identified in the workflow of the application. View elements coordinate and arrange the user interaction with the application, and present the information when required. These elements are also tightly knotted with model and the controller elements. The controller is also partitioned into five elements which include "mapping controller", "building list controller", "hazard selection controller", "DAL" (Data Access Layer) and "computation controller". These are the core application modules which bind the application together. The partitioning of the controller is based on the sequence of actions to take place and the events that instantiate those tasks.

The cooperation among these elements is indirect since they do not know the existence of one another. Hence, the MVC design pattern facilitate the structuring and organization of these independent application modules based on their responsibility in the application; and it is also possible to replace or include new one's depending on the application requirement [39]. Due to this, the application modules are of two types; namely: Permanent application modules and volatile application modules.

Permanent Application Modules: These are the backbone for all Cyber-applications. They carry the logic of the application, provide mechanism to access external data sources and work as a link between the permanent and volatile application modules. Typical examples for this case are the "DAL" and "Data Feed Service (DFS)" modules.

The DAL application module is responsible to store and manipulate states, typically in a data store. In other words, any exchange of information to communicate with the reader is handled by this "DAL" application module. The DFS is responsible to retrieve the data which is suitable for the pre-defined data structure of the application from external services. This application module adopt the concepts of SQL/MED or DDDAS approach depending on the situation of the data. Both of these modules are always available for Cyber-application.

Volatile Application Modules: These application modules are one's that are dynamically incorporated to cyber-application at run time when required. They can be computational models or

third-party data sources.

Generally, the partitioning is done based on the identified tasks in the workflow. Therefore, responsibility is distributed between the application modules to realize the final structure of the application. Each of the tasks involved in the workflow of the application need to be assigned to the appropriate application module. Responsibility distribution is the part where the developer specifies a clear separation on "who does what and where". For example, every Cyber-application requires permanent application module to handle data transfer between the application and third-party data sources such as the DFS and DAL application modules. Therefore, the designer is responsible for assigning a permanent application module which is responsible for retrieving the datasets and making them available to be used by the application adopting different mechanisms. Cyber-applications also require another application module to depict the core logic of the application. However, the number of application modules to assign for the application depends on the tasks identified in the workflow and the purpose of the application. In general, responsibility distribution involves identifying the appropriate application modules and answering the question "what happens where?" or "who takes care of what?".

4.2.6 Interface Specification (Step 6)

This step relates to the specification of interfaces between the partitioned application modules. The interface specification is used to locate and call appropriate application modules participating in one of the internal workflows that realize the purpose of the application. It specifies the relationship between application modules and additionally provides information on who initiates the call based on events. In order to specify interfaces, the interaction structure between the application modules should be defined. Interface specification can be in terms of messages containing methods which include information about the parameters, events, callers, callees and return values/types.

The interface specification of partitioned application modules should show clearly the defined sets of interaction between them in order to have smooth communication and exchange processed outputs. The interaction between the applications modules follows **Event-based paradigm**. List of possible events occurred in the workflow specification step instantiates the interaction between application modules. An event is any certain incidence that has importance in the communication between the application modules. Events could be user generated like user gestures such as mouse clicks and keystrokes or system generated like the arrival of a message and the completion of task [39]. Defining the interaction is useful to compose the application modules into a complete application at the later stage of the development process. Event-based paradigm is an essential design principle which helps us to avoid losing control on sequences of interaction. Therefore, interface specification provides an implementation detail of an application module in terms of its interactions with other modules and processes in order to render a required result.

In Figure 4.4, the main task of the Controller is to listen to events and respond accordingly. It is mostly concerned with the retrieval, manipulation and management of the application data stored in the application DBMS or data residing outside. A controller could have different types of interfaces. For example, an event manager interface which has methods to define the set up of an event identifier that we want our controller to listen on and should simply return Model and View elements to render results; or another interface which has methods to handle multiple HTTP requests throughout the life-cycle of the application using GET, POST, PUT, DELETE methods.

Controllers can receive events forwarded by the view and then decides what they mean and what to do with them. For example, controllers communicate with the View to determine if a building is being selected by the user, and they call methods on the building model to handle the

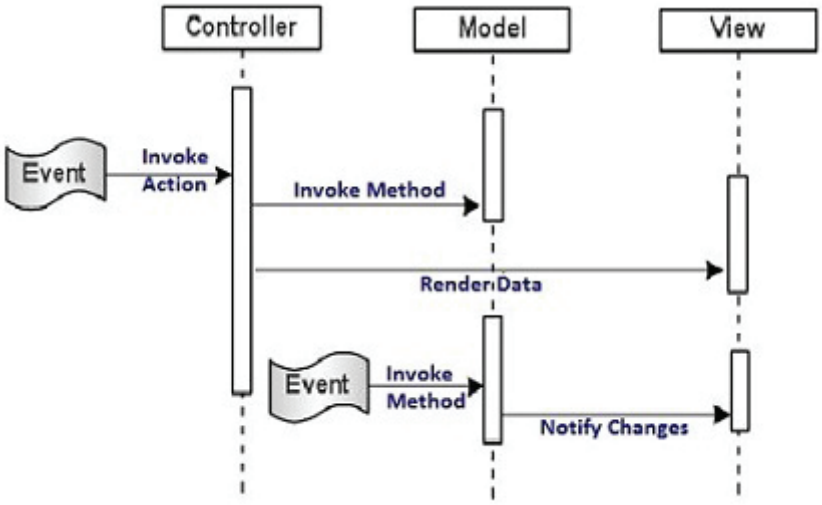


Figure 4.4: Simple Event-based Interaction in MVC design pattern.

next request. An event can also invoke action in the appropriate Model if needed. For example, when new data is integrated into the application; any change on the Model state will result in View updating its presentation because it is registered for change notifications. This is an example of push-based mode which will be explained briefly later in this section.

Event-based paradigm is sometimes called publish/subscribe style which support a flexible and effective interaction between software components. The development of a publish/subscribe mechanism for information store in MVC chains event driven linked data concept. For example, users can send request to compute landslide vulnerability value for a single building. The participating application modules execute this task and provide response. It is useful to separates event-processing logic from the rest of a program's code. Most Web-based applications are event-driven, they preserves both the structure and the semantics of the application modules and they usually display events as part of their application interface to understand interaction detail between components [30].

There are two modes in event-based interaction: Pull-based and Push-based [25]. In pull-based mode, at regular interval data can be read and explored on-the-fly. This mode is basically used for monitoring scenarios (e.g., read the sensor status for every thirty minute or get rainfall data every hour). In push-based mode, a function produces an event when some kinds of conditions are met and the View register with the model for change notification (e.g., send SMS notification for registered users if certain thresholds are reached or exceeded). Each of the participating application modules perceives the notification of an event as an external effect which determines a change in their internal state. Both push and pull modes are valid and can be used for interface specification. The examples above explain what they both mean and the situations they can be used. However, the developer should decide which mode while specifying the interaction interface depending on his/her application requirement.

The main intention of this step is to show the developer how to define the interaction between the application modules in order to avoid losing control on sequence. Each element in the MVC design pattern corresponds with each other in very specific ways. Communication is initiated by a sequence of events that are triggered by a user interacting with the application. There are communication patterns which are allowed and restricted. The usual way where the MVC elements communicate is when users interact with a view, views interact with the controller and controllers

may interact with views. Controllers also communicate with other controllers and they may also communicate with the model. Some of the restricted communication patterns between the elements are: users can not directly interact with controller or models, views cannot communicate with view or models cannot interact with models. Therefore, the developer need to set the interaction structure based on events that occur in the application and build the communication interface between the application modules separate from the event handling modules. The outcomes of the above consecutive design steps are an application workflow and a set of application modules with clearly defined interaction structure.

4.2.7 Realization (Step 7)

Up until this point, we have the necessary development steps for the construction of Cyber-applications. Therefore, this step focuses on the instantiation of the entire application. We have to build the application to realize the final structure through the design steps. This includes instantiating all the dependencies such as data store connections, instantiating the data schema, loading identified application modules, and compilation of codes. To realize the application, the developer should set up a routing that tells the application to start from a particular application module. As a result, the application module will be called to instantiate the application every time the application is running.

4.3 CONCLUSION

In conclusion, this chapter defines a design method for the development of Cyber-applications. We first outlined different concepts that are relevant for the purpose of our design method and how they are adopted in our methodology. Then, we defined the design principle and steps which are used for the purpose of Cyber-application. In the design process, the internal structure of Cyber-application is partitioned into a set of application modules to show the modular and dynamic behavior of the application. Our design steps achieve the separation of concern principle by assigning different application modules to handle separate task in the application. Additionally, we also provided mechanisms which helps to manipulate third party data sources and to dynamically incorporate new data and components into the application. Generally, our design method provides a mechanism to combine widely available data sources (Services) and required functionalities; and converts them into a module structure in the application to provide solutions for domain problems. The approach is useful to develop web applications which intend to provide solution for spatial problems over the web environment.

Chapter 5

Design and Implementation of Quantitative Multi-hazard Vulnerability Analysis Application

5.1 INTRODUCTION

In this chapter, we describe the Cyber-application development process through a simple application. Section 5.2 and Section 5.3 presents the implementation environment used for development and the underlying architecture for the application respectively. In Section 5.4, we want to demonstrate that the concepts documented in previous chapter are actually valid. For example, separation of concern is something that we follow as a design principle. Hence, it needs to be validated by our simple application among other concepts such as partitioning of the application, workflow specification, data manipulation and event-based component interaction. A use case has been identified to build the prototype for demonstration purpose. Therefore, this section presents a complete description of the use case and its implementation aligned with the design steps that are defined in Chapter 4. We also present the evaluation of the prototype built in Section 5.5.

5.2 IMPLEMENTATION ENVIRONMENT AND SETUP

This section discusses the implementation set up used to implement the Quantitative multi-hazard vulnerability analysis application. It includes the development platform used as well as the setup and the software's. The lists are summarized in Table 5.1 below.

Table 5.1 Relevant Software and Platforms

Software's and Platforms	Purpose
Microsoft Visual Studio 2013	Development platform
ASP.Net MVC 3	Framework
C#	Front-End programming Language
JavaScript, Jquery	Scripting Languages
HTML, CSS	Interface Design
PostgreSQL 9.2	Database Design

We used Microsoft Visual studio 2013 the web developer edition, as the development platform for the construction of the application since it enables us to create our application on the basis of Model-View-Controller (MVC) design pattern. It includes project templates and solutions for developing MVC based web applications. ASP.Net MVC 3 framework is used for the purpose of our application since the design method for the development of Cyber-application is based on the MVC design pattern. It is a lightweight and testable framework which provides support for test-driven environment and separation of application tasks. The programming language used at front-end for the development of the application is C#. For the client side scripting, we use

Jquery, HTML, and CSS. For the back-end, we used PostgreSQL to store the application's data. PostgreSQL is open source software that is one of the well known relational database system.

In ASP.net MVC controllers are C# classes, mostly derived from the *System. Web.Mvc.Controller* class. When a request is sent to the URL with an action method, the controller classes are executed to perform some operations on the domain model and select view to display to the client. The ASP.net MVC framework delivers a choice for view engines, in this particular choice, a view engine which process ASPX pages using streamlined versions of the Web Forms markup syntax is used. The framework does not enforce any constraints on the implementation of our own model. We can create a model using regular C# objects and implement persistence using any data storage. We are free to select any technology that will interoperate with the .NET framework if we are using asp.net MVC framework. The architecture of these technology elements based on the ASP.net MVC framework is included in Appendix E.1.

5.3 ARCHITECTURE OF THE APPLICATION

The architecture presented in Figure 5.1 shows the actual functionality of the application and it is based on the MVC design pattern. This architecture emphasizes on achieving the basic separation of concerns principles and unit testability of the application. In MVC framework, the most essential separation is between the data management and application logic [11]. The architecture supports a normal web server interaction. The application send request to the server and the server responds to the web browser in a similar fashion.

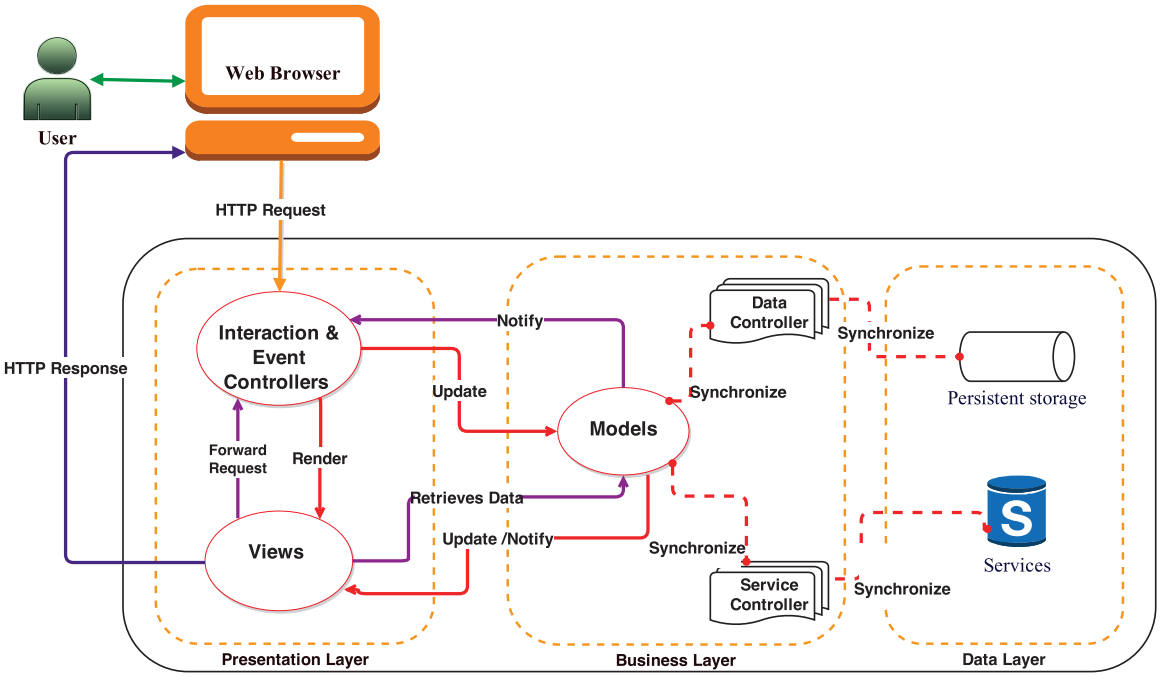


Figure 5.1: Architecture showing Components of the Application

This architecture is a 3-tier model. This three tier architecture creates a boundary to implement the elements MVC architecture. The models, controllers and views of the application are distributed through the different layers of the three tier model. The presentation layer contains the graphical user interfaces and reports; the business layer contains the business rules, data management and data manipulation. The data layer contains data storage and data retrieval objects.

Therefore, the distribution of the MVC elements in this three tier architecture is based on their responsibility in the application. Since the Model contains a lot of complicated business logic and spans the data layer, it is represented in the business layer. The view elements exist in the presentation layer and this allows the developer to render results. The controller is distributed between the presentation and business layer. In this architecture, we present three controllers as an example to show the distribution: Interaction controller, data controller and service controller. These controllers are distributed in the architecture layers based on their responsibility in the application. The interaction controller responds to requests from multiple browsers and devices. MVC framework breaks the previous standard web request processing scheme. This is because; the MVC web requests are handled by the appropriate controller instead of being processed by web pages. Based on the request coming from the URL of the browser (user/observer), ASP.net MVC framework chooses and executes the appropriate controller. The Data controller and the Service controller are located in the business layer. The responsibility of the Data controller is to provide data access and persistence operations against the storage database of the system. It is used to bind the data to the model elements. The Service controller is used to access the data that are outside the boundaries of our application, such as web services and external systems.

5.4 QUANTITATIVE MULTI-HAZARD VULNERABILITY APPLICATION DEVELOPMENT

This section presents the validation of the methodology developed. Therefore, the design and implementation process for developing quantitative multi-hazard vulnerability analysis application follows the design steps that are discussed in Section 4.2.

5.4.1 Step 1: Problem Definition.

As stated in the methodology, the first step for the development of Cyber-applications is defining the problem. Therefore, we summarize the problem and use case identified for multi-hazard vulnerability assessment as follow.

Use Case Identification

Quantitative multi-hazard vulnerability analysis has been identified as a use case. Mainly, because its requirement to use different datasets to perform vulnerability analysis for the elements at risk. Additionally, we want to demonstrate that Cyber-applications can be implemented for real world problem. This use case presents us the opportunity to choose between different datasets; and use them in the computational models for the purpose of the application. Moreover, the use case enables the application to perform complex tasks since the analysis will be performed for multiple hazards. Handling complex tasks in the web environment is one of the behaviors of Cyber-application. Therefore, each of the concepts about Cyber-application and the steps we will follow to create the application can be demonstrated using this identified use case.

The test datasets corresponds to Monte Albino area, in Nocera Inferiore (Southern Italy). This area has 10 catchments as well as 10 open slopes[1]. The area is threatened by hyperconcentrated flows, floods, mudflows and landslides hazards. There are over 300 plus buildings in the area and all of these buildings are assumed to be at risk. Therefore, we want to use the vulnerability analysis of the elements at risk as an example to demonstrate our concepts by constructing an application. This analysis only uses buildings and the population information which are considered to be the important elements at risk.

[1]open slopes are usually slippery and this open slope was affected by a first disaster which caused destruction of some buildings in 2005

Use Case Description

The Quantitative multi-hazard vulnerability analysis application should be concerned with a use case that determines vulnerability value for buildings. The vulnerability analysis will be carried out for three types of hazards. Namely: landslide, flooding and mudflows. The application should provide risk analysis for elements at risk particularly for buildings. It should enable users to select building of interest and provide the vulnerability value based on the selected hazard type. The analysis should be performed for each building to check which buildings are exposed for the hazard. The application requires different datasets that are available for Nocera region and external services like open layers for visualization purpose. For example, to provide visualization of the buildings in the Google map for users; enable them to select a single building for determining the vulnerability value depending on the hazard type they choose.

Data requirement identification

This step in the design process includes the identification of data that are required for the development of the application. The required datasets relevant for the purpose of the prototype are presented in Table .

Table 5.2 Required Datasets

Hazard type	Required Computational models	Required Datasets
Flooding	FLO-2D	Flood depth 20 years
		Flood depth 100 years
		Flood velocity 20 years
		Flood velocity 100 years
Landslides	Empirical assessment of runout zones	Landslide distance
		Landslide runout
Mudflows	Safety factor analysis TRIGRS, runout with FLO-2D	Mudflow depth
		Mudflow velocity
-	-	Building information
		Building geometry

Based on the requirements of the application, the computational model FLO-2D for flooding requires maximum depth and velocity of the flooding for each building. To give the option for users to select datasets based on year, the application requires 20 years and 100 years flood depth and velocity datasets. Mudflow Depth and velocities modeled with TRIGGRS and FLO-2D requires rainfall event of 200 year return period. To compute the landslide vulnerability, Landslide distance and Landslide runout datasets are required. The required Landslide distance dataset indicates how far the landslide moves further than the building and it is the empirically derived runout distance of landslides from ten open slopes in between valleys. Landslide runout describes how far the building is located from the toe of the unstable slope. Building information dataset is also required to provide information about the Materials of the building, number of Floors, Occupancy, People, Area, Value and etc for each building in the area.

5.4.2 Step 2: Data Structure Design.

Following the step in the development process, we create a PIM to represent the data and information which will circulate through the application. This PIM shows the platform independence part of the application. It is based on the description of the use case and the data requirements of the

application. The diagram of the PIM is presented in Appendix E.3 to show simple logical entities that represent the information base for the quantitative multi-hazard application and the relationships among them. In this PIM, five major entities are identified. Namely: Buildings, Landslide, Flooding20, Flooding100 and Mudflows. All the entities have their own attribute to describe their behavior as shown in the figure. This PIM needs to be transformed into platform specific models (PSMs) which are basically C# model elements of MVC. These models map properly to the PIM models.

5.4.3 Step 3: Data source identification.

After identifying the data requirement and creating the appropriate data models for the data that will be used by the application, the next step in the development process is to identify the suitable data sources from the available CyberInfrastructure that fits the models. For this, we identified a dataset that fits our purpose and the models we have created in the previous steps. The data was collected by the University of Salerno in the context of the EU SAFELAND project on the hazard and risk analysis of the area [5]. Table 5.3 shows the identified datasets for the quantitative multi-hazard analysis application.

Table 5.3 Example of Data Source Identification

Name	**General Information** (What is it?)	**Ownership** (Who is responsible?)	**Data Characteristics** (how it is measured?)	**Data Constraint**
Flood depth	Depth of flood	University of Salerno	modeled with FLO-2D with rainfall event	none
Flood velocity	Speed of flood	University of Salerno	modeled with FLO-2D with rainfall event	none
Landslide distance	Distance of landslide from the building	University of Salerno	Empirically derived distance of the runout of landslides from 10 open slopes b/n valleys	none
Landslide runout	Location of building from the landslide	University of Salerno	File of the 10 runout areas with info.	none
Mudflow depth	Depth of mudflow from the building	University of Salerno	modeled with TRIGGRS FLO-2D with rainfall	none
Mudflow velocity	Speed of mudflow	University of Salerno	modeled with TRIGGRS FLO-2D with rainfall	none
Building	General information about the building	University of Salerno	In terms of number of floor, people inside, value and so on	none

5.4.4 Step 4: Workflow specification.

Based on the requirements of the application and the identified dataset, we can now identify the general application workflow in order to categorize tasks and events that take place in the application. First we need to initialize the building list, the map, the hazard selection option and so on. Figure 5.2 shows the general workflow for the application.

The next process in the workflow is when the user selects a specific building from the list and request to perform vulnerability assessment for a chosen hazard type from the available options. There should be three options: Landslide, Flooding and mudflow analysis. Users should be able to choose between these hazard types to come up with risk analysis result. For a single building they will be able to compute three different hazard analyses. Computations of these analyses require the

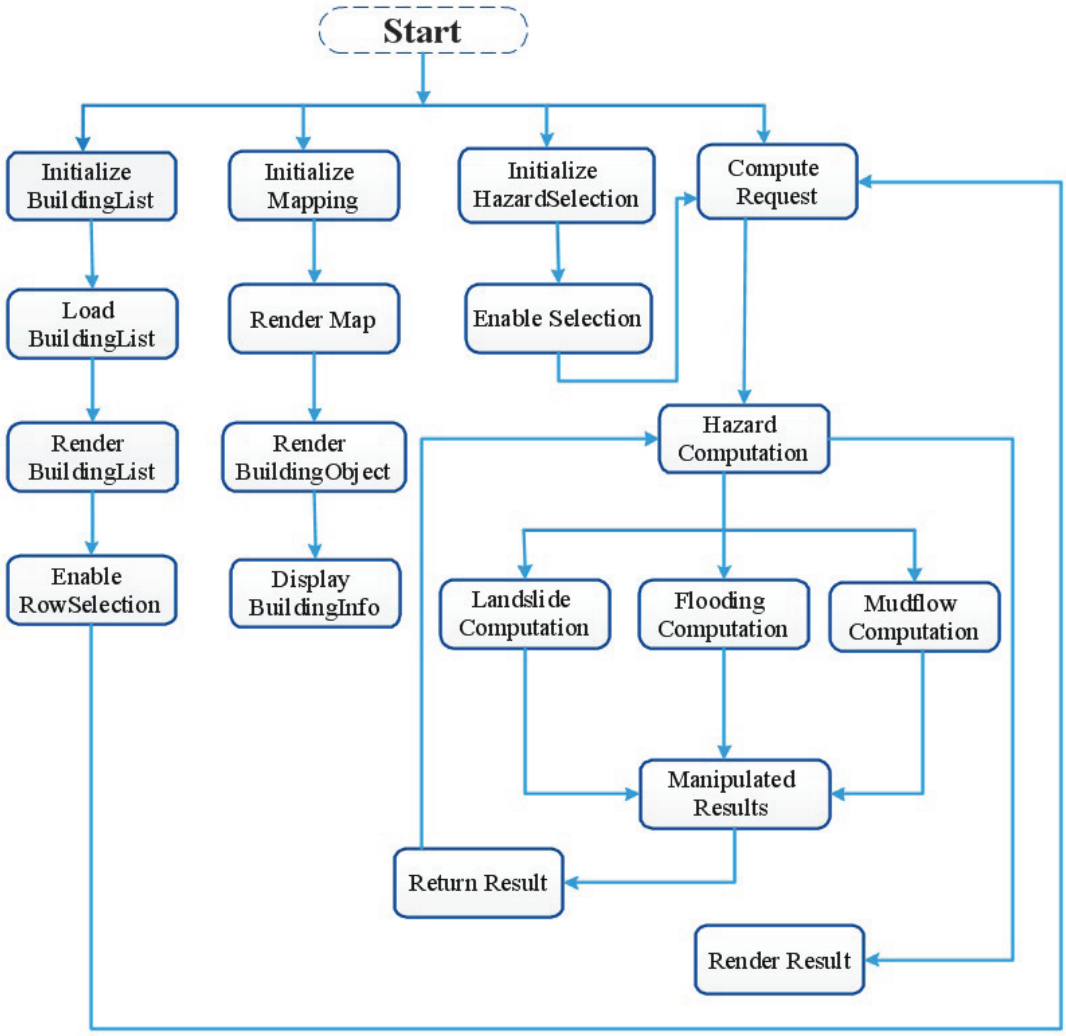


Figure 5.2: General Workflow Diagram for the Application

use of different datasets. Each of these analysis will pass through three major steps in the workflow in order to provide a complete output for users and achieve the application objective. The three major steps in the workflow of these computations are analysis of the datasets, building exposure analysis, and vulnerability Analysis. Therefore, based on the choice of the user, the workflows of the computations which are described below will be performed. These computations are behind computations and will be handled in a testing environment. However, the results of each of these computations will be used in the application. We describe their workflow in the next subsections.

Flooding Computation Workflow

The datasets for computing the exposure of a building for flooding are Buildings dataset, Flood Depth 20 years, Flood Velocity 20 years, Flood Depth 100 years, and Flood Velocity 100 years datasets. Basically, here the user is given the opportunity to choose which year dataset to use in the computation. After selecting the appropriate dataset, the computation begins by joining the buildings dataset with the selected flood depth dataset. The next step is to join the columns of the

two attribute tables and determine the maximum values per building. In the same step, the same thing works for the flood velocity; join buildings with flood velocity dataset and read the maximum values per building. The final step of the building exposure analysis is to determine how many buildings flooded taking a threshold of 10 centimeters. The output of the final computation will be used as an input for the Vulnerability analysis. This means, various levels of Vulnerability depending on the depth and velocity of flooding exposure analysis result are calculated. For simplification purpose refer the diagram in the Figure 5.3.

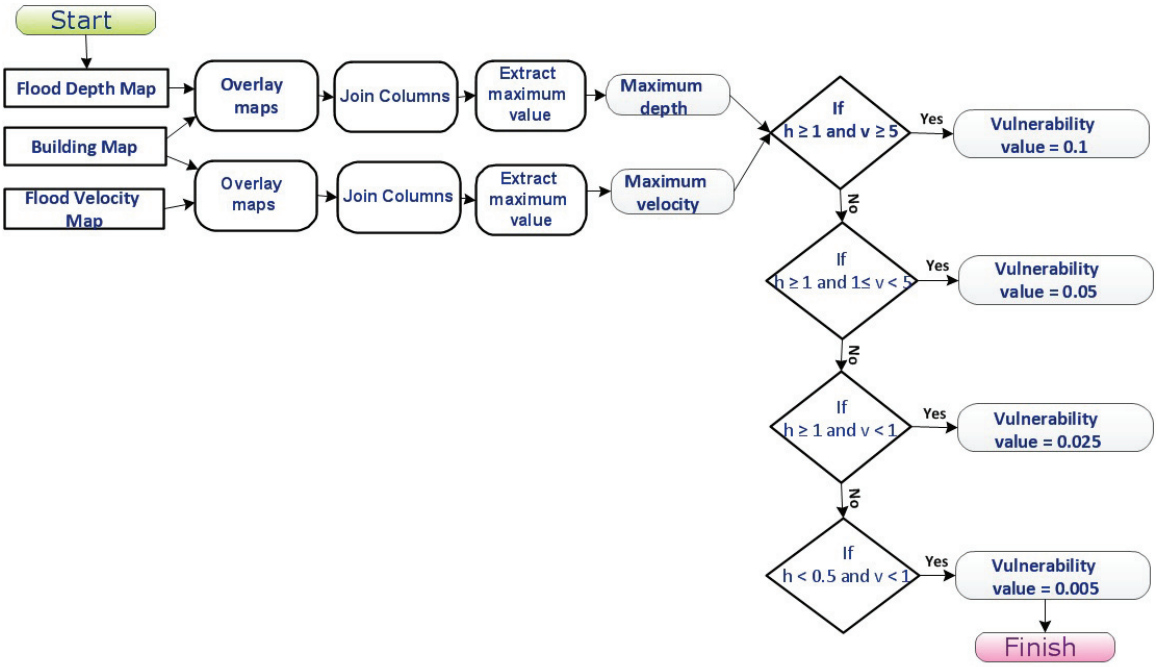


Figure 5.3: Workflow for determining flooding vulnerability value

Mudflow Computation Workflow

For the mudflow computation there exist three datasets identified. Namely: Buildings, Mudflow depth and mudflow velocity datasets. The logic behind determining the mudflow vulnerability value for each building follows the same workflow like the flooding computation. However, sets of vulnerability values will be assigned based on the maximum depth and velocity of the exposure analysis result are different. For example, if depth ≥ 1 and velocity ≥ 7, the respective value for the vulnerability is 1. If depth ≥ 1 and $3 \leq$ velocity < 7, the vulnerability value is 0.8. And finally, if depth < 0.5 and velocity < 3, the value is 0.05 and so on.

Landslide Computation Workflow

The datasets required as an input for this computation are: Building, Landslide runout and Landslide Distance datasets. The workflow behind this landslide computation is different than the above two hazard types. However this has been discussed in detail in Section 4.2.4.

5.4.5 Step 5: Application partitioning.

In reference to the task identified in the general workflow specification and the data model of the application, we are going to partition the application in this step. First, we separate the workflow into three main concerns; the data, the application logic and the presentation. To identify these parts in the structure of the application, we use the MVC design pattern. The Model represents the data concern, the View represents the presentation of the result and the Controller represents the core application logic. Once again, each of these elements are going to be partitioned into different parts such that they can handle and represent independent tasks in the application.

Creation of Model Classes

Partitioning the Model into different Model classes is based on the data model of the application. The data model (PIM) we created in Step 2 will be transformed into C# Model classes. As discussed previously, the model represents the domain entities and core information that the application needs to access and manipulate. Therefore, to represent the data models inside the application we create five model classes. Model class "building", "landslide", "flooding20", "flooding100", and "mudflows"; each to represent the storage in the application and have one to one correspondence with the entities of the data model. These Model classes will be instantiated and be used in the computation of the application. We have presented two examples for the creation of the "building" Model classes and "landslide" model class in Section 4.2.2 and we will present few more here in Listing 5.1 and 5.2; and the rest will be found in the Appendix A.

Listing 5.1: Model Class for "Flooding"

```
1  //Model Class for flooding 100 years information
2  namespace VulnerabilityAnalysis.Models
3  {
4      public class Flooding100Models
5      {
6          public double Build_ID { get; set; }
7          public double Height_100 { get; set; }
8          public double Velocity_100 { get; set; }
9          public double Building_Value { get; set; }
10     }
11 }
```

Listing 5.2: Model Class for "Mudflows"

```
1  //Model Class for Mudflows information
2  namespace VulnerabilityAnalysis.Models
3  {
4      public class MudflowModels
5      {
6          public double Build_ID { get; set; }
7          public double Mudflow_Height { get; set; }
8          public double Mudflow_Velocity { get; set; }
9          public double Building_Value { get; set; }
10     }
11 }
```

We will use these Model classes to represent their respective entities in the data schema. Each instance of the "Flooding" or "Mudflow" object corresponds to the instance within the PIM and each property of these classes will be mapped into the column in the table. This means the Models contain properties corresponding to the data model entity.

For the quantitative multi-hazard application, we have not created any model classes concerned with validation. However, Cyber-applications need to have model classes that are responsible

for validating the stores available in the application. This includes returning an error, rejecting unnecessary changes in the application store, and validating input datasets.

At this stage of the design and implementation process of the quantitative multi-hazard app, we have handled how the data is stored and represented inside the application structure. The methodology provides three mechanisms to fit the available or identified data to the model. However in our case, the available data fits the model. Therefore, we are just going to introduce the data to our application. To do this, we first create the tables in PostgreSQL and populate these tables with the actual data. Listing 5.3 shows the query statements used to create the tables and integrity constraints using PostGIS.

Listing 5.3: Queries used when creating tables

```
1  CREATE TABLE Building(
2       Building_id integer,
3       Material varchar(80),
4       Floor double,
5       Occupancy varchar(80),
6       People double;
7       Area double,
8       Value double,
9       geom geometry,
10      PRIMARY KEY(Building_id)
11 );
12
13 CREATE TABLE Landslide(
14      Building_id integer REFERENCES Building(Building_id),
15      Landslide_runout double,
16      Landslide_distance double,
17 );
18
19 CREATE TABLE Flooding_20(
20      Building_id integer REFERENCES Building(Building_id),
21      Height_20 double,
22      Velocity_20 double
23 );
24
25 CREATE TABLE Flodding_100(
26      Building_id integer REFERENCES Building(Building_id),
27      Height_100 double,
28      Velocity_100 double
29 );
30
31 CREATE TABLE Mudflows(
32      Building_id integer REFERENCES Building(Building_id),
33      Height double,
34      Velocity double
35 );
```

The Creation of Views

Considering the requirements of the application, we create three main views and three floating views. The “MapView”, to display a Google map as a geographical reference and it should also show the list of building objects overlaid for the particular area. This will allow users to visualize the vector representation of the buildings. The “HazardAnalysis” view, to display three options: landslide, flooding and mudflows. Each of these options needs to have “compute” buttons to calculate for vulnerability. For “Flooding”, there will be two options to let user’s select different datasets for the selected computation. A “GridView”, to display detailed information for each of the building objects; such as Material, Floors, occupancy, Value, etc. The rest of the views are floating views which will be used to display results for their respective hazard types. Therefore,

each of the hazards need to have independent views to display building vulnerability results for users.

The user can perform three operations in the presentation layer (the View). 1) Select a building from the map or the grid view to compute a vulnerability value. 2) Select which computational model (hazard analysis) they want to use from the available list. Possibly landslide, flooding or mudflow. 3) Then click on the relative "compute" button and get detailed information about the building and its vulnerability value to the different hazards types. Therefore for users to interact with the data, we create each of the above mentioned views based on their responsibility in the application. The source code for creating all of views is presented in Appendix C. However, for the purpose of a brief demonstration, we will discuss how we create few of the Views in the user interface. Refer Listing 5.4 and the explanation to understand how we created the "GridView".

Listing 5.4: Creation of the "GridView"

```
1
2  <div id="existingSlideView" style="width: 1155px;border:double; height: 450px">
3     <h5 style="height: 3px;margin-left: 2px;margin-top: 1px;color: green">BUILDINGS
          INFORMATION VIEW</h5>
4        <div id="BuildingTableContainer" style="width: 1153px; color: black"></div>
5        <div id="SelectedRowList">0</div>
6  </div>
7
8  <script type="text/javascript">
9         $(document).ready(function () {
10
11             $('#BuildingTableContainer').jtable({
12
13                 paging: true,
14                 sorting: true,
15                 defaultSorting: 'Build_ID ASC',
16                 selecting: true,
17                 multiselect: false,
18                 selectingCheckboxes: true,
19
20                 actions: {
21                     listAction: '/Home/BuildingList'
22                 },
23
24                 fields: {
25                     Build_ID: {
26                         key: true,
27                         list: true,
28                         width: '85px',
29                         title: 'Building ID',
30                         type: 'float'
31                     },
32                     Materials: {
33                         title: 'Materials',
34                         width: '350px'
35                     },
36                     Floors: {
37                         title: 'Floors',
38                         width: '120px',
39                         type: 'float'
40                     },
41                     Occupancy: {
42                         title: 'Occupancy',
43                         width: '250px'
44                     },
45                     People: {
46                         title: 'People',
47                         width: '140px',
48                         type: 'float'
49                     },
```

```
                    Area: {
                        title: 'Area',
                        width: '140px',
                        type: 'float'
                    },
                    Value: {
                        title: 'Value',
                        width: '200px',
                        type: 'double'
                    },
                },
            //Load Building list from server
        $('#BuildingTableContainer').jtable('load');

        });
</script>
```

In the above listing, we first use “div” which is an HTML tag to define a section in the user interface. The rest of the code is a Javascript code. We use Jtable a Jquery command to build the building table container. Following this, we set all the corresponding properties for the table. After handling the layout of the table, we inserted seven fields which are basically information about the buildings. This grid needs to be associated with the data store to work properly.

Listing 5.5: Results view

```

$("#dialog-form").dialog({
            title: "Results",
            autoOpen: false,
            height: 510,
            width: 400,
            modal: true,
            buttons: {
                Ok: function() {
                    $(this).dialog("close");
                }
            }
 });
```

Listing 5.5 shows how we create the floating view using Jquery UI dialog to display results. In the same listing, we also set the title and property for the dialog form. The corresponding HTML source codes which are useful to render the field sets inside this dialog form to display the results for the appropriate analysis are presented in the Appendix C.4. These Views do not work by themselves they need to be linked with a controller in order to be functional. For example, to display results whenever they are called by the appropriate controller.

The Controllers

Up to this point, we have handled how the data is stored and represented in the application according to the methodology. We have also created views to display information and results for users. What is left is to bind the application all together. The main responsibilities of the controllers are responding to requests and handling some of the processing in the application. Therefore, now we have to create different controllers which handle separate tasks in the application.

We identify five core controllers in the application which are “BuildingListing”, “Mapping”, “HazardSelection”, “DAL” (Data Access Layer) and “Executive” controllers. “BuildingListing” controller handles the request for automatic loading of the building list on the grid. The “Mapping” controller is responsible for the interaction with external services. In this particular case, it handles the communication with web map service (for example, Google maps). The “HazardSelection” controller is responsible for checking the selected hazard type and the selected row list

from the building grid table to forward the request for the appropriate controller. The "DAL" controller is responsible to manipulate the state of the data and used to bind the data to the model class of the application. The last controller is the "Executive" controller. This controller is responsible for processing the request from the hazard selection controller. It calls the "DAL" controller to get the data from the database based on the building Id forwarded by the "HazardSelection" controller and render a result on the appropriate view. The whole source code for all the controllers is available at the Appendix.

The "BuildingListing" Controller

This Controller responds to Ajax requests for automatic loading of building list on the grid. It calls the "DAL" to get all the buildings data from the database. It counts the number of buildings to show per page and finally return the result in the "GridView". Refer Listing 5.6. Anytime the application page is open, the grid will automatically load the data for the table when results are returned by this controller.

Listing 5.6: The "BuildingListing" Controller

```
1  namespace VulnerabilityAnalysis.Controllers
2  {
3      public class BuildingListing : Controller
4      {
5          [HttpPost]
6           public JsonResult BuildingList(int jtStartIndex = 1, int jtPageSize = 0, string
7           jtSorting = null)
8           {
9               try
10              {
11                  //Get data from database
12                  var buildings = DAL.GetAllBuildings();
13                  var bCount = buildings.Count;
14
15                  buildings = (from b in buildings
16                      where b.Build_ID >= jtStartIndex && b.Build_ID <= jtStartIndex + 10
17                      select b).ToList();
18
19                  //Return result to jTable
20                  return Json(new { Result = "OK", Records = buildings, TotalRecordCount =
21                  bCount });
22              }
23              catch (Exception ex)
24              {
25                  return Json(new { Result = "ERROR", Message = ex.Message });
26              }
27          }
28      }
29  }
```

The "Mapping" Controller

The "Mapping" Controller handles the communication with web mapping service. We are using Google maps as a base layer to display the building polygons. This helps the user to visualize the area and buildings which are threatened by the hazard. The communication is handled by a JavaScript API which is the most popular mapping API. This map can be customized based on interest. We first load the Google maps Javascript API and then we create and configure the map. This Javascript code contains a function called *Intitialize()* to run once when the page is loaded. There exist many options that can be set for the map but for the purpose of our demonstration we only used few. For example, we set the map to the location of the test site. We also add a marker

to locate the area for visualization purpose. After handling the loading of the map task, we overlaid the KML layer which contain all the building polygons on the top of Google map. The last step for this whole thing is to load the map on the user interface. The code for this controller is attached in Appendix B.3.

The "HazardSelection" Controller

This controller first checks which hazard analysis and which building ID are selected. If a building is selected and the selected hazard type is "Landside"; it will forwards the request with the building ID as a parameter for the appropriate controller using the "POST" request method to handle the actual task. The same thing works for "Flooding" and "Mudflow" analyses. This controller only handles the events and issues requests; but the actual task to be performed is handled by another controller which in this case is the "Executive" controller. In Listing 5.7, we present one of function from the "HazardSelection" controller which explains the above description for the case of "Landslide". Refer the rest of the functions in Appendix B.4.

Listing 5.7: Landslide function from the "HazardSelection" Controller

```
1
2  function LandSlide() {
3              var hazardSelected = $("input[@name=Hazards]:checked").val();
4              var buildid = $('#SelectedRowList').text();
5
6              if (hazardSelected == 'Landslide' && buildid > 0) {
7                 \$.ajax({
8                       url: "/Home/LandSlide",
9                       type: "POST",
10                      data: { buildId: buildid },
11                      dataType: "html",
12                      success: function(result) {
13                          $("#dialog-form").html(result);
14                          $("#dialog-form").dialog("open");
15                      }
16                  });
17
18             }
19 }
```

The "DAL" Controller

The DAL Controller is created to manipulate states in a database. When this controller is invoked, it will automatically connect to the storage, issue the proper query and return the results. Basically, it is just a code we write to interact with the data source and manipulate datasets. In this application module, appropriate functions are included based on the requirements of the application. For example, the application has to display list of buildings for users. To handle this operation, we have created the *GetAllBuildings()* function. This function handles setting up a connection with the data store, reading the data using SqlDataReader, selecting the appropriate columns using regular queries and finally returns information about lists of available buildings. This controller also includes the following methods: *GetLandSlide()*, *GetFlooding20()*, *GetFlooding100()*, and *GetMudflows()*. All of these methods handle similar but separate tasks. When they are invoked, will connect to the database, issue the appropriate query, and return the results. Listing 5.8 shows the "DAL" controller used for the quantitative multi-hazard application with only two of the methods. The rest of functions for this controller are included in Appendix B.2. It is recommended to make the code in the "DAL" controller database independent. This will make it much easier to switch the back-end database without affecting any of the application modules if necessary.

Listing 5.8: The "DAL" Controller

```
1
2  namespace VulnerabilityAnalysis.Controllers
3  {
4      public static class DAL
5      {
6          public static List<Building> GetAllBuildings()
7          {
8              var connectionString =ConfigurationManager.AppSettings["LandSlideConnString"];
9              var buildings = new List<Building>();
10             using (var connection = new SqlConnection(connectionString))
11             {
12                 SqlDataReader reader;
13                 var cmd = new SqlCommand();
14                 cmd.CommandText = "Select * from [dbo].[Buildings]";
15                 cmd.CommandType = CommandType.Text;
16                 cmd.Connection = connection;
17
18                 connection.Open();
19                 try
20                 {
21                     reader = cmd.ExecuteReader();
22                     while (reader.Read())
23                     {
24                         buildings.Add(new Building()
25                         {
26                             Build_ID = reader.GetDouble(0),
27                             Materials = reader.GetString(1),
28                             Floors = reader.GetDouble(2),
29                             Occupancy = reader.GetString(3),
30                             People = reader.GetDouble(4),
31                             Area = reader.GetDouble(5),
32                             Value = reader.GetDouble(6)
33
34                         });
35                     }
36                 }
37                 catch (Exception ex)
38                 {
39                     throw ex;
40                 }
41
42                 reader.Close();
43                 cmd.Dispose();
44                 connection.Close();
45             }
46
47             return buildings;
48         }
49
50         public static LandslideModels GetLandSlide(double buildId)
51         {
52             var connectionString =ConfigurationManager.AppSettings["LandSlideConnString"];
53             var landSlide = new LandslideModels();
54             using (var connection = new SqlConnection(connectionString))
55             {
56                 SqlDataReader reader;
57                 var cmd = new SqlCommand();
58            cmd.CommandText = "Select * from [dbo].[Landslides] Where Build_ID=" + buildId;
59                 cmd.CommandType = CommandType.Text;
60                 cmd.Connection = connection;
61
62                 connection.Open();
63                 try
64                 {
65                     reader = cmd.ExecuteReader();
66                     while (reader.Read())
67                     {
```

```
68                          landSlide = new LandslideModels()
69                          {
70                              Build_ID = reader.GetDouble(0),
71                              Landslide_runout = reader.GetDouble(1),
72                              Landslide_distance_max = reader.GetDouble(2),
73                              Landslide_distance_beyond = reader.GetDouble(3),
74                              Building_Value = reader.GetDouble(4),
75
76                          };
77                      }
78                  }
79                  catch (Exception ex)
80                  {
81                      throw ex;
82                  }
83
84                  reader.Close();
85                  cmd.Dispose();
86                  connection.Close();
87              }
88
89              return landSlide;
90          }
91  }
```

The "Executive" Controller

This controller is responsible for handling requests coming from the "HazardSelection" controller. When the user selects one of the hazard types, the request will be directly forwarded to this controller from the "event handler" which is basically the "HazardSelection" controller. What it does is, it accepts the request and executes it. To execute the request, it calls the appropriate function from "DAL" controller to get the relative data from the database based on the forwarded building Id. The processing step for the data behind the computation is explained in the workflow specification in Section 5.4.4 for all the three hazard types. Finally, the result will be returned and the same controller renders this result on the appropriate view. To show how the tasks are handled by this controller, we present Listing 5.9 which shows the action result only for the Flooding computation request. For the rest of the code refer Appendix B.5.

Listing 5.9: Function that handles flooding request in the "Executive" Controller

```
1
2  public ActionResult Flooding(double buildId, int year)
3  {
4    if (year == 20)
5    {
6        var flooding20 = DAL.GetFlooding20(buildId);
7        return this.PartialView("Flooding20PV", flooding20);
8    }
9        else
10       {
11           var flooding100 = DAL.GetFlooding100(buildId);
12           return this.PartialView("Flooding100PV", flooding100);
13       }
14 }
```

The user is expected to select which year dataset to use for "Flooding" analysis in the user interface. Therefore, in the above listing, the method accepts two parameters the "buildId" and "year" from the *Flooding()* function which is found in the "HazardSelection" controller. It will first check if the year selected is 20. If this is the case, it will call the *GetFlooding20()* function from "DAL" controller to get the flooding vulnerability information for the particular building. The results then will be rendered in the partial view "Flooding20PV". Otherwise, the selected year is

100 and the same operation will be performed for the relative year and displays the result on the appropriate view.

In general, the "Executive" controller returns different types of action results when a request is made. *ActionResult* is used to perform core operations on behalf of the action method. As far as we have seen, we have created different controllers and distribute application responsibilities across them. It is always a good practice to start distributing responsibilities by deciding how the views are going to be controlled in the application.

Results of the partitioning

Figure 5.4 shows how the solution structures for the quantitative multi-hazard application looks like after the partitioning is completed. We encapsulate all the components of the application inside a project name "Landslide". We use the .Net Framework 4 as a development framework. The figure shows the partitioning of the MVC design pattern. The codes of the application are grouped under the Model, the View and the Controller elements.

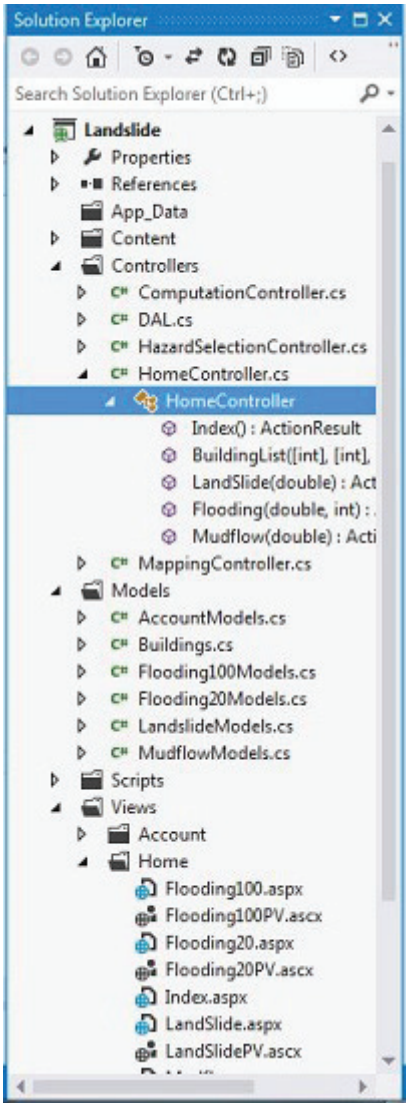


Figure 5.4: The Structure of Our Solution

5.4.6 Step 6: Interface specification.

We have all the partitioned application modules now. Therefore according to the methodology, we need to define their communication pattern based on events. This basically means, we will build the interaction interface between the partitioned components.

Example of event-Based Interactions

In the Quantitative multi-hazard application, there exist three "compute" buttons that are already created when constructing view elements. Therefore, we will define the click action on these buttons such that users can interact with the view and requests for vulnerability value based on preceding parameter selection for each hazard type. Since their interaction is event-based, we identify events and define call-back functions that are triggered as a response to these events. On the click event of each "compute" buttons, the corresponding functions from the "HazardSelection" controller will be invoked as shown in Listing 5.10. The call-back functions then initializes the appropriate controlling functions for the selected hazard type from the "Executive" controller.

Listing 5.10: On Click Events

```
1
2 <button id="btnLandslide" style="margin-left: 144px" onclick="LandSlide();return false;">
      Compute</button>
3
4 <button id="btnFlooding" style="margin-left: 150px" onclick="Flooding();return false;">
      Compute</button>
5
6 <button id="btnMudflow" style="margin-left: 150px" onclick="Mudflow();return false;">
      Compute</button>
```

As we can see, the communication between the application modules is based on events. The event can be instantiated from the view and be the base for all other application modules to interact. Another aspect of the event-based interaction we can demonstrate here is the change in selection of a row in the building grid list. This event is handled by a function which will append the selected building Id to a temporary variable named "record". Refer Listing 5.11 below to understand the results of the whole function.

Listing 5.11: Selection Changed event handler function

```
1
2 //Register to selectionChanged event to handler events
3  selectionChanged: function() {
4              var $selectedRows = $('#BuildingTableContainer').jtable('selectedRows');
5                  $('#SelectedRowList').empty();
6                     if ($selectedRows.length > 0) {
7                       $selectedRows.each(function() {
8                          var record = $(this).data('record');
9                             $('#SelectedRowList').append(record.Build_ID);
10                      });
11                    } else {
12                      //No rows selected
13                      $('#SelectedRowList').append('No row selected! Select rows to see
                            here...');
14                    }
15 }
```

Generally, at this stage of the design process, we have all the required partitioned application modules with clearly defined interaction structure to be able to run the application.

5.4.7 Step 7: Realization.

The final step of this design process is to instantiate the application. The final result is, the Quantitative multi-hazard analysis application. All of the models, the views and controller elements are now functional and perform as required. Figure 5.5 shows how the final application looks like. We can see the "GridView" loaded with buildings data, the map is operational, the buildings are overlaid in the map and also all the available hazard types are displayed on the top left of the page.

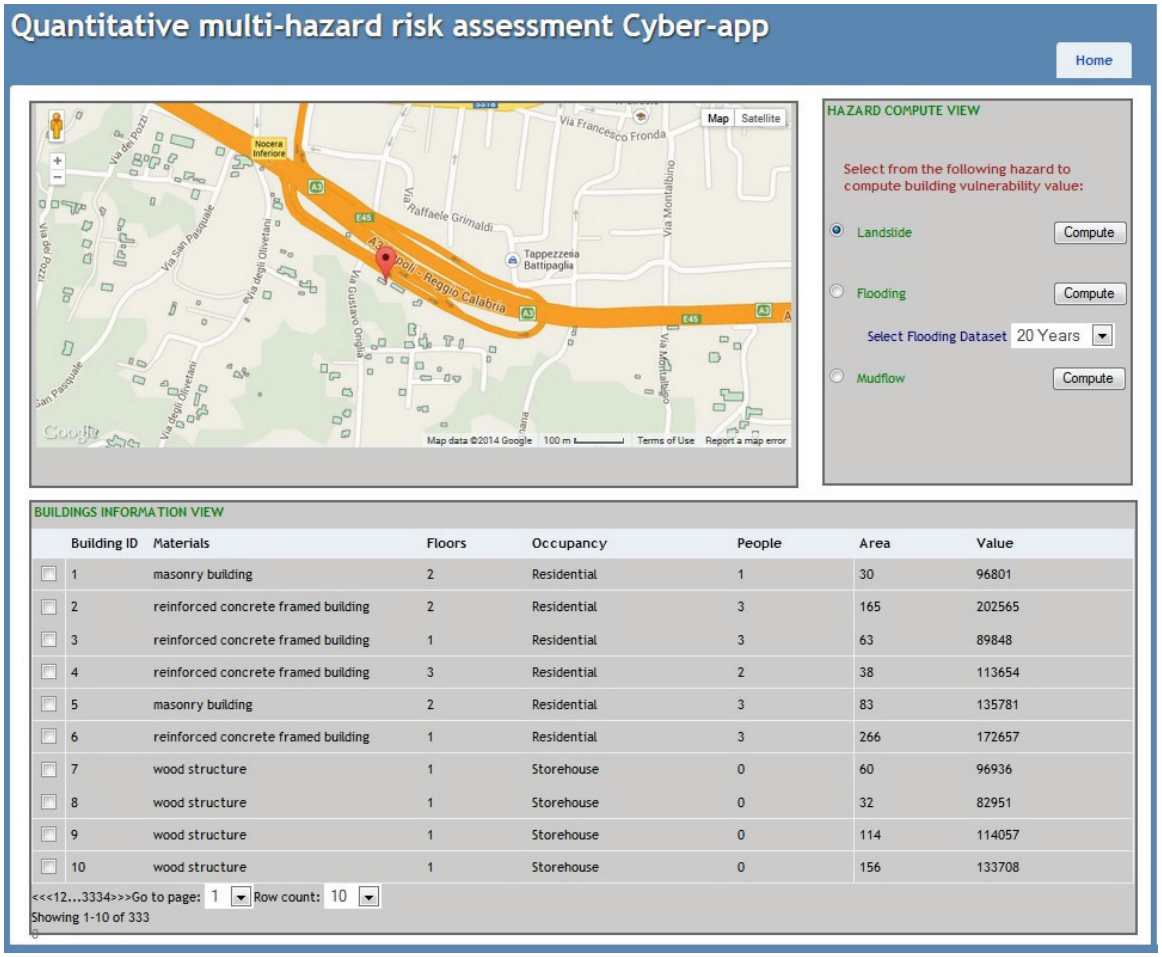


Figure 5.5: Application page layout

In Figure 5.6, it is visible that a small information window is displayed in the "MapView". This happens when one of the buildings is being selected from the "MapView", it gives detailed information about that particular building.

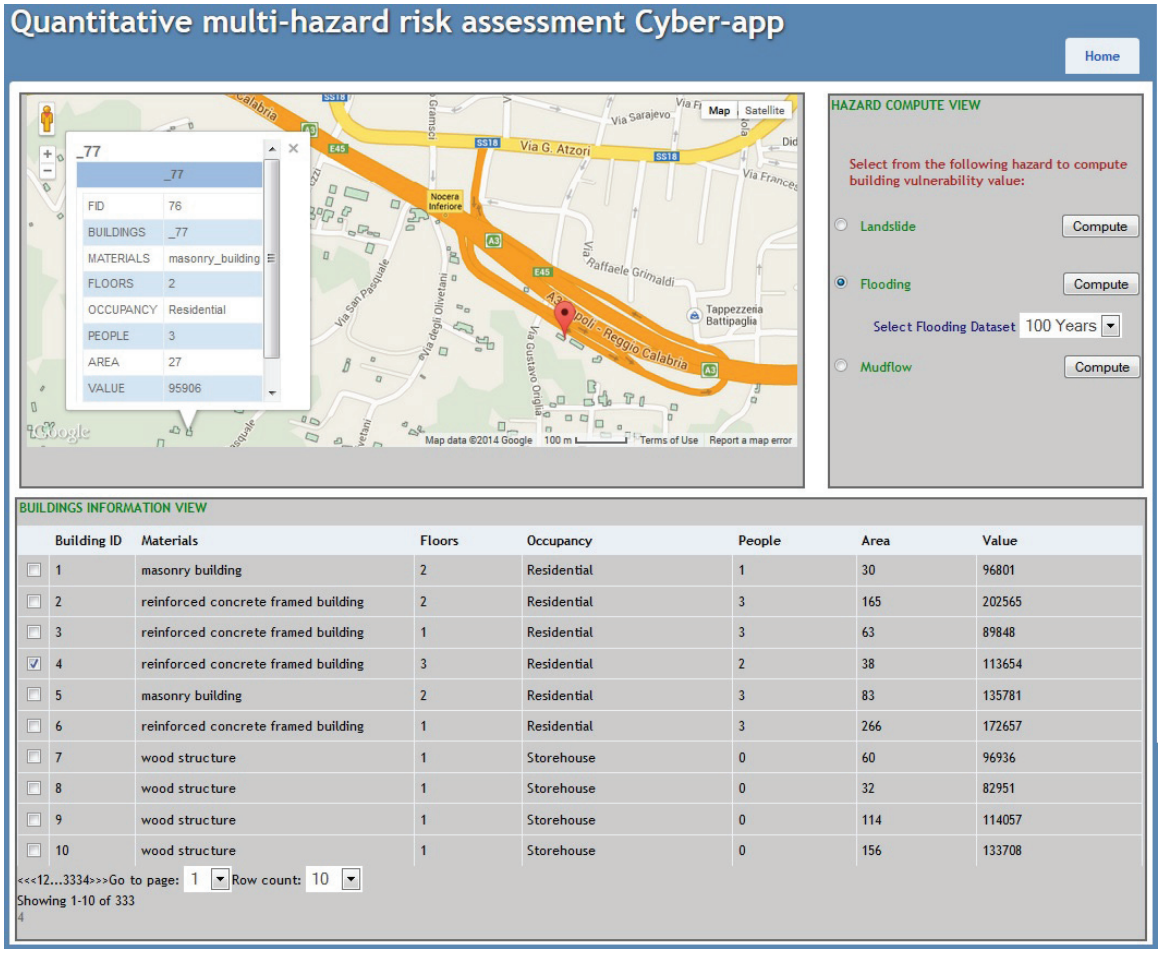


Figure 5.6: Building information on-click

The "ResultsView" which is basically a floating view is shown in Figure 5.7. This view presents results when a request is made. As an example, building 4 is selected and the result shows the vulnerability value for flooding. This view presents all the necessary information that helps in assessing the vulnerability value. (E.g., Depth and Velocity. The values are 0.16 meter and 0.14 meter per second respectively). The resulting vulnerability value for the selected building is 0.005. This means the building is vulnerable for flooding. Refer the scale to understand the resulting values in Appendix E.2.

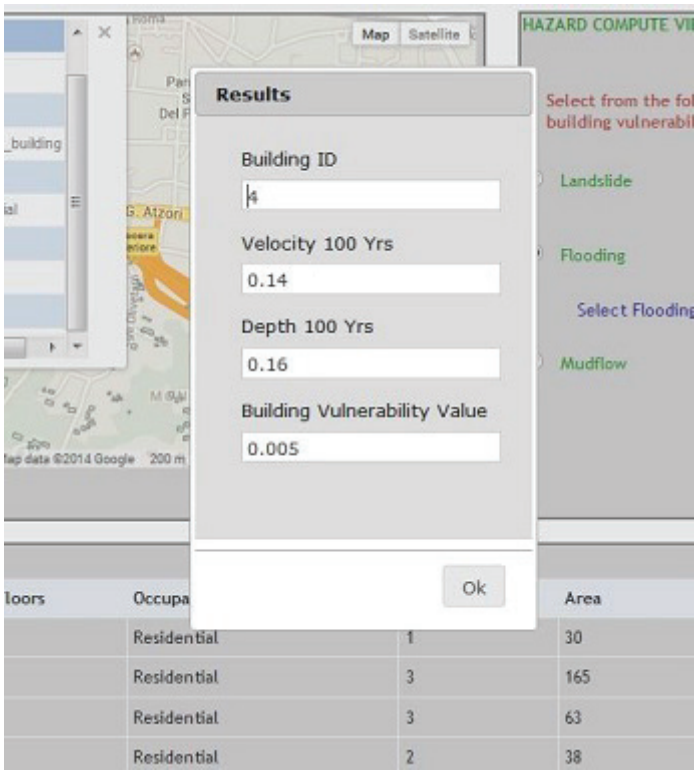


Figure 5.7: Results view

5.5 EVALUATION OF THE APPLICATION

We were able to demonstrate many of the concepts in our design method though the implementation of the Quantitative multi-hazard analysis application. All the above mentioned functionalities of the application are operational. This small application is enough to validate most of our concepts for the purpose of our research project. However, improvements can be made to this application. To mention few:

- It would be more convenient for the users if better visualization of the results is provided.
- Including additional components which can let users choose which parameters should be included in the computational model is another functionality that can be provided.
- Include real-time data and more computational modules into the application to be able to test the concept of the DDDAS method.

5.6 CONCLUSION

This chapter demonstrates the basic concepts of the design method. We were able to develop a small application through the design steps from the methodology. The basic concepts that are tested through the Quantitative multi-hazard analysis application are data formalization, workflow specification, application partitioning, event based components interaction, and responsibility distribution between application components. Therefore, we can say all of these concepts are valid and can be used for developing any kinds of Cyber-applications.

Chapter 6
Summary, Conclusions and Recommendations

6.1 INTRODUCTION

In this chapter we restate our research project and summarize the basic concepts. Section 6.2 presents summary of the achievements. We also review the research objectives and methods used to determine if all the research questions are answered. This will help us to point out the strength and limitation of the thesis. Based on this, we present suggestion for related future research in the final section of this chapter.

6.2 SUMMARY

The main objective we set out to accomplish in this research project was to define a design method for the development of Cyber-application. To achieve this objective, we first reviewed basic concepts relevant for the creation of a Cyber-applications, which are presented in Chapter 2. The review includes available data sources, geospatial cyberInfrastructure and its various advantages towards recent application developments. Cloud computing and other related concepts which greatly participate in the growth of spatial research areas have also been discussed. Cyber-application is one example, which takes full advantages of the widely available data sources, infrastructures and cloud based development platforms to provide better solutions for real world problem. The availability of different data sources and computational models in the CyberInfrastructure enable the application to provide full functionalities for users. In Chapter 3, we reviewed the developments of web application together with their development perspectives, few of web application types and proper methodologies which guide their development process. Most of the methodologies reviewed focuses on the navigational design and handle the separation of the data aspect from the presentation to preserve consistency of data presentation. However, Cyber-application requires a different approach since the main focus is the exploitation of new data sources and the integration of new data and components into the application. Therefore, we defined a new approach to guide the development process and we overlaid some of their strong aspects in our methodology.

We introduced an MVC and DDDAS based design method in Chapter 4 for the development process and evaluated it through the creation of quantitative multi-hazard analysis application. This methodology offers clean separation of concern and laid out properly each of the design steps that are involved in the development process. Our approach uses the principle of application partitioning to separate the application into independent modules such that each can deal with separate concern. The MVC Design pattern is used to identify and structure the partitioned application modules. We also provided three situations and mechanisms to handle the data aspects of the application. The SQL/MED is one mechanism to exploit and manage external data sources based on the requirements of the application, using commands and regular SQL queries. If we do not have an already existing data, we can use this mechanism to get access to different datasets that fits the predefined structure of the application. The DDDAS method is another mechanism which is used to treat new data that is going to be integrated into an already executing application. We use

this mechanism if the new data that is discovered at run-time does not fit with the predefined data structure of our application. This method involves set of steps to reach to the required outcome. It has been already described in Chapter 4.

In Chapter 5, Quantitative Multi-hazard analysis is identified as a use case since it allows us to demonstrate our concepts. This application is developed by following the design steps. Through this application we were able to demonstrate the concepts of data identification, workflow specification, the principle of application partitioning and responsibility distribution, interface specification, and etc. The source codes used to develop the quantitative multi-hazard application are included in the appendix. All in all, we were able to define a design method for the creation of Cyber-application and validate the methodology through the development of a prototype.

6.3 CONCLUSIONS

In this section, we present general conclusions derived from the research questions. We outline and discuss the achievements of the research questions posed in the beginning of this research project. We also present the problems encountered along with the limitations.

How to define a design method for the development of Cyber-application?

This question is fully covered and answered in Chapter 4. We present a solid design method for the construction of Cyber-applications. We were fortunate enough to realize from the literature review on web applications and their proper methodologies presented in Chapter 3, prior to define a proper design method for any types of web application; we first need to understand the purpose and requirements of the application; such that we can point out the design principle and design steps which guide the development process. Therefore, to define the design method, we first found out the basic design principles. Based on this, we defined a structured set of steps which are directly relevant for the development of the application. The definition of the design method starts from the identification of a problem domain and the description of the data requirements. The design steps also includes the design of data structure, the identification of data sources, the specification of workflow, the partitioning of application based on the tasks and events discovered when specifying the workflow, the specification of interaction structure between the application modules and finally the realization of the complete application structure. Each of the steps involves a number of concepts and mechanism that helps to fully compile the definition of the design method. We believe this robust methodology is useful in guiding developers of Cyber-application; since it considers all aspects of the application behavior and proposed different mechanisms for all the concerns. Additionally, we were able to test if the concepts are valid in the design method through a prototype and the result is fair enough.

What are the types of problems that can be addressed with the application?

The question draws its answer from the description given to the behavior of Cyber-application. These applications have a modular and dynamic behavior which offers an enhanced capability for integration of new data into the application. Therefore, they can be used for complex real world problems which require the use of different data sources or computational models to provide better and relatively accurate solution. This conclusion statement is made because we have actually tested the application in a real world problem. We built a Quantitative Multi-hazard analysis application which has been presented in Chapter 5, to prove the theory and demonstrate the fact that the application can be used for real world problems. In general, Cyber-applications can be

adopted for spatial problem that require real time data feeds to the computational model to enhance prediction capability, or provide accurate results for any other form of risk assessments due to their dynamic behavior.

Which application paradigm can we use to identify the components in the development of cyber-applications?

We proposed the MVC design pattern and the DDDAS approach as a base for defining the methodology for the construction of Cyber-applications in Chapter 4. The development of cyber-application requires the partitioning of the entire application into independent application modules; and the MVC design pattern was found a better fit to identify all of the partitioned modules. Each of the elements in MVC represents each of the modules that are required in the applications structure. We have considered other approaches (design patterns) as well; however, they do not comply with the purpose of the application. MVC is chosen because it naturally enforces separation of concern and ensures the independence of the application modules such that unit testing is possible. We can dare to say this design pattern was found to be well suited for the purpose of our design method since we have also tested it through a prototype.

The other application paradigm proposed for integration of new data and components into the application is the DDDAS method. This method is found to be useful because it provides the capability to fit new data or computational modules into an already existing application structure. DDDAS is a new design method and it is not widely practiced yet. Although, the concepts of this method are found to be strongly valuable for the purpose of our design method and the requirements of Cyber-applications, we did not have the chance to test it. This is because; the method requires more data and computational modules to be able to realize the application.

How to design the underlying architecture for Cyber-applications based on the design pattern?

This research question has been answered in Chapter 5. We presented the underlying architecture which supports the modular and dynamic structure of Cyber-applications. The architecture is defined based on the MVC design pattern and the concepts of three tier architecture. It has three layers; namely: the presentation, the business and the data layers. Each of the MVC elements of the application are distributed between these layers according to their responsibility in the application. Multiple aspects of cyber-application are considered in the architecture such as the services which provide different data's, data integration components, persistent data stores and so on.

How to validate the correctness and completeness of the design method for Cyber-application?

This question is about determining if our concepts are actually valid. Therefore, to validate the correctness and completeness of the design method, we developed a small application based on the proposed methodology. The sequence of steps that were proposed in the design method helped us to identify and separate different concerns such as initialization of a task, loading of data, rendering of a result, and so on based on the requirements of the application; and these results in implementable modules that realized the desired purpose of the application. For example, the "DAL" controller is an instance of such fact which is responsible for connecting with underlying database, issuing a query and returning the results in the application. It handles the manipulation of states in the data source. Therefore, all of the partitioned application modules handle separate tasks in the application since responsibility is already distributed between them. These application modules communicate through a well specified interface which is based on Event-based

paradigm. And as it is defined in the design method, we finally realized the structure of the application. The conclusion we draw from this brief summary is that, we were again fortunate enough to validate the correctness of the design method through the Quantitative Multi-hazard analysis Cyber-application which, its development process is fully described in Chapter 5. Even if the results were fair enough to validate the methodology, there still exist some shortcomings of the prototype built which will be discussed in the next section.

All in all, in the beginning of this research project, there were three challenges for Cyber-applications which are mentioned in the paper of Morales and de By [39].

1. A design method which provide useful framework to guide the development process.
2. Selection of a robust application paradigm to assemble disparate services into the application.
3. The underlying architecture to support the modular and dynamic behavior of Cyber-app.

Therefore, we strongly believe that we overcome these challenges through the learning process and the achievements of the research questions. We were able to define a design method which can result in cyber-applications that are moderate and can be adopted for any kinds of real world problem. We presented different mechanism for assembling data and services into the application and we also defined a simple architecture which takes into account multiple behavior of the application.

6.4 RECOMMENDATIONS

Based on the discussion on the previous sections, the following areas are identified and recommended for future work:

- Address the conversion of data.
- Extend the current functionality of the prototype.
- Include new data or component integration to the application to demonstrate the concepts of DDDAS.
- Assess the complexity and accessibility of cyber-applications.

Address the conversion of different data formats.

In our design method, we are assuming that we are able to read standard services (For example, Web Feature Service (WFS), Web Mapping Service (WMS), OGC standard services) and convert them into module structuring in the application using the mechanisms proposed in Chapter 4. However, our design method does not take into account the situation where the identified data sources or services cannot be converted automatically into a suitable format or structure for Cyber-application. What if we cannot convert them and create an intermediate data structure to be used by the application? This is something which should be considered. Further research should be conducted for the conversion of data that is going to be integrated into the application.

Extend the the functionality of the Quantitative multi-hazard cyber-app.

Even if this small application was suitable enough to validate the basic concepts in our methodology, the current functionality of the application can be enhanced by including more components to the system. For example, to provide better visualization of the results, perform analysis for individual risk, societal risk (the number of people that might be harmed by the hazards), and economic risk (expected loss in terms of economic value) and so on. This could be an interesting point of view to enhance the capability of the application.

Include more data and computational modules integration.

The prototype can be extended to a much more complete application to validate all the concepts in the methodology such as the DDDAS method. As stated in the conclusion, more data and computational modules are needed to be able to fully test this method. This particular aspect is addressed in Chapter 4 data Source identification step, such that other researchers can take the method proposed here and have a clear idea where to start if they want to include additional modules.

Assess the complexity and accessibility of Cyber-applications.

It is obvious that the target of this research is not the application itself; it is the methodology to get it done. However, it is a possible further direction to see the impact of the methodology in the complexity of the application as a whole. This can be done by building a complete Cyber-application by following the development stages and assess the impacts of the methodology in terms of accessibility and complexity.

Bibliography

[1] José Alfonso Aguilar, Irene Garrigós, Jose-Norberto Mazón, and Juan Trujillo. An mda approach for goal-oriented requirement analysis in web engineering. *J. UCS*, 16(17):2475–2494, 2010.

[2] H. Altarawneh and A. El Shiekh. A theoretical agile process framework for web applications development in small software firms. In *Software Engineering Research, Management and Applications, 2008. SERA '08. Sixth International Conference on*, pages 125–132.

[3] Luciano Baresi, Sebastiano Colazzo, Luca Mainetti, and Sandro Morasca. *W2000: A Modelling Notation for Complex Web Applications*, chapter 11, pages 335–364. Springer Berlin Heidelberg, 2006.

[4] Silvana Castano, Luigi Palopoli, and Riccardo Torlone. *A General Methodological Framework for the Development of Web-Based Information Systems*, volume 1921 of *Lecture Notes in Computer Science*, chapter 12, pages 128–139. Springer Berlin Heidelberg, 2000.

[5] Westen Cees van, Cascini Leonardo, Ferlisi Settimio, Sorbino Giuseppe, and Cuomo Sabatino. Tutorial on the use of gis for landslide risk assessment. the case study of monte albino, nocera inferiore, italy. 2011.

[6] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Elsevier Science, 2003.

[7] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, 33(16):137–157, 2000.

[8] Frederica Darema. *Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements*, volume 3038 of *Lecture Notes in Computer Science*, chapter 86, pages 662–669. Springer Berlin Heidelberg, 2004.

[9] Frederica Darema. Cyberinfrastructures of cyber-applications-systems. *Procedia Computer Science*, 1(1):1287–1296, 2010.

[10] O. M. F. De Troyer and C. J. Leune. Wsdm: a user centered design method for web sites. *Computer Networks and ISDN Systems*, 30(17):85–94, 1998.

[11] John Deacon. Model-view-controller (mvc) architecture. *Online][Citado em: 10 de marÃğo de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, 2009.

[12] Stijn Dekeyser and Richard Watson. Extending google docs to collaborate on research papers. *Toowoomba, Queensland, AU: The University of Southern Queensland, Australia*, 23:2008, 2006.

[13] Matthew DeMeritt. Simplifying citizen reporting. 2011.

[14] Damiano Distante, Paola Pedone, Gustavo Rossi, and Gerardo Canfora. *Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces*, volume 4607 of *Lecture Notes in Computer Science*, chapter 38, pages 457–472. Springer Berlin Heidelberg, 2007.

[15] Roland Eckert and Gunther Specht. Challenge of design data exchange between heterogeneous database schema. 2004.

[16] M.J Escalona and Gustavo Aragón. Ndt. a model-driven approach for web requirements. *Software Engineering, IEEE Transactions on*, 34(3):377–390, 2008.

[17] Renato Ferreira, Gagan Agrawal, and Joel Saltz. Data parallel language and compiler support for data intensive applications. *Parallel Computing*, 28(5):725–748, 2002.

[18] Piero Fraternali. Tools and approaches for developing data-intensive web applications: a survey. *ACM Comput. Surv.*, 31(3):227–263, 1999.

[19] Franca Garzotto. *Ubiquitous Web Applications*, volume 2151 of *Lecture Notes in Computer Science*, chapter 1, pages 1–1. Springer Berlin Heidelberg, 2001.

[20] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. *HDMA Model-Based Approach to Hypertext Application Design*, pages 794–806. Morgan Kaufmann, San Francisco, 2002.

[21] Christophe Gnaho. *Web-Based Information Systems Development - a User Centered Engineering Approach*, volume 2016 of *Lecture Notes in Computer Science*, chapter 11, pages 105–118. Springer Berlin Heidelberg, 2001.

[22] Jaime Gómez and Cristina Cachero. Oo-h method: extending uml to model web interfaces. *Information modeling for internet applications*, pages 144–173, 2003.

[23] Michael F Goodchild. in the world of web 2.0. *International Journal*, 2:24–32, 2007.

[24] Halepis Harriette. 5 things you need to know about cloud computing, 2013.

[25] Dat Dac Hoang, Hye-Young Paik, and Chae-Kyu Kim. Service-oriented middleware architectures for cyber-physical systems. *International Journal of Computer Science and Network Security*, 12(1):79–87, 2012.

[26] Barbara Hofer. Geospatial cyberinfrastructure and geoprocessing web: A review of commonalities and differences of e-science approaches. *ISPRS International Journal of Geo-Information*, 2(3):749–765, 2013.

[27] M. Jazayeri. Some trends in web application development. In *Future of Software Engineering, 2007. FOSE '07*, pages 199–213.

[28] Do-Hyun Kim and Kim Min-Soo. Web gis service component based on open environment. In *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International*, volume 6, pages 3346–3348 vol.6.

[29] Nora Koch and Andreas Kraus. *Towards a Common Metamodel for the Development of Web Applications*, volume 2722 of *Lecture Notes in Computer Science*, chapter 92, pages 497–506. Springer Berlin Heidelberg, 2003.

[30] Seung C. Lee and Ashraf I. Shirani. A component based methodology for web application development. *Journal of Systems and Software*, 71(1-2):177–187, 2004.

[31] A. Leff and J. T. Rayfield. Web-application development using the model/view/controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International*, pages 118–127.

[32] David J Maguire and Paul A Longley. The emergence of geoportals and their role in spatial data infrastructures. *Computers, environment and urban systems*, 29(1):3–14, 2005.

[33] Maciej Malawski, Jan Meizner, Marian Bubak, and Pawel Gepner. Component approach to computational applications on clouds. *Procedia Computer Science*, 4(0):432–441, 2011.

[34] Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800(145):7, 2011.

[35] Jim Melton, Jan Eike Michels, Vanja Josifovski, Krishna Kulkarni, and Peter Schwarz. Sql/med: a status report. *SIGMOD Rec.*, 31(3):81–89, 2002.

[36] Emilia Mendes, Nile Mosley, and Steve Counsell. *The Need for Web Engineering: An Introduction*, chapter 1, pages 1–27. Springer Berlin Heidelberg, 2006.

[37] Microsoft. Data identification, 2014.

[38] R. J. Mitchell. Applying the principle of separation of concerns in software development. *Annual Review in Automatic Programming*, 14, Part 1(0):21–27, 1988.

[39] Javier Morales and Rolf A. de By. Cyber-applications as gateway to data-rich digital earth systems. *International Journal of Digital Earth*, pages 1–18, 2013.

[40] San Murugesan and Yogesh Deshpande. Meeting the challenges of web application development: the web engineering approach, 2002.

[41] San Murugesan, Yogesh Deshpande, Steve Hansen, and Athula Ginige. *Web Engineering: a New Discipline for Development of Web-Based Systems*, volume 2016 of *Lecture Notes in Computer Science*, chapter 2, pages 3–13. Springer Berlin Heidelberg, 2001.

[42] John A Olson. Data as a service: Are we in the clouds? *Journal of Map and Geography Libraries*, 6(1):76–78, 2009.

[43] Ying Ouyang, Jia En Zhang, and Shi Ming Luo. Dynamic data driven application system: Recent development and future perspective. *Ecological Modelling*, 204(12):1–8, 2007.

[44] J Ponce, AH Torres, MJ Escalona, M Mejías, and FJ Domínguez-Mayo. Developing web geographic information system with the ndt methodology. 2012.

[45] Eeva Savolainen. Cloud service models. 2012.

[46] Daniel Schwabe, Gustavo Rossi, and Simone D. J. Barbosa. Systematic hypermedia application design with oohdm, 1996.

[47] Florian Schwendener. Sql/med and more. 2011.

[48] Jonathan Strickland. How cloud computing works, 2008.

[49] Isakowitz Toms, Edward A. Stohr, and P. Balasubramanian. Rmm: a methodology for structured hypermedia design. *Commun. ACM*, 38(8):34–44, 1995.

[50] Reenskaug Trygve. Mvc xerox parc 1978-79. *Trygve/MVC*, 1978.

[51] MarceloAugustoSantos Turine, MariaCristinaFerreira De Oliveira, and PauloCesar Masiero. Hyscharts: A statechart-based environment for hyperdocument authoring and browsing. *Multimedia Tools and Applications*, 8(3):309–324, 1999.

[52] William Voorsluys, James Broberg, and Rajkumar Buyya. Introduction to cloud computing. *Cloud computing: Principles and paradigms*, pages 2–44, 2011.

[53] Chaowei Yang, Michael Goodchild, Qunying Huang, Doug Nebert, Robert Raskin, Yan Xu, Myra Bambacus, and Daniel Fay. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth*, 4(4):305–329, 2011.

[54] Chaowei Yang, Robert Raskin, Michael Goodchild, and Mark Gahegan. Geospatial cyberinfrastructure: Past, present and future. *Computers, Environment and Urban Systems*, 34(4):264–277, 2010.

[55] Peng Yue, Hongxiu Zhou, Jianya Gong, and Lei Hu. Geoprocessing in cloud computing platforms: a comparative analysis. *International Journal of Digital Earth*, 6(4):404–425, 2012.

[56] Fujun Zhu, Mark Turner, Ioannis Kotsiopoulos, Keith Bennett, Michelle Russell, David Budgena, Pearl Breretona, John Keane, Paul Layzell, and Michael Rigby. Dynamic data integration using web services. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 262–269. IEEE.

Appendix A
Creation Of the Models

A.1 CREATION OF ALL THE MODEL CLASSES

Listing A.1: Complete Source Code for Buildings Model Class creation

```
1  //Model class for Buildings
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel.DataAnnotations;
5  using System.Linq;
6  using System.Web;
7
8  namespace Landslide.Models
9  {
10     public class Building
11     {
12
13         public double Build_ID { get; set; }
14
15         [DataType(DataType.MultilineText)]
16         public string Materials { get; set; }
17
18         public double Floors { get; set; }
19         public string Occupancy { get; set; }
20         public double People { get; set; }
21         public double Area { get; set; }
22         public double Value { get; set; }
23
24     }
25 }
```

Listing A.2: Complete Source Code for Flooding 20 Years Model Class

```
1  //Model class Flooding 20 years
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6
7  namespace Landslide.Models
8  {
9      public class Flooding20Models
10     {
11
12         public double Build_ID { get; set; }
13         public double Height_20 { get; set; }
14         public double Velocity_20 { get; set; }
15         public double Building_Value { get; set; }
16
17     }
18 }
```

Listing A.3: Complete Source Code for Flooding 100 Years Model Class

```
//Model class for Flooding 100 years
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Spatial;

namespace Landslide.Models
{
    public class Flooding100Models
    {
        public double Build_ID { get; set; }
        public double Height_100 { get; set; }
        public double Velocity_100 { get; set; }
        public double Building_Value { get; set; }
    }
}
```

Listing A.4: Complete Source Code for Model Class of Landslide

```
//Landslide hazard Model
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Landslide.Models
{
    public class LandslideModels
    {
        public double Build_ID { get; set; }
        public double Landslide_runout { get; set; }
        public double Landslide_distance_max { get; set; }
        public double Landslide_distance_beyond { get; set; }
        public double Building_Value { get; set; }
    }
}
```

Listing A.5: Complete Source Code for Model Class of Mudflows

```
//Model class for Mudflows
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Landslide.Models
{
    public class MudflowModels
    {
        public double Build_ID { get; set; }
        public double Height { get; set; }
        public double Velocity { get; set; }
        public double Building_Value { get; set; }
    }
}
```

Appendix B
The Controllers

B.1 BUILDING LISTING CONTROLLER

Listing B.1: Complete Source Code for Building Listing Controller

```
1
2   using System;
3   using System.Collections.Generic;
4   using System.Linq;
5   using System.Web;
6   using System.Web.Mvc;
7   using Landslide.Models;
8
9   namespace VulnerabilityAnalysis.Controllers
10  {
11      public class BuildingListing : Controller
12      {
13       [HttpPost]
14        public JsonResult BuildingList(int jtStartIndex = 1, int jtPageSize = 0, string
15        jtSorting = null)
16        {
17            try
18            {
19                //Get data from database
20                var buildings = DAL.GetAllBuildings();
21                var bCount = buildings.Count;
22
23                buildings = (from b in buildings
24                    where b.Build_ID >= jtStartIndex && b.Build_ID <= jtStartIndex + 10
25                    select b).ToList();
26
27                //Return result to jTable
28                return Json(new { Result = "OK", Records = buildings, TotalRecordCount =
29                bCount });
30            }
31            catch (Exception ex)
32            {
33                return Json(new { Result = "ERROR", Message = ex.Message });
34            }
35        }
36    }
37  }
```

B.2 CREATION OF DAL CONTROLLER

Listing B.2: Complete Source Code for creation of DAL (Data access layer)

```
1
2   using System;
3   using System.Collections.Generic;
4   using System.Configuration;
5   using System.Data;
6   using System.Data.SqlClient;
7   using System.Linq;
```

```
8   using System.Web;
9   using System.ComponentModel.DataAnnotations;
10  using System.Globalization;
11  using System.Web.Mvc;
12  using System.Web.Security;
13  using System.Data.Sql;
14  using  System.Data.Spatial.DbGeometry;
15
16
17  namespace Landslide.Models
18  {
19      public static class DAL
20      {
21          public static List<Building> GetAllBuildings()
22          {
23              var connectionString = ConfigurationManager.AppSettings["LandSlideConnString
                    "];
24              var buildings = new List<Building>();
25              using (var connection = new SqlConnection(connectionString))
26              {
27                  SqlDataReader reader;
28                  var cmd = new SqlCommand();
29                  cmd.CommandText = "Select * from [dbo].[Buildings]";
30                  cmd.CommandType = CommandType.Text;
31                  cmd.Connection = connection;
32
33                  connection.Open();
34                  try
35                  {
36                      reader = cmd.ExecuteReader();
37                      while (reader.Read())
38                      {
39                          buildings.Add(new Building()
40                          {
41                              Build_ID = reader.GetDouble(0),
42                              Materials = reader.GetString(1),
43                              Floors = reader.GetDouble(2),
44                              Occupancy = reader.GetString(3),
45                              People = reader.GetDouble(4),
46                              Area = reader.GetDouble(5),
47                              Value = reader.GetDouble(6)
48
49                          });
50                      }
51                  }
52                  catch (Exception ex)
53                  {
54                      throw ex;
55                  }
56
57                  reader.Close();
58                  cmd.Dispose();
59                  connection.Close();
60              }
61
62              return buildings;
63          }
64
65          public static LandslideModels GetLandSlide(double buildId)
66          {
67              var connectionString = ConfigurationManager.AppSettings["LandSlideConnString
                    "];
68              var landSlide = new LandslideModels();
69              using (var connection = new SqlConnection(connectionString))
70              {
71                  SqlDataReader reader;
72                  var cmd = new SqlCommand();
73                  cmd.CommandText = "Select * from [dbo].[Landslides] Where Build_ID=" +
                        buildId;
```

```
74                  cmd.CommandType = CommandType.Text;
75                  cmd.Connection = connection;
76
77                  connection.Open();
78                  try
79                  {
80                      reader = cmd.ExecuteReader();
81                      while (reader.Read())
82                      {
83                          landSlide = new LandslideModels()
84                          {
85                              Build_ID = reader.GetDouble(0),
86                              Landslide_runout = reader.GetDouble(1),
87                              Landslide_distance_max = reader.GetDouble(2),
88                              Landslide_distance_beyond = reader.GetDouble(3),
89                              Building_Value = reader.GetDouble(4),
90
91                          };
92                      }
93                  }
94                  catch (Exception ex)
95                  {
96                      throw ex;
97                  }
98
99                  reader.Close();
100                 cmd.Dispose();
101                 connection.Close();
102             }
103
104             return landSlide;
105         }
106
107         public static Flooding20Models GetFlooding20(double buildId)
108         {
109             var connectionString = ConfigurationManager.AppSettings["LandSlideConnString
                    "];
110             var flooding20 = new Flooding20Models();
111             using (var connection = new SqlConnection(connectionString))
112             {
113                 SqlDataReader reader;
114                 var cmd = new SqlCommand();
115                 cmd.CommandText = "Select * from [dbo].[Floodings20] Where Build_ID=" +
                        buildId;
116                 cmd.CommandType = CommandType.Text;
117                 cmd.Connection = connection;
118
119                 connection.Open();
120                 try
121                 {
122                     reader = cmd.ExecuteReader();
123                     while (reader.Read())
124                     {
125                         flooding20 = new Flooding20Models()
126                         {
127                             Build_ID = reader.GetDouble(0),
128                             Height_20 = reader.GetDouble(1),
129                             Velocity_20 = reader.GetDouble(2),
130                             Building_Value = reader.GetDouble(3)
131                         };
132                     }
133                 }
134                 catch (Exception ex)
135                 {
136                     throw ex;
137                 }
138
139                 reader.Close();
140                 cmd.Dispose();
```

```
141                 connection.Close();
142             }
143 
144             return flooding20;
145         }
146 
147         public static Flooding100Models GetFlooding100(double buildId)
148         {
149             var connectionString = ConfigurationManager.AppSettings["LandSlideConnString
                    "];
150             var flooding100 = new Flooding100Models();
151             using (var connection = new SqlConnection(connectionString))
152             {
153                 SqlDataReader reader;
154                 var cmd = new SqlCommand();
155                 cmd.CommandText = "Select * from [dbo].[Floodings100] Where Build_ID=" +
                        buildId;
156                 cmd.CommandType = CommandType.Text;
157                 cmd.Connection = connection;
158 
159                 connection.Open();
160                 try
161                 {
162                     reader = cmd.ExecuteReader();
163                     while (reader.Read())
164                     {
165                         flooding100 = new Flooding100Models()
166                         {
167                             Build_ID = reader.GetDouble(0),
168                             Height_100 = reader.GetDouble(1),
169                             Velocity_100 = reader.GetDouble(2),
170                             Building_Value = reader.GetDouble(3)
171                         };
172                     }
173                 }
174                 catch (Exception ex)
175                 {
176                     throw ex;
177                 }
178 
179                 reader.Close();
180                 cmd.Dispose();
181                 connection.Close();
182             }
183 
184             return flooding100;
185         }
186 
187         public static MudflowModels GetMudflow(double buildId)
188         {
189             var connectionString = ConfigurationManager.AppSettings["LandSlideConnString
                    "];
190             var mudFlows = new MudflowModels();
191             using (var connection = new SqlConnection(connectionString))
192             {
193                 SqlDataReader reader;
194                 var cmd = new SqlCommand();
195                 cmd.CommandText = "Select * from [dbo].[Mudflows] Where Build_ID=" +
                        buildId;
196                 cmd.CommandType = CommandType.Text;
197                 cmd.Connection = connection;
198 
199                 connection.Open();
200                 try
201                 {
202                     reader = cmd.ExecuteReader();
203                     while (reader.Read())
204                     {
205                         mudFlows = new MudflowModels()
```

```
206                             {
207                                 Build_ID = reader.GetDouble(0),
208                                 Height = reader.GetDouble(1),
209                                 Velocity = reader.GetDouble(2),
210                                 Building_Value = reader.GetDouble(3)
211                             };
212                         }
213                     }
214                     catch (Exception ex)
215                     {
216                         throw ex;
217                     }
218 
219                     reader.Close();
220                     cmd.Dispose();
221                     connection.Close();
222                 }
223 
224                 return mudFlows;
225             }
226         }
227 
228     }
```

B.3 MAPPING CONTROLLER

Listing B.3: Complete Source Code for Creating and Initializing the Map

```
1 
2   function initialize() {
3         var mapOptions = {
4             zoom: 15, center: new google.maps.LatLng(40.7516, 14.6467),
5              mapTypeId: google.maps.MapTypeId.ROADMAP,
6              scaleControl: true
7 
8         };
9         var map = new google.maps.Map(document.getElementById('map'), mapOptions);
10 
11 
12        var marker = new google.maps.Marker({
13            position: new google.maps.LatLng(40.7363, 14.6406),
14            map: map,
15            title: 'Study Area - Nocera Inferiore SA, Italy!'
16 
17         });
18 
19         var building = new google.maps.KmlLayer({
20             url: 'https://sites.google.com/site/mykmlbuildings/mykml/Buildings_LayerToKML.
                   kmz',
21 
22         });
23 
24         building.setMap(map);
25 
26   }
27 
28   function loadScript() {
29         var script = document.createElement('script');
30         script.type = 'text/javascript'; script.src = 'https://maps.googleapis.com/maps/
               api/js?v=3.exp&sensor=false&' + 'callback=initialize';
31         document.body.appendChild(script);
32 
33   }
34 
35   window.onload = loadScript;
```

B.4 HAZARD SELECTION CONTROLLER

Listing B.4: Selected Hazard type Controller

```

function LandSlide() {
    var hazardSelected = $("input[@name=Hazards]:checked").val();
    var buildid = $('#SelectedRowList').text();

    if (hazardSelected == 'Landslide' && buildid > 0) {
        $.ajax({
            url: "/Home/LandSlide",
            type: "POST",
            data: { buildId: buildid },
            dataType: "html",
            success: function(result) {
                $("#dialog-form").html(result);
                $("#dialog-form").dialog("open");
            }
        });
    }

}

function Flooding() {
    var hazardSelected = $("input[@name=Hazards]:checked").val();
    var buildid = $('#SelectedRowList').text();
    if (hazardSelected == 'Flooding' && buildid > 0) {
        var year = $('#FloodingYear').val();
        $.ajax({
            url: "/Home/Flooding",
            type: "POST",
            data: { buildId: buildid, year: year },
            dataType: "html",
            success: function(result) {
                $("#dialog-form").html(result);
                $("#dialog-form").dialog("open");
            }
        });
    }
}

function Mudflow() {
    var hazardSelected = $("input[@name=Hazards]:checked").val();
    var buildid = $('#SelectedRowList').text();
    if (hazardSelected == 'Mudflow' && buildid > 0) {
        $.ajax({
            url: "/Home/Mudflow",
            type: "POST",
            data: { buildId: buildid },
            dataType: "html",
            success: function(result) {
                $("#dialog-form").html(result);
                \$("#dialog-form").dialog("open");
            }
        });
    }
}
```

B.5 THE EXECUTIVE CONTROLLER

Listing B.5: The Executive Controller to respond to analysis requests

```

using System;
using System.Collections.Generic;
```

```
4   using System.Linq;
5   using System.Web;
6   using System.Web.Mvc;
7   using Landslide.Models;
8
9           public ActionResult LandSlide(double buildId)
10          {
11              var landSlide = DAL.GetLandSlide(buildId);
12              return this.PartialView("LandSlidePV", landSlide);
13          }
14
15          public ActionResult Flooding(double buildId, int year)
16          {
17              if (year == 20)
18              {
19                  var flooding20 = DAL.GetFlooding20(buildId);
20                  return this.PartialView("Flooding20PV", flooding20);
21              }
22              else
23              {
24                  var flooding100 = DAL.GetFlooding100(buildId);
25                  return this.PartialView("Flooding100PV", flooding100);
26              }
27          }
28
29          public ActionResult Mudflow(double buildId)
30          {
31              var mudFlow = DAL.GetMudflow(buildId);
32              return this.PartialView("MudFlowPV", mudFlow);
33          }
34      }
35  }
```

B.6 EVENT-HANDLING FUNCTION

Listing B.6: Complete Source Code for Selection Change Function

```
1
2   //Register to selectionChanged event to hanlde events
3   selectionChanged: function() {
4       //Get all selected rows
5       var $selectedRows = $('#BuildingTableContainer').jtable('selectedRows');
6
7       $('#SelectedRowList').empty();
8       if ($selectedRows.length > 0) {
9           //Show selected rows
10          $selectedRows.each(function() {
11              var record = $(this).data('record');
12              $('#SelectedRowList').append(record.Build_ID);
13          });
14      } else {
15          //No rows selected
16          $('#SelectedRowList').append('No row selected! Select rows to see here...');
17        }
18   }
```

Appendix C
Creation Of the Views

C.1 CREATION OF THE THREE MAIN VIEWS

Listing C.1: Main views Creation

```
1
2  <%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits="System.Web.
       Mvc.ViewPage" %>
3
4  <asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
5      Home Page
6  </asp:Content>
7  <asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
8      <div id="riskView" style="width: 320px; border:double;background-color: #CCC;height:
           400px">
9          <h5 style="height: 3px;margin-left: 2px;margin-top: 1px;color: green">HAZARD
               COMPUTE VIEW</h5>
10         <br/><br />
11             <div class="inputgroup">
12                 <h5 style="height: 3px;margin-left: 20px;margin-top: 1px;color: brown">
                       Select from the following hazard to compute building vulnerability
                       value:</h5><br/><br/>
13                     <%:Html.RadioButton("Hazards","Landslide",true)%>
14                     <label style="margin-left: 5px;color: green">Landslide</label>
15                     <button id="btnLandslide" style="margin-left: 144px" onclick="
                           LandSlide();return false;">Compute</button>
16                     <br/><br/><br/>
17                     <%:Html.RadioButton("Hazards","Flooding")%>
18                     <label style="margin-left: 5px;color: green">Flooding</label>
19                     <button id="btnFlooding" style="margin-left: 150px" onclick="
                           Flooding();return false;">Compute</button>
20                     <br/><br/>
21                     <label style="margin-left: 45px;color: darkblue">Select Flooding
                           Dataset</label>
22                     <select id="FloodingYear">
23                         <option value="20">20 Years</option>
24                         <option value="100">100 Years</option>
25                     </select>
26                     <br/><br/>
27                     <%:Html.RadioButton("Hazards","Mudflow")%>
28                     <label style="margin-left: 5px;color: green">Mudflow</label>
29                     <button id="btnMudflow" style="margin-left: 150px" onclick="Mudflow
                           ();return false;">Compute</button>
30                 </div>
31                 <br/><br/><br/><br/>
32             </div>
33             <div id="dialog-form" style="background-color: gainsboro"></div>
34             <div id="mapView" style="width: 800px; border:double; height: 400px;
                   background-color: #CCC;margin-top:-403px">
35
36         <div id="map" style="width: 790px; height: 360px;margin-left: 5px"></div>
37     </div>
38     <br/>
39     <div id="existingSlideView" style="width: 1155px;border:double; height: 450px;
           background-color: #CCC">
```

```
40        <h5 style="height: 3px;margin-left: 2px;margin-top: 1px;color: green">BUILDINGS
              INFORMATION VIEW</h5>
41       <div id="BuildingTableContainer" style="width: 1153px; color: black"></div>
42      <div id="SelectedRowList">0</div>
43     </div>
44   </asp:Content>
```

C.2 CREATION OF THE GRID VIEW

Listing C.2: Complete Source Code for Creation of the Grid view

```
1
2   <script type="text/javascript">
3       $(document).ready(function () {
4
5           $('#BuildingTableContainer').jtable({
6               paging: true,
7               sorting: true,
8               defaultSorting: 'Build_ID ASC',
9               selecting: true,
10              multiselect: false,
11              selectingCheckboxes: true,
12
13              actions: {
14                  listAction: '/Home/BuildingList'
15
16              },
17              fields: {
18                  Build_ID: {
19                      key: true,
20                      list: true,
21                      width: '85px',
22                      title: 'Building ID',
23                      type: 'float'
24                  },
25                  Materials: {
26                      title: 'Materials',
27                      width: '350px'
28                  },
29                  Floors: {
30                      title: 'Floors',
31                      width: '120px',
32                      type: 'float'
33                  },
34                  Occupancy: {
35                      title: 'Occupancy',
36                      width: '250px'
37                  },
38                  People: {
39                      title: 'People',
40                      width: '140px',
41                      type: 'float'
42                  },
43                  Area: {
44                      title: 'Area',
45                      width: '140px',
46                      type: 'float'
47                  },
48                  Value: {
49                      title: 'Value',
50                      width: '200px',
51                      type: 'double'
52                  },
53
54          $('#BuildingTableContainer').jtable('load');
55      });
56        </script>
```

C.3 CREATION OF THE RESULTS VIEW

Listing C.3: Dialog Form for Creation of Results View

```
1
2  $("#dialog-form").dialog({
3        title: "Results",
4        autoOpen: false,
5        height: 510,
6        width: 400,
7        modal: true,
8        buttons: {
9          Ok: function() {
10           $(this).dialog("close");
11         }
12       }
13 });
```

Listing C.4: Landslide Results view

```
1
2  <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<Landslide.Models.
       LandslideModels>" %>
3  <div>
4         <fieldset>
5             <div class="editor-label">
6                     <%: Html.Label("Building ID") %>
7             </div>
8             <div class="editor-field">
9                 <%: Html.TextBoxFor(m => m.Build_ID) %>
10                <%: Html.ValidationMessageFor(m => m.Build_ID) %>
11            </div>
12
13            <div class="editor-label">
14                <%: Html.Label("Landslide runout")%>
15            </div>
16            <div class="editor-field">
17                <%: Html.TextBoxFor(m => m.Landslide_runout) %>
18                <%: Html.ValidationMessageFor(m => m.Landslide_runout) %>
19            </div>
20
21            <div class="editor-label">
22                <%: Html.Label("Landslide maximum distance")%>
23            </div>
24            <div class="editor-field">
25                <%: Html.TextBoxFor(m => m.Landslide_distance_max) %>
26                <%: Html.ValidationMessageFor(m => m.Landslide_distance_max) %>
27            </div>
28
29            <div class="editor-label">
30                <%: Html.Label("Landslide Distance beyond the building")%>
31            </div>
32            <div class="editor-field">
33                <%: Html.TextBoxFor(m => m.Landslide_distance_beyond) %>
34                <%: Html.ValidationMessageFor(m => m.Landslide_distance_beyond) %>
35            </div>
36
37            <div class="editor-label">
38                <%: Html.Label("Building Vulnerability Value") %>
39            </div>
40            <div class="editor-field">
41                <%: Html.TextBoxFor(m => m.Building_Value)%>
42                <%: Html.ValidationMessageFor(m => m.Building_Value) %>
43            </div>
44        </fieldset>
45
46      </div>
```

Listing C.5: Mudflows Results View

```
1
2  <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<Landslide.Models.
       MudflowModels>" %>
3  <div>
4          <fieldset>
5
6                  <div class="editor-label">
7                      <%: Html.Label("Building ID") %>
8                  </div>
9                  <div class="editor-field">
10                     <%: Html.TextBoxFor(m => m.Build_ID) %>
11                     <%: Html.ValidationMessageFor(m => m.Build_ID) %>
12                 </div>
13
14                 <div class="editor-label">
15                     <%: Html.LabelFor(m => m.Height) %>
16                 </div>
17                 <div class="editor-field">
18                     <%: Html.TextBoxFor(m => m.Height) %>
19                     <%: Html.ValidationMessageFor(m => m.Height) %>
20                 </div>
21
22                 <div class="editor-label">
23                     <%: Html.LabelFor(m => m.Velocity) %>
24                 </div>
25                 <div class="editor-field">
26                     <%: Html.TextBoxFor(m => m.Velocity) %>
27                     <%: Html.ValidationMessageFor(m => m.Velocity) %>
28                 </div>
29
30                 <div class="editor-label">
31                     <%: Html.Label("Building Vulnerability Value") %>
32                 </div>
33                 <div class="editor-field">
34                     <%: Html.TextBoxFor(m => m.Building_Value) %>
35                     <%: Html.ValidationMessageFor(m => m.Building_Value) %>
36                 </div>
37
38         </fieldset>
39 </div>
```

Listing C.6: Flooding 20 Years Results view

```
1
2
3  <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<Landslide.Models.
       Flooding20Models>" %>
4  <div>
5          <fieldset>
6
7                  <div class="editor-label">
8                      <%: Html.Label("Building ID") %>
9                  </div>
10                 <div class="editor-field">
11                     <%: Html.TextBoxFor(m => m.Build_ID) %>
12                     <%: Html.ValidationMessageFor(m => m.Build_ID) %>
13                 </div>
14
15                 <div class="editor-label">
16                     <%: Html.Label("Velocity 20 Yrs") %>
17                 </div>
18                 <div class="editor-field">
19                     <%: Html.TextBoxFor(m => m.Height_20) %>
```

```
                <%: Html.ValidationMessageFor(m => m.Height_20) %>
            </div>

            <div class="editor-label">
                <%: Html.Label("Depth 20 Yrs") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Velocity_20) %>
                <%: Html.ValidationMessageFor(m => m.Velocity_20) %>
            </div>

            <div class="editor-label">
                <%: Html.Label("Building Vulnerability Value") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Building_Value)%>
                <%: Html.ValidationMessageFor(m => m.Building_Value) %>
            </div>

        </fieldset>
</div>
```

Listing C.7: Flooding 100 Years Results view

```

<%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<Landslide.Models.
    Flooding100Models>" %>
<div>
        <fieldset>

            <div class="editor-label">
                <%: Html.Label("Building ID") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Build_ID) %>
                <%: Html.ValidationMessageFor(m => m.Build_ID) %>
            </div>

            <div class="editor-label">
                <%: Html.Label("Velocity 100 Yrs") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Height_100) %>
                <%: Html.ValidationMessageFor(m => m.Velocity_100) %>
            </div>

            <div class="editor-label">
                <%: Html.Label("Depth 100 Yrs") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Velocity_100) %>
                <%: Html.ValidationMessageFor(m => m.Velocity_100) %>
            </div>

            <div class="editor-label">
                <%: Html.Label("Building Vulnerability Value") %>
            </div>
            <div class="editor-field">
                <%: Html.TextBoxFor(m => m.Building_Value)%>
                <%: Html.ValidationMessageFor(m => m.Building_Value) %>
            </div>
        </fieldset>

</div>
```

Appendix D
Implementation start-ups

D.1 STARTING POINT FOR IMPLEMENTATION OF APPLICATION HOME PAGE

Listing D.1: Master Set-up for Application Home page

```
1
2  <%@ Master Language="C#" Inherits="System.Web.Mvc.ViewMasterPage" %>
3
4   <!DOCTYPE html>
5  <html>
6  <head runat="server">
7      <title><asp:ContentPlaceHolder ID="TitleContent" runat="server" /></title>
8      <!-- Include one of jTable styles. -->
9      <link href="../../Content/Site.css" rel="stylesheet" type="text/css" />
10
11     <!-- Include jTable script file. -->
12     <script src="../Scripts/jquery-1.7.1.min.js" type="text/javascript"></script>
13     <script src="../Scripts/jquery-ui-1.8.11.min.js" type="text/javascript"></script>
14     <script src="../Scripts/jquery.jtable.min.js" type="text/javascript"></script>
15
16     <link href="../../Content/jquery-ui.css" rel="stylesheet" type="text/css" />
17     <link href="../../Content/jquery.ui.dialog.css" rel="stylesheet" type="text/css" />
18     <link href="../Content/themes/metro/darkgray/jtable.min.css" rel="stylesheet" type="
           text/css" />
19     <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
20     <script src="https://maps.googleapis.com/maps/api/js?key=
           AIzaSyAsze9sFkb3VivhO3NXHoNQQ4_2ifnGIyA&sensor=false" type="text/javascript"></
           script>
21     <script type="text/javascript">
22
23     </script>
24      <style type="text/css">
25         #mapView,#existingSlideView,#riskView
26         {
27           border: 2px solid green;
28         }
29         #riskView
30         {
31             margin-left: 830px;
32         }
33     </style>
34 </head>
35 <body>
36     <div class="page">
37         <div id="header">
38             <div id="title">
39                 <h1>Quantitative multi?hazard risk assessment Cyber-app</h1>
40             </div>
41             <div id="logindisplay">
42                 <br/>
43                 <%--<% Html.RenderPartial("LogOnUserControl"); %>--%>
44             </div>
45             <div id="menucontainer">
46                 <ul id="menu">
47                     <li><%: Html.ActionLink("Home", "Index", "Home")%></li>
```

```
48                          <%—<li><%: Html.ActionLink("Register", "Register", "Home")%></li
                                >—%>
49                      </ul>
50                  </div>
51              </div>
52              <div id="main">
53                  <asp:ContentPlaceHolder ID="MainContent" runat="server" />
54                  <div id="footer">
55                  </div>
56              </div>
57          </div>
58      </body>
59      </html>
```

Appendix E
Additionals

E.1 TECHNOLOGICAL ARCHITECTURE BASED ON ASP.NET MVC FRAMEWORK

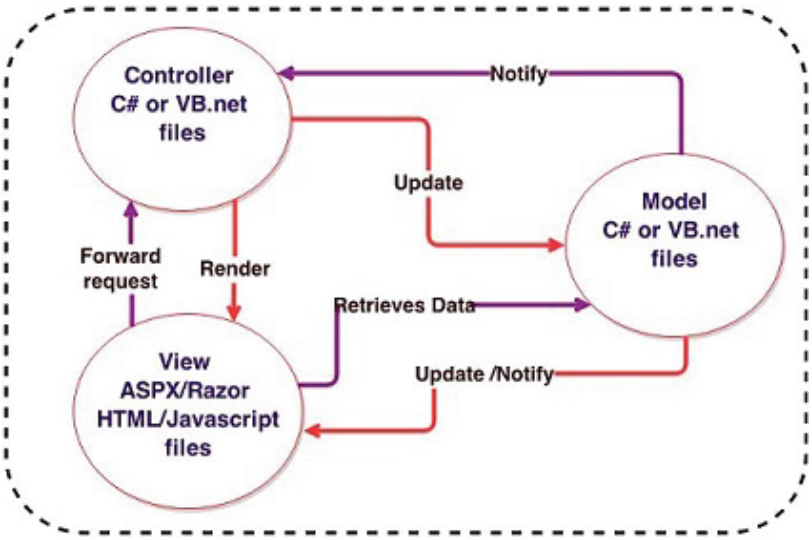


Figure E.1: Diagram of Technologies Used

E.2 SETS OF VULNERABILITY VALUE SCALES

Flood		Mudfows	
h ≥ 1 and v ≥ 5	0.1	h ≥ 1 and v ≥ 7	1
h ≥ 1 and 1≤ v < 5	0.05	h ≥ 1 and 3 ≤ v < 7	0.8
0.5 ≤ h < 1 and v ≥ 5	0.05	0.5 ≤ h < 1 and v ≥ 7	0.8
h ≥ 1 and v < 1	0.025	h ≥ 1 and v < 3	0.4
0.5 ≤ h < 1 and 1 ≤ v < 5	0.025	0.5 ≤ h < 1 and 3 ≤ v < 7	0.4
h < 0.5 and v ≥ 5	0.025	h < 0.5 and v ≥ 7	0.4
0.5 ≤ h < 1 and v < 1	0.01	0.5 ≤ h < 1 and v < 3	0.2
h < 0.5 and 1 ≤ v < 5	0.01	h < 0.5 and 3 ≤ v < 7 0.4	0.4
h < 0.5 and v < 1	0.005	h < 0.5 and v < 3	0.05

Figure E.2: Sets of values [5]

E.3 PIM MODELS FOR THE USE CASE

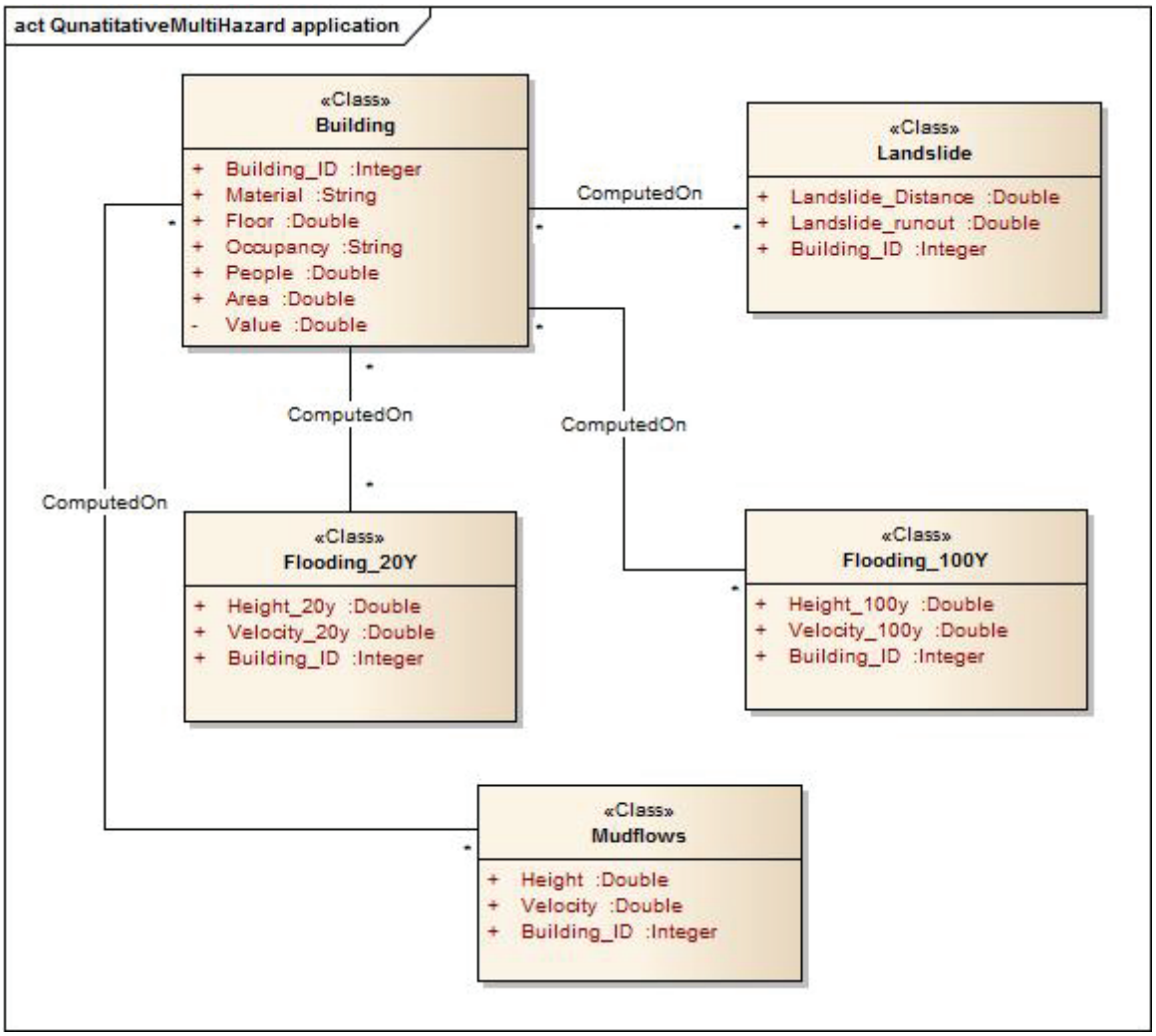


Figure E.3: PIM