Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Diploma Thesis

Information System Design in UML

Biniyam Hundesa Erana

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Biniyam Hundesa Erana

Informatics

Thesis title

Information System Design in UML

Objectives of thesis

The aim is to design a particular Information System application using UML and to develop a prototype web application using C#. In order to ascertain this, there are two phases that should be followed. The first phase contains a definition of an Information System with its component, scaled research on different UML modeling standards and defining various system design concepts. The second phase is building UML model for adventure camping management system which will be used as a typical application area of an Information System. A web-based prototype application will be implemented using C# programming language. Finally, a general analysis will be carried out on the advantages and limitations of the UML designs.

Methodology

In achieving the objectives of the study, a literature review on the UML modeling will be carried out and which also include the current development on such modeling procedures. Design maintainable and understandable UML model for adventure camping management system by formulating the necessary requirements. A prototype application will be Implemented for the given application area using an object-oriented programming language called C#. Finally, discussions, comparisons, and new creative ideas will be noted for future research purposes.

The proposed extent of the thesis

60 – 80 pages

Keywords

Unified Modeling Language(UML), Adventure Camping Management System, Web Application, Object-Oriented Design

Recommended information sources

Arlow, J. and Neustadt, I. (2005). UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd ed., Boston, MA: Addison Wesley.

Bennett, S., McRobb, S. and Farmer, R. (2005). Object-Oriented Systems Analysis And Design: Using UML, 3th ed., New York, NY: McGraw-Hill.

Rosenberg, D. and Stephens, M. (2007) Use Case Driven Object Modeling with UML: Theory and Practice, New York, NY: Apress.

Rumpe, B. (2016). Modeling with UML: Language, Concepts and Methods, Berlin, Germany: Springer.

Vrana, I. (2016). Projecting of Information Systems with UML, Prague, Czechia: Faculty of Economics & Management, CULS Prague.

Expected date of thesis defence

2017/18 SS – FEM

The Diploma Thesis Supervisor

doc. RNDr. Dana Klimešová, CSc.

Supervising department

Department of Information Engineering

Electronic approval: 1. 12. 2017	Electronic approval: 1. 12. 2017
Ing. Martin Pelikán, Ph.D.	**Ing. Martin Pelikán, Ph.D.**
Head of department	Dean

Prague on 14. 03. 2018

Declaration

I declare that I have worked on my diploma thesis titled "System Information Design in UML" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any third person.

In Prague 2018

Biniyam Hundesa Erana

Acknowledgment

I would like to thank my thesis supervisor doc. RNDr. Dana Klimešová, CSc. of the Faculty of Economics and Management at Czech University of Life Sciences Prague for the deep support of my study and research. Her support and guidance in the right direction helped me to construct this thesis.

Besides my advisor, I would like to thank my parents, my teachers and my fellow classmates for providing me with unfailing support and continuous encouragement.

I express my sincere gratitude to the Almighty God for showering me with all the blessings.

Information System Design in UML

Návrh Informačního Systému v UML

Souhrn

Informační technologie (IT) a informační systémy (IS) jsou páteřní strukturou pro všechny malé a velké organizace. IT se týká zejména technologie, především hardwaru, softwaru a telekomunikací, včetně zařízení. Informační systémy (IS) jsou způsoby, kterými lidé v organizaci stále více využívají technologie, shromažďují, zpracovávají, ukládají a používají informace. Organizace musí navrhnout a modelovat informační systém pomocí standardního modelovacího nástroje, známého jako Unified Modeling Language (UML). Tento dokument definuje základní terminologie, používané metodiky a postupy implementace. Současná aktualizovaná verze UML zpracovávala detailní strukturu a pracovní tok informačního systému. Teoretická část pokrývá široké spektrum spektra klíčových konceptů informačního systému, designu a UML. Aplikace Adventure Camping Management System používána jako případová studie, aby demonstrovat jasný obraz a viditelnost teoretických aspektů. Adventure Camping Management System je webová aplikace implementovaná pro podporu probíhající každodenní činnosti a informační tok hostitelské firmy pro řízení kempu. Tato hostitelská společnost má k dispozici několik kempů v různých lokalitách po celém světě. Primární činnost Adventure Camping je prodej balíčku kempování pro veřejnost. Typický sváteční balíček pro táboření se skládá z jednoho nebo více stanu a několika základních položek pro vyhrazené časové období. Tři analytické modely, které jsou třídním modelem, interakčním modelem a stavovým modelem používané s jejich příslušnými UML diagramy pro návrh systému. Prováděná prototypová webová aplikace, která jasně ukazuje vliv modelování ve vývoji. Diskutovány jsou také osvědčené postupy, Cloud Computing a aktuální technologický vývoj pro používání webových aplikací a UML.

Klíčová slova: UML, webová aplikace, Systém řízení kempu, objektově orientovaný design, Cloud Computing

Summary

Information Technology (IT) and Information Systems (IS) are the backbones for all small to high-level organizations. IT refers particularly to the technology, primarily the hardware, software, and telecommunications, including devices. Information systems (IS) are the way by which people in an organization, increasingly utilizing technology, gather, process, store, and use information. An organization needs to design and model an information system using a standard modeling tool, known as Unified Modeling Language (UML). This document defines the basic terminologies, the methodologies that are being used, and the implementation procedures. The current released version of UML used to elaborate the detailed structure and workflow of an information system. The theoretical part covers a wide range of spectrum to the key concepts of an information system, design, and UML. Adventure Camping Management System used as a case study to see the clear picture and visibility of the theoretical aspects. Adventure Camping Management System is a web-based application implemented to support the ongoing day-to-day activities and information flow of adventure camping host company. The primary business of Adventure Camping is selling camping holiday packages to the general public. The three analysis models which are the class model, the interaction model and the state model used with their respective UML diagrams to design the system. A prototype web application implemented to show clearly the effect of modeling in the development process. Best practice, cloud computing and current technological developments for using web-based applications and UML are discussed as well.

Keywords: UML, Web-based Application, Adventure Camping Management System, Object-Oriented Design, Cloud Computing

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The recent development of the Internet and the World Wide Web makes it appear that the world is moving towards to completely a new technology. In fact, the Web - now considered to be a major activator of the way people accesses and views information - is the result of numerous projects that have been carried out over the last 40 years.

Information systems have become the cornerstone for most of the organizations. Banks could not process transactions, governments could not receive taxes, hospitals could not follow and provide treatment to patients without the support of information systems. An information system is a collection of several components that interact with each other to produce information. Information system design is often viewed as a stage in the system development life-cycle, concerned with the detailed "laying out" of system software.

The Unified Modeling Language (UML) is a standard modeling language that facilitate communication and interaction between all participants in the development process. The UML is appropriate and sufficient for modeling of an enterprise information system. The current version of UML which is 2.5 used in this document to model a particular real-world enterprise information system.

Adventure camping is a supervised program for children, teenagers, and/or families conducted during vacation periods in some countries. Adventure Camping Management System is a web-based application implemented to support the ongoing day-to-day activities and information flow of adventure camping host company. This host company has multiple campgrounds available in a different location around the world. The primary business of Adventure Camping is selling camping holiday packages to the general public. A typical camping holiday package consists of one or more tents and few essential items for a reserved period of time.

This document is about modeling Information system using UML. The document also investigating the role of UML in the software modeling and maintenance, investigate the reported research and provide different aspects of UML model inconsistencies. An adventure camping management system will be considered as a case study to formulate all the necessary topics mentioned here.

2. Objectives and Methodology

Objectives of thesis: The aim is to design a particular Information System application using UML and to develop a prototype web application using C#. In order to ascertain this, there are two phases that should be followed. The first phase contains a definition of an Information System with its component, scaled research on different UML modeling standards and defining various system design concepts. The second phase is building UML model for adventure camping management system which will be used as a typical application area of an Information System. A web-based prototype application will be implemented using C# programming language. Finally, a general analysis will be carried out on the advantages and limitations of the UML designs.

Methodology: In achieving the objectives of the study, a literature review on the UML modeling will be carried out and which also include the current development of such modeling procedures. Design maintainable and understandable UML model for adventure camping management system by formulating the necessary requirements. A prototype application will be Implemented for the given application area using an object-oriented programming language called C#. Finally, discussions, comparisons, and new creative ideas will be noted for future research purposes.

3. Theoretical Background

3.1 Information Systems

Why study information systems? Why do businesses need information technology? Information systems have become as integrated into our daily business activities as accounting, finance, operations management, marketing, human resource management, or any other major business function. Information systems and technologies are vital components of successful businesses and organizations some would say they are business imperatives [26].

Organizations offer products to customers to make money. These products can be goods or services. In most organizations, huge volumes of data accumulate data of products, data of customers, data of employees, data of the delivery of products, and data from other sources. These data, therefore, play an important role in contemporary organizations and must be stored, managed, and processed, which is where information systems come into play [24].

An information system (IS) can be any organized combination of people, hardware, software, communications networks, data resources, and policies and procedures that stores, retrieves, transforms, and disseminates information in an organization. People rely on modern information systems to communicate with one another using a variety of physical devices (hardware), information processing instructions and procedures (software), communications channels (networks), and stored data (data resources) [26].

Information systems (IS) existed in organizations long before the advent of information technology (IT) and, even today, there is still many ‘systems’ present in organizations with technology nowhere in sight. IT refers specifically to technology, essentially hardware, software, and telecommunications networks, including devices of all kinds: computers, servers, routers, and all types of software: operating systems and personal productivity tools. IT facilitates the acquisition and collection, processing, storing, delivery, sharing, and presentation of information and other digital contents.

Information systems (IS) are the means by which people and organizations, increasingly utilizing technology, gather, process, store, use and disseminate information. Some information systems are totally automated by IT. For example, airlines, comparison websites, banks and some public agencies have systems where no human intervention is required.

Achieving organizational change on any scale can be difficult, even without the introduction of new technology. Another term that is frequently used along with IS and IT is an application, i.e. an application of IT to handle information in some way. Essentially, an application refers to software, or a combination of software and hardware, used to address or enable a business or personal activity. These applications can be purchased, pre-written software programs for a particular business activity or developed ‘in-house’ to provide particular functionality. Many applications for personal productivity as well as business use are now delivered via mobile devices of all kinds and increasingly they are being provisioned from the cloud [29].

3.2 Object-Oriented Systems Development

Object-oriented programming (OOP) is an architype based on "objects". Objects may contain data. Data can be represented in the form of fields, commonly known as attributes; and some action code can also be represented in the form of procedures, commonly known as methods. Object-oriented programming (OOP) refers to a particular type of computer programming in which developers define not only the data type of a data structure but also the types of operations (functions) that can be applied to the data structure. In addition, developers can create relationships between one object and another.

What is object-oriented systems development? By looking the way object-oriented analysis and design is learned and practiced in some places, it can be generalized as many professionals simply adopt an object-oriented programming language or use fragments of an object-oriented based development. It is not sufficient to organize the system architecture in tiers and modules if the code implemented inside of it is disorganized. Some programmers organize the system adequately in classes and packages, but they still write spaghetti code inside the methods of these classes and packages. In addition, other developers still use top-down functional decomposition inside methods, which is not appropriate when using object-oriented programming (top-down decomposition is adequate if structured programming is used instead).

In order to build code that is really object-oriented, developers should learn the techniques of delegation and responsibility assignment, which can lead to reusable code and low coupling. It is useless to invest heavily in object-oriented CASE (computer-aided software engineering) tools without learning the way to think in terms of object-oriented programming [5, 39].

Object-Oriented development has the following phases [36]:

- **Object-Oriented Analysis (OOA):** is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.

  The primary tasks in the object-oriented analysis (OOA) are:
  - Identifying objects
  - Organizing the objects by creating object model diagram
  - Defining the internals of the objects, or object attributes
  - Defining the behavior of the objects, i.e., object actions
  - Describing how the objects interact

  The common models used in OOA are use cases and object models.

- **Object-Oriented Design (OOD):** involves the implementation of the conceptual model produced during object-oriented analysis.

  The implementation details generally include:
  - Restructuring the class data (if necessary),
  - Implementation of methods, i.e., internal data structures and algorithms,
  - Implementation of control, and
  - Implementation of associations.

Grady Booch has defined object-oriented design as "*a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design*".

- **Object-Oriented Programming (OOP):** is a programming paradigm based upon objects (having both data and methods) that aim to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

The important features of object-oriented programming are:

- Bottom-up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object’s data
- The interaction between objects through functions
- Reusability of design through the creation of new classes by adding features to existing classes

Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Grady Booch has defined object-oriented programming as “*a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships*”.

3.3 What is Unified Modeling Language (UML)?

World development and lifestyle become more dependent on software. Software-intensive systems, as technological progress, as well as social interaction, are growing in size, distribution, complexity, and importance. However, advancement of these systems changes the potential of software industry and information system flow. Various software development projects come short before completion in different ways but they all share common symptoms. Some of them are inaccurate or incomplete end-user preferences; inability to deal with modification of the requirements. Some of the problems and their major causes can be avoided by designing more rigorous development process. Deployment of notation language, like UML, usually facilitate communication and interaction between all participants in the development process.

UML is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system. The UML is appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real-time embedded systems. It is a very expressive language, addressing all the views needed to develop and then deploy such system [6].

UML is the successor to the wave of object-oriented analysis and design (OOA&D) methods that appeared in the late '80s and early '90s. It most directly unifies the methods of Booch, Rumbaugh (OMT), and Jacobson, but its reach is wider than that. The UML went through a standardization process with the OMG (Object Management Group) and is now an OMG standard.

The UML is called a modeling language, not a method. Most methods consist, at least in principle, of both a modeling language and a process. The modeling language is the (mainly graphical) notation that methods use to express designs. The process is their advice on what steps to take in doing a design [13].

System design on any reasonably large scale is difficult. Anything from a simple desktop application to a full multi-tier enterprise-scale system can be made up of hundreds and potentially thousands of software and hardware components. In system design, you model for one important reason: to manage complexity. Modeling helps you see the forest for the tree, allowing you to focus on, capture, document, and communicate the important aspects of your systems design.

A model is an abstraction of the real-world thing. When you model a system, you abstract away any details that are irrelevant or potentially confusing. Your model is a simplification of the real system, so it allows the design and viability of the system to be understood, evaluated and criticized quicker than if you had to dig through the actual system itself [15, 32].

What is Modeling Language?

A modeling language can be made up of pseudo-code, actual code, pictures, diagrams, or long passages of description; in fact, it is pretty much anything that helps you describe the system. A modeling language can be anything that contains a notation (a way of expressing the model) and a description of what that notation means (a- meta-model). But why should you consider using UML when there are so many different ways of modeling, including many you could make up your own?

Every approach to modeling has different advantages and disadvantages, but UML has six main advantages [15]:

- It is a formal language- each element of the language has a defined meaning.
- It is concise- the entire language is made up of simple notation.
- It is comprehensive- It describes all important aspects of a system.
- It is scalable- scale up and scale down where needed.
- It is built on lessons learned- UML is a collection of best practices in the object-oriented community.
- It is a standard- UML is controlled by an open standard group with active contributions from a worldwide group of vendors and academics.

3.3.1 History of UML

Prior to 1994, the OO methods worlds were a bit of a mess. There were several competing visual modeling languages and methodologies all with their strengths and weaknesses and all with their supporters and detractors. In terms of visual modeling languages, the clear leaders were Booch (the Booch Method) and Rumbaugh (Object Modeling Technique or OMT), who between them had over half the market. On the methodologies side, Jacobson had by far the strongest case as although many authors claimed to have a "method", all that many of them actually had was a visual modeling syntax and a collection of more or less useful guidelines [3].

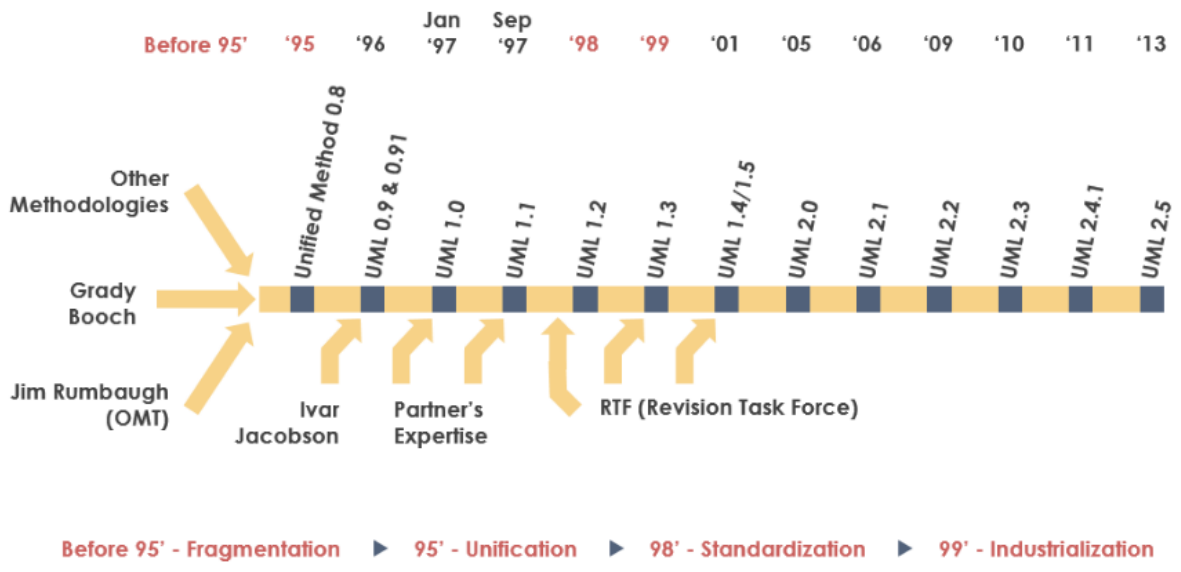


Figure 1: UML version release (Source: [37])

There was a very early attempt at unification in 1994 with Coleman's fusion method. However, although laudable, this attempt did not involve the original authors of the constituent methods (Booch, Jacobson, and Rumbaugh) and was also quite late to market with a book explaining the approach. Fusion was rapidly overtaken by the course of events when, in 1994, Booch and Rumbaugh joined Rational Corporation to work on UML.

In 1996, the Object Management Group (OMG) produced a request for proposal (RFP) for an OO visual modeling language, and UML was submitted. In 1997, OMG accepted the UML and the first open, industry-standard OO visual modeling language was born [3, 13]. OMG released the first standardize modeling language UML v1.1. Since then, UML has gone through several revisions and refinements leading up to the current 2.5 release. Each revision has tried to address problems and shortcomings identified in the previous versions, leading to an interesting expansion and contraction of the language [31].

In 2000, UML 1.4 introduced a significant extension to UML by the addition of action semantics. These describe the behavior of a set of primi-tive actions that may be implemented by specific action languages. Action semantics plus an action language allow the detailed specification of the behavioral elements of UML models (such as class operations) directly in the UML model. This was a very significant development as it made the UML specification computationally complete and thus it became possible to make UML models executable.

UML 2 introduces quite a lot of new visual syntax. Some of this replaces (and clarifies) existing l.x syntax, and some of it is completely new and represents new semantics added to the language. UML has always provided many options about how a particular model element may be displayed, and not all of these will be supported by every modeling tool. Some syntactic options are quite specialized and so we mention them only in passing, if at all.

Although UML 2 makes many syntactic changes to UML compared to UML1.x, the good news is that the fundamental principles remain more or less the same. Modelers who are accustomed to using previous versions of UML should experience an easy transition to UML 2. In fact, the deepest changes introduced in UML 2 have been to the UML metamodel and will not be encountered directly by most modelers. The UML metamodel is a model of the UML language that is itself expressed in a subset of UML. It precisely defines the syntax and semantics of all of the UML modeling elements that you will encounter. These changes to the UML metamodel have largely been about improving the precision and consistency of the UML specification [3, 15].

3.4 UML 2

The first thing to notice about the UML is that there are many different diagrams (models) to get working on it. The reason for this is that there are many dimensions to look at a system from many different viewpoints. A software development will have many stakeholders that take different responsibilities. All of the stakeholders are interested in different aspects of the system, and each of them requires a same or different level of detail.

The first version of UML allowed people to communicate designs unambiguously, convey the essence of a design, and even capture and map functional requirements to their software solutions. However, the world changed more fundamentally with the recognition that systems modeling, rather than just software modeling, could also benefit from a unified language such as UML. The driving factors of component-oriented software development, model-driven architectures, executable UML, and the need to share models between different tools placed demands on UML that it had not originally been designed to meet [15, 34].

Also, UML 1.x and all of its previous revisions were designed as a unified language for humans. When it became important for models to be shared between machines specifically between Computer Aided Systems Engineering (CASE) tools UML 1.x was again found wanting. UML 1.x's underlying notation rules and its meta-model were (ironically) not formally defined enough to enable machine-to-machine sharing of models.

Two reasonably new approaches to system development inspired many of the improvements made in UML 2.0. In a nutshell, Model Driven Architectures (MDAs) provide a framework that supports the development of Platform Independent Models (PIMs) models that capture the system in a generic manner that is divorced from concerns such as implementation language and platform.

PIMs can then be transformed into separate Platform Specific Models (PSMs) that contain concrete specifications for a particular system deployment (containing details such as implementation language and communications protocols, etc.). MDA requires a formally structured and interoperable meta-model to perform its transformations, and this level of meta-model is now provided by UML 2.0.

The designers of UML 2.0 were very careful to ensure that UML 2.0 would not be too unfamiliar to people who were already using UML 1.x. Many of the original diagrams and associated notations have been retained and extended in UML 2.0. However, new diagram types have been introduced to extend the language just enough so that it can support the latest best practices.

With Version 2.0, UML has evolved to support the new challenges that software and system modelers face today. What began many years ago as a unification of the different methods for software design has now grown into a unified modeling language that is ready and suitable to continue to be the standard language for the many of different tasks involved in software and systems design [15].

Table 1:UML diagrams release (Source: [15])

Diagram type	**What can be modeled?**	**Originally introduced by UML**
Use Case	Interactions between your system and users or other external systems. Also helpful in mapping requirements to your systems.	UML 1.x
Activity	Sequential and parallel activities within your system.	UML 1.x
Class	Classes, types, interfaces, and the relationships between them.	UML 1.x
Object	Object instances of the classes defined in class diagrams in configurations that are important to your system.	Informally UML 1.x
Sequence	Interactions between objects where the order of the interactions is important.	UML 1.x
Communication	The ways in which objects interact and the connections that are needed to support that interaction.	Renamed from UML 1.x's collaboration diagrams
Timing	Interactions between objects where timing is an important concern.	UML 2.0

Interaction Overview	Used to collect sequence, communication, and timing diagrams together to capture an important interaction that occurs within your system.	UML 2.0
Composite Structure	The internals of a class or component and can describe class relationships within a given context.	UML 2.0
Component	Important components within your system and the interfaces they use to interact with each other.	UML 1.x, but takes on a new meaning in UML 2.0
Package	The hierarchical organization of groups of classes and components.	UML 2.0
State Machine	The state of an object throughout its lifetime and the events that can change that state.	UML 1.x
Deployment	How your system is finally deployed in a given real-world situation.	UML 1.x

3.4.1 UML Diagrams

Unified Modeling Language specification version 2.5 has many types of diagrams, which are divided into two categories. Some types represent structural information, and the rest represent general types of behavior, including a few that represent different aspects of interactions.

Structure diagrams show the static structure of the objects in a system. That is, they depict those elements in a specification that are irrespective of time. The elements in a structure diagram represent the meaningful concepts of an application and may include abstract, real-world and implementation concepts. For example, a structure diagram for an airline reservation system might include classifiers that represent seat assignment algorithms, tickets, and a credit authorization service. Structure diagrams do not show the details of dynamic behavior, which are illustrated by behavioral diagrams. However, they may show relationships to the behaviors of the classifiers exhibited in the structure diagrams.

Behavior diagrams show the dynamic behavior of the objects in a system, including their methods, collaborations, activities, and state histories. The dynamic behavior of a system can be described as a series of changes to the system over time. Behavior diagrams can be further classified into several other kinds as illustrated in Figure 2.

In UML 2.5 there are 14 types of UML diagrams, which are divided into two categories:

- 7 diagram types represent structural information
- Another 7 represent general UML diagram types for behavioral modeling, including four that represent different aspects of interactions.

The constructs contained in each of the UML diagrams are described in the clauses indicated below [27].

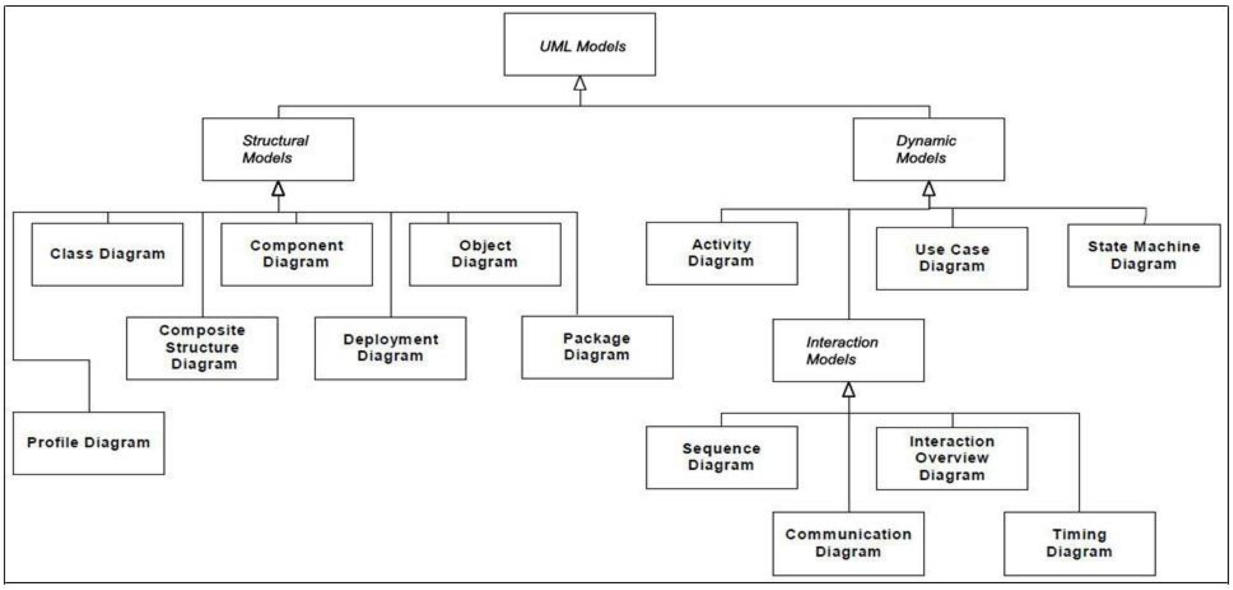


Figure 2: UML diagram classifications (Source: [27])

3.4.2 UML 2.5 Structure Diagrams

Structural diagrams are used to capture the physical organization of the things in your system i.e., how one object relates to another. There are seven types of structure diagram. These are Class Diagram, Component Diagram, Deployment Diagram, Object Diagram, Package Diagram, Composite Structure Diagram, Profile Diagram. Only key diagrams are explained in this document.

Class Diagram

The UML Reference Manual [6] defines a class as "The descriptor for a set of objects that share the same attributes, operations, methods, relation- ships, and behavior." We could summarize this by saying that a class is a descriptor for a set of objects that have the same features. Every object is an instance of exactly one class. Here are some useful ways to think about classes. Think of a class as being a template for objects- a class determines the structure (set of features) of all objects of that class. All objects of the same class must have the same set of operations, the same set of attributes, and the same set of relationships, but may have different attribute values [3].

A class is a specification or template that all objects of that class (instances) must follow. Each object of the class has specific values for the attributes defined by the class and will respond to messages by invoking the operations defined by the class. Depending on their state, different objects may respond to the same message in different ways [3, 6]. The main purpose of class diagram is to model the static aspect of an application. Class diagrams are the only diagrams which can be directly mapped and linked with object-oriented languages and thus widely used at the time of creation. It is the most common and widely used UML diagram in the coder community.

Relationships

Classes in isolation would not provide much insight into how a system is designed. UML provides several ways of representing relationships between classes. Each of UML relationship represents a different type of connection between classes and has subtleties that aren't fully captured in the UML specification. UML defines the following relationships [31]:

1. **Association**

Associations are stronger than dependencies and typically indicate that one class retains a relationship to another class over an extended period of time. The lifelines of two objects linked by associations are probably not tied together (meaning one can be destroyed without necessarily destroying the other). Associations are typically read as "...has a...". For example, if you have a class named Window that has a reference to the current mouse cursor, you would say "Window has a Cursor".

2. **Association Class**

Often the relationship between two elements isn't a simple structural connection. If the association between two elements is complex, you can represent the connection using an association class. An association class is an association that has a name and attributes, like a normal class. You show an association class like a regular class with a dashed line connecting it to the association it represents.

3. Association Qualifiers

Relationships between elements are often keyed, or indexed, by some other value. UML provides association qualifiers to capture such information. A qualifier is typically an attribute of the target element, though this isn't required. You show a qualifier by placing a small rectangle between the association and the source element. Draw the name of the qualifier (usually the name of an attribute) in the rectangle.

4. Aggregation

Aggregation is a stronger version of association. Unlike association, aggregation typically implies ownership and may imply a relationship between lifelines. Aggregations are usually read as "...owns a...". You show an aggregation with a diamond shape next to the owning class and a solid line pointing to the owned class.

5. Composition

Composition represents a very strong relationship between classes, to the point of containment. Composition is used to capture a whole-part relationship. The "part" piece of the relationship can be involved in only one composition relationship at any given time. The lifetime of instances involved in composition relationships is almost always linked; if the larger, owning instance is destroyed, it almost always destroys the part piece. UML does allow the part to be associated with a different owner before destruction, thus preserving its existence, but this is typically an exception rather than the rule. A composition relationship is usually read as "...is part of...", which means you need to read the composition from the part to the whole. You show a composition relationship using a filled diamond next to the owning class and a solid line pointing to the owned class.

6. Generalization/Inheritance

A generalization relationship conveys that the target of the relationship is a general or less specific, version of the source class or interface. Generalization relationships are often used to pull out commonality between difference classifiers. Generalizations are usually read as "...is a...", starting from the more specific class and reading toward the general class. You show a generalization relationship with a solid line with a closed arrow, pointing from the specific class to the general class.

Unlike associations, generalization relationships are typically not named and don't have any kind of multiplicity. UML allows for multiple inheritances, meaning a class can have more than one generalization with each representing an aspect of the decedent class. However, some modern languages (e.g., Java and C#) don't support multiple inheritances; interfaces and interface realization are used instead.

7. Interfaces

An interface is a classifier that has declarations of properties and methods but no implementations. You can use interfaces to group common elements between classifiers and provide a contract a classifier that provides an implementation of an interface must obey. Some modern languages, such as C++, don't support the concept of interfaces; UML interfaces are typically represented as pure abstract classes. Other languages, such as Java, do support interfaces but don't allow them to have properties. The moral is that you should be aware of how your model is going to be implemented when modeling your system. The common representation is the standard UML classifier notation with the stereotype «interface».

8. Dependency

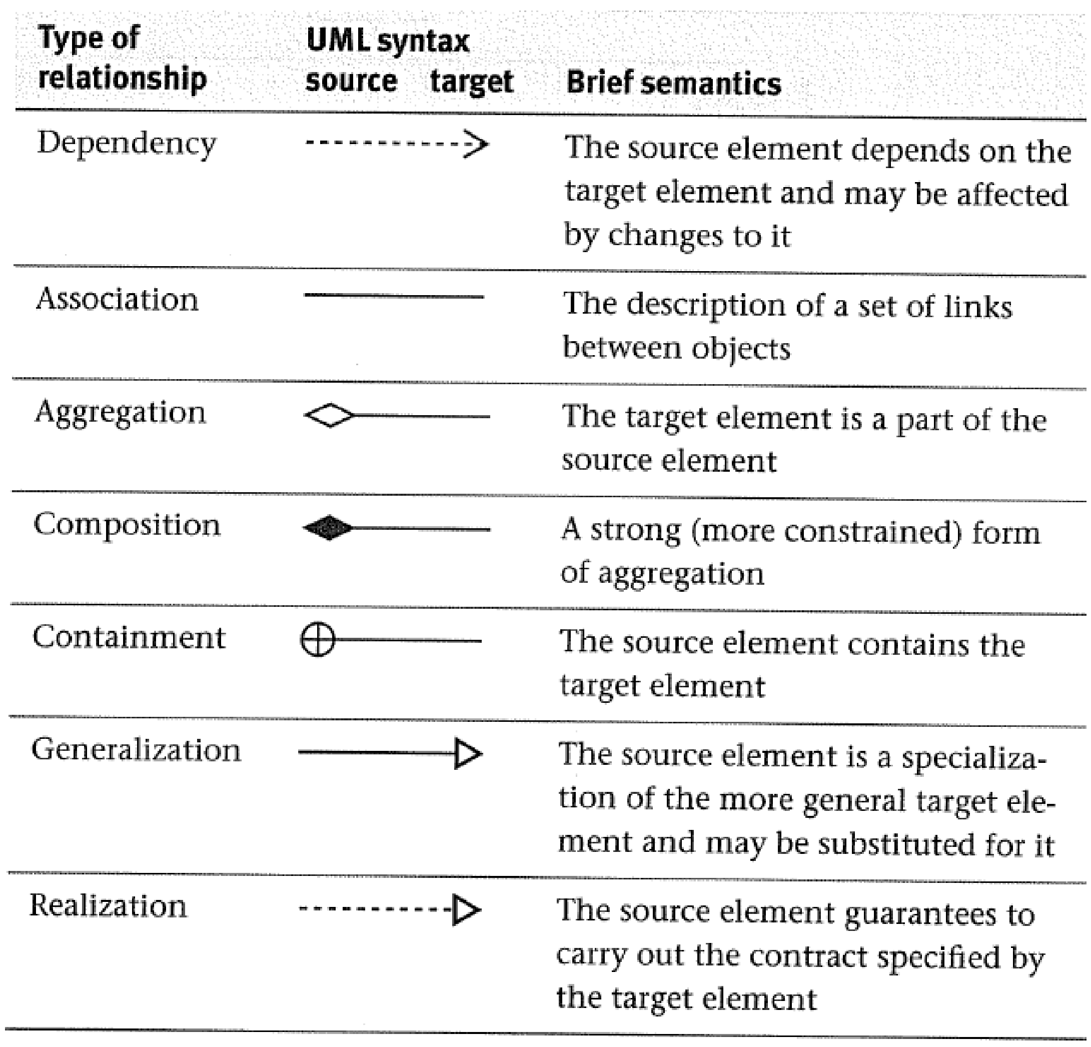
The weakest relationship between classes is a dependency relationship. Dependency between classes means that one class uses, or has knowledge of, another class. It is typically a transient relationship, meaning a dependent class briefly interacts with the target class but typically doesn't retain a relationship with it for any real length of time. Dependencies are typically read as "...uses a...". You show a dependency between classes using a dashed line with an arrow pointing from the dependent class to the class that is used.

9. Multiplicity

Because associations typically represent lasting relationships, they are often used to indicate attributes of a class. You can express how many instances of a particular class are involved in a relationship. If you don't specify a value, a multiplicity of 1 is assumed. To show a different value, simply place the multiplicity specification near the owned class [6, 15, 31].

Table 2: Multiplicity of instances of an entity (Source: [3])

Indicator	Meaning
0..1	Zero or 1
1	Exactly 1
1..1	Exactly 1
0..*	Zero or more
*	Zero or more
1..*	1 or more



Type of relationship	UML syntax source target	Brief semantics
Dependency	- - - - - - - - - ->	The source element depends on the target element and may be affected by changes to it
Association	———	The description of a set of links between objects
Aggregation	◇———	The target element is a part of the source element
Composition	◆———	A strong (more constrained) form of aggregation
Containment	⊕———	The source element contains the target element
Generalization	———▷	The source element is a specialization of the more general target element and may be substituted for it
Realization	- - - - - - - - - ▷	The source element guarantees to carry out the contract specified by the target element

Figure 3: Relationships between classes (Source: [3])

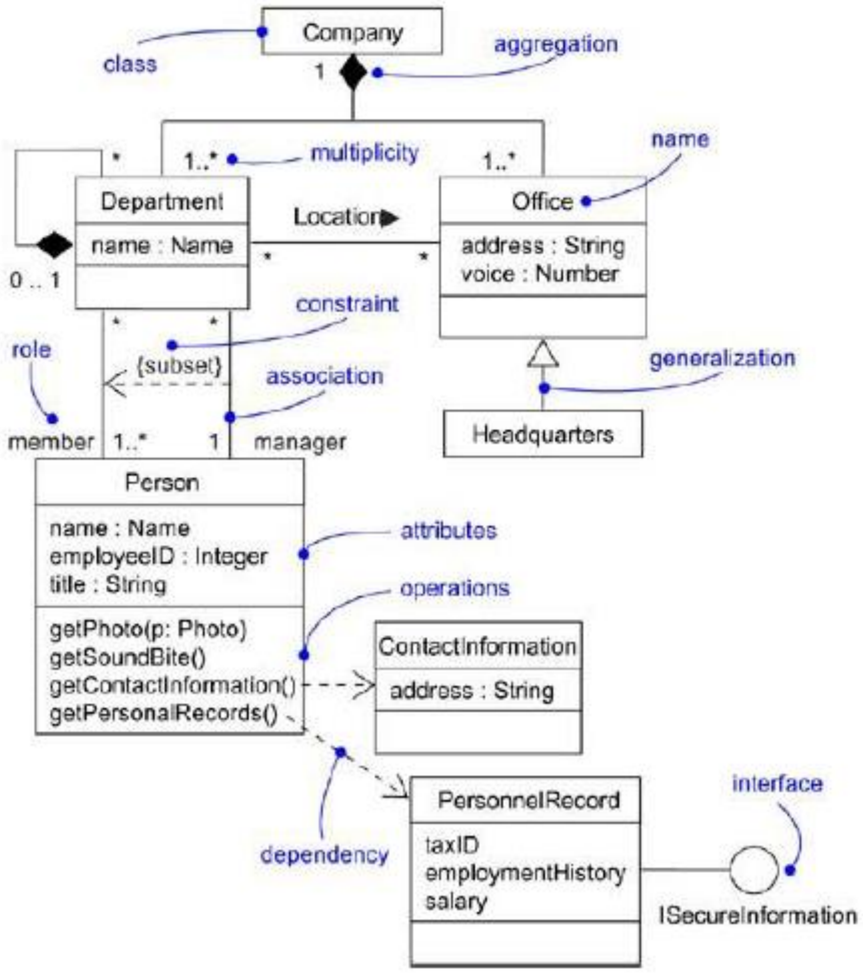


Figure 4: UML class diagram (Source: [6])

3.4.3 UML 2.5 Behavior Diagrams

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are two primary kinds of behavioral things. First, an interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. Second, a state model is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events [6].

There are seven types of behavior diagrams. These are Use Case Diagram, Activity Diagram, State Diagram, Sequence Diagram, Communication Diagram, Interaction Overview Diagram, Timing Diagram. Only key diagrams are explained in this document.

Use Case Diagram

Use cases are a way to capture system functionality and requirements in UML. Use case diagrams consist of named pieces of functionality (use cases), the persons or things invoking the functionality (actors), and possibly the elements responsible for implementing the use cases (subjects). Use cases represent distinct pieces of functionality for a system, a component, or even a class. Each use case must have a name that is typically a few words describing the required functionality. The common representation is an oval with the name of the use case in the center [31].

The behavior of a use case can be described by means of interaction diagrams (sequence and collaboration diagrams), activity charts and state diagrams or by pre-conditions and post-conditions, as well as natural language text, where appropriate. Use case is a high-level description of what the system is supposed to do, whose aim is to capture the system requirements. In other words, if a use case represents a user interaction, many variants of this user interaction can be described [1].

Use Case Relationship

- **Include**

You can factor out common functionality from several use cases by creating a shared, included use case. Unlike in use case extend (discussed next), the use case that includes another use case is typically not complete on its own. The included functionality isn't considered optional; it is factored out simply to allow for reuse in other use cases.

- **Extend**

UML provides the ability to plug in additional functionality to a base use case if specified conditions are met. UML clearly specifies that a base use case should be a complete use case on its own. The extension use cases are smaller in scope and represent additional functionality, so they may not be useful outside the context of the base use case [31].

- **Generalization**

A generalization relationship is a parent-child relationship between use cases. The child use case in the generalization relationship has the underlying business process meaning but is an enhancement of the parent use case.

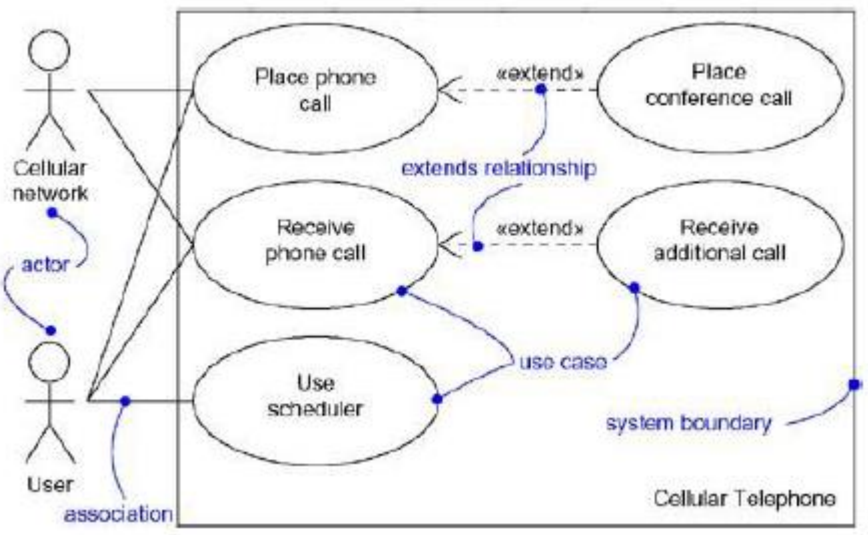


Figure 5: UML use case diagram (Source: [6])

Activity Diagram

Activity diagrams are often called "object-oriented flowcharts". They allow you to model a process as an activity that consists of a collection of nodes connected by edges. An activity can be attached to any modeling element for the purpose of modeling its behavior. The element provides the context for the activity, and the activity may refer to features of its context. Activity diagrams are usually easily understood by stakeholders. This is be- cause most stakeholders will have had some exposure to flowcharts in some form or another. Activity diagrams can, therefore, be a great communication mechanism provided you keep them simple. Activities often start with a single control node, the initial node, that indicates the place where execution will begin when the activity is invoked. One or more final nodes indicate places where the activity terminates. In UML 2, activity diagrams have completely new semantics based on Petri Nets. This has two advantages [3, 15, 33].

- The Petri Net formalism provides greater flexibility in modeling different types of flow.
- There is now a clear distinction in UML between activity diagrams and state machines.

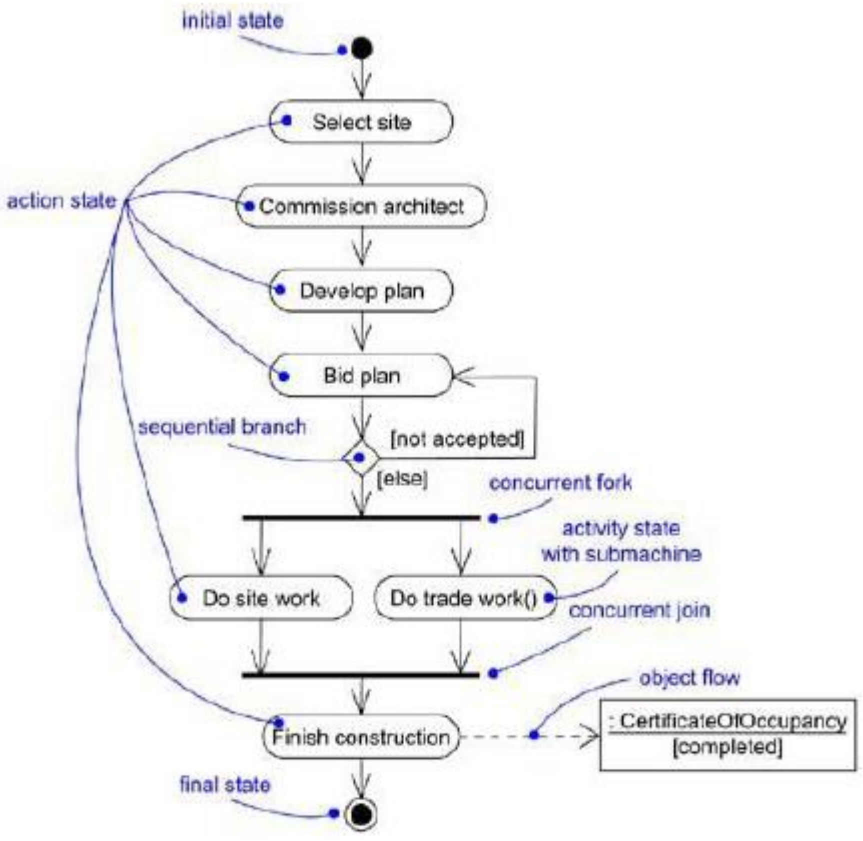


Figure 6: UML activity diagram (Source: [6])

State Diagram

State diagrams capture the behavior of a software system. State diagrams can be used to model the behavior of a class, subsystem, or entire application. A state is shown using the basic rectangle notation, with the name of the state shown in the top compartment. A state can be either active or inactive. A state is considered active as soon as it is entered because of some transition. Similarly, a state is considered inactive immediately after leaving the state. A transition shows the relationship, or path, between two states. It represents the actual change in the configuration of a state as it heads from one state to the next. Each transition can have a guard condition that indicates if the transition can even be considered (enabled), a trigger that causes the transition to execute if it is enabled, and any effect the transition may have when it occurs [31, 33].

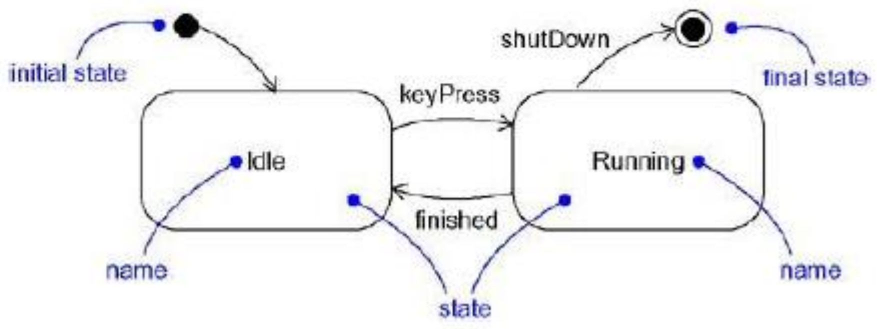


Figure 7: UML state diagram (Source: [6])

Sequence Diagram

Interaction diagrams are defined by UML to emphasize the communication between objects, not the data manipulation associated with that communication. Interaction diagrams focus on specific messages between objects and how these messages come together to realize functionality. While composite structures show what objects fit together to fulfill a particular requirement, interaction diagrams show exactly how those objects will realize it. You can show the details of an interaction using several different notations; however, sequence diagrams are by far the most common.

You show participants in an interaction using a rectangle called a lifeline. The term lifeline illustrates UML's bias toward representing interaction diagrams using the sequence diagram notation. When shown in sequence diagrams, participants have a dashed line dropping down from a rectangle that shows how long the object is actually in existence. Communication between participants can take many different forms: method calls, sending a signal, creating an instance, destroying an object, etc., all of which are collectively called messages. A message specifies the kind of communication, its sender, and its receiver. The most common use of messages is to represent method calls between two objects. When messages are used to indicate a method call, you can show the parameters passed to the method in the message syntax [3, 31].

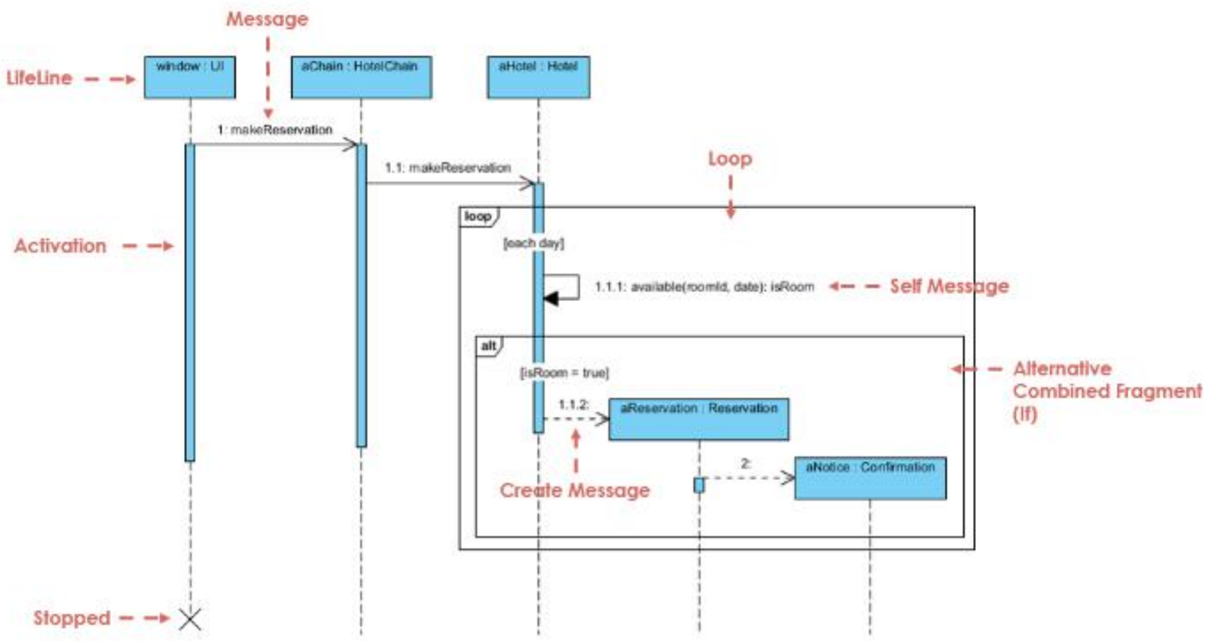


Figure 8: UML sequence diagram (Source: [37])

- A use-case model is built and the actors are connected to use cases. Each use case represents a task in which the actor participates.
- For each use case, a sequence diagram needs to be built. Each sequence diagram specifies the main interaction steps to be achieved for each task (i.e. use case). Some of the interaction steps in a sequence diagram can be deployed in another sequence diagram (or sub-diagram) [1].

3.4.4 Analysis Model

Successful analysis model determines, WHAT should be done, without restrictions on HOW to do it and without implementation decisions. It is vital to understand the real-world system from the problem formulation and to extract its essential properties into a model.

Output of analysis are three models:

- Class model (object model- OM)
- State (dynamic model- DM)
- Interaction model (IM)

All three models discussed in section 3.4.2 and 3.4.3 under structure diagrams and behavior diagrams.

Models offer to view a system from three different viewpoints:

Object Model: expresses a static structure of objects in a system, mainly classes, associations and generalizations (attributes and operations are secondary).

State Model: describes changes of object properties in time and their mutual stimulation. State model is a collection of state diagrams.

Interaction Model: describes cooperation of objects in several levels of abstraction:

- Top: use case
- Middle: sequence diagram
- The most detailed: activity diagram

Each model focuses on a certain aspect and leaves remaining without explanation. All the three models are needed for a full system understanding [38].

3.5 UML and the Software Development Process

When you are using UML to model a software system, the "degree of UML" you apply is partially influenced by the software development process you use. A software development process is a recipe used for constructing software determining the capabilities it has, how it is constructed, who works on what, and the timeframes for all activities. Processes aim to bring discipline and predictability to software development, increasing the chance of success of a project. Since UML is the language for modeling your software, it's an important part of the software development process [15].

Most modern development processes can be considered as agile. Other methodologies include a waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming. Only key methodologies are explained in this document.

3.5.1 Waterfall Development

The waterfall method attempts to pin down the requirements early in the project lifecycle. After gathering requirements, software design is performed in full. Once the design is complete, the software is implemented. The problem with this method is that if a change in requirements occurs, the impact can be devastating [15].

The waterfall process assumes that there will be no changes made after a phase has completed, something that seems quite contrary to empirical and statistical evidence. Change is a natural part of life, not just of software engineering. The attitude toward change that waterfall approaches espouse is that it is expensive, undesirable, and most damningly avoidable. Waterfall methods assert that change can be circumnavigated by spending more time on requirements and design so that changes simply do not occur during subsequent phases. This is preposterous because change will always happen. Waterfall methodologies are also document-centric, generating a lot of documentation that does not directly improve the software product [11].

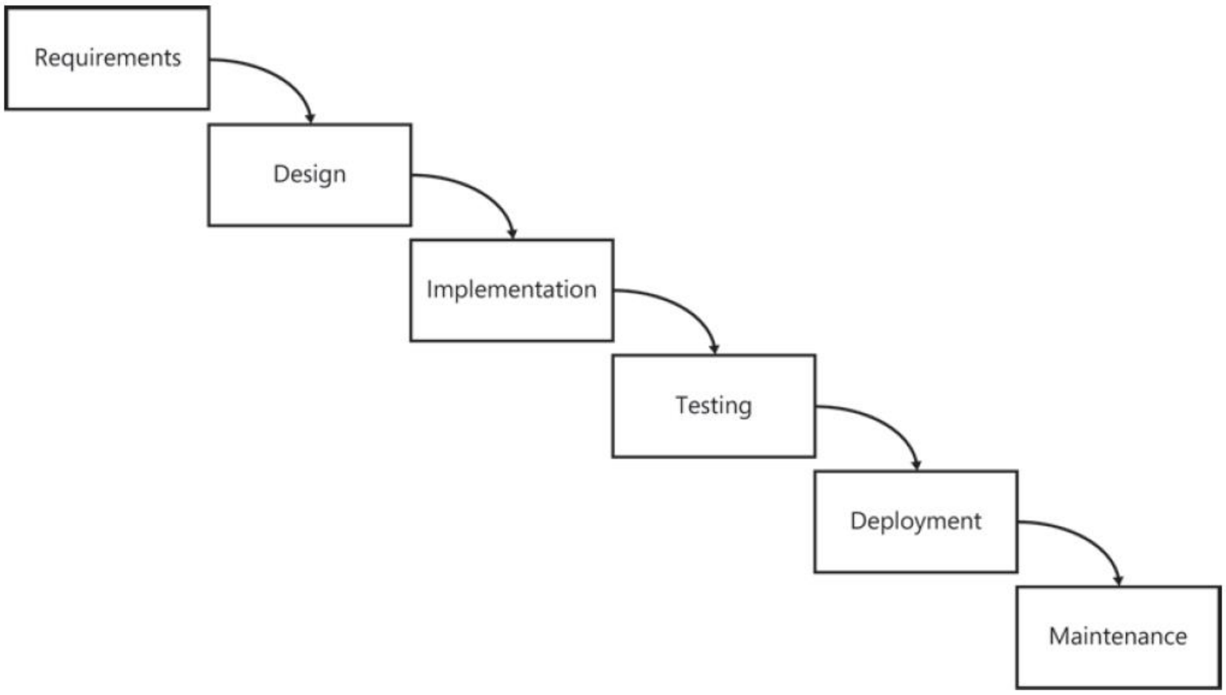


Figure 9: Waterfall development phases (Source: [11])

3.5.2 Rapid Application Development

Rapid Application Development (RAD) is a condensed development process that produces a high-quality system with low investment costs. Scott Stiner, CEO & President of UM Technologies, said: “*This RAD process allows our developers to quickly adjust to shifting requirements in a fast-paced and constantly changing market*.” The ability to quickly adjust is what allows such a low investment cost. The Rapid Application Development method is divided into four phases: Requirements Planning, User Design, Construction, and Cutover. The user design and construction phases are repeated until the user approves that all of the requirements are met.

RAD is most effective for projects with a well-defined business objective and a clearly defined user group, but which are not computationally complex. It is especially useful if the project is of small to medium size and time sensitive. However, it requires a stable team composition with highly skilled developers and users who are deeply knowledgeable about the application area. Deep knowledge is essential when working on a condensed development timeline that requires approval after each construction phase. If you don't have these requirements, RAD may not work well for your organization [17].

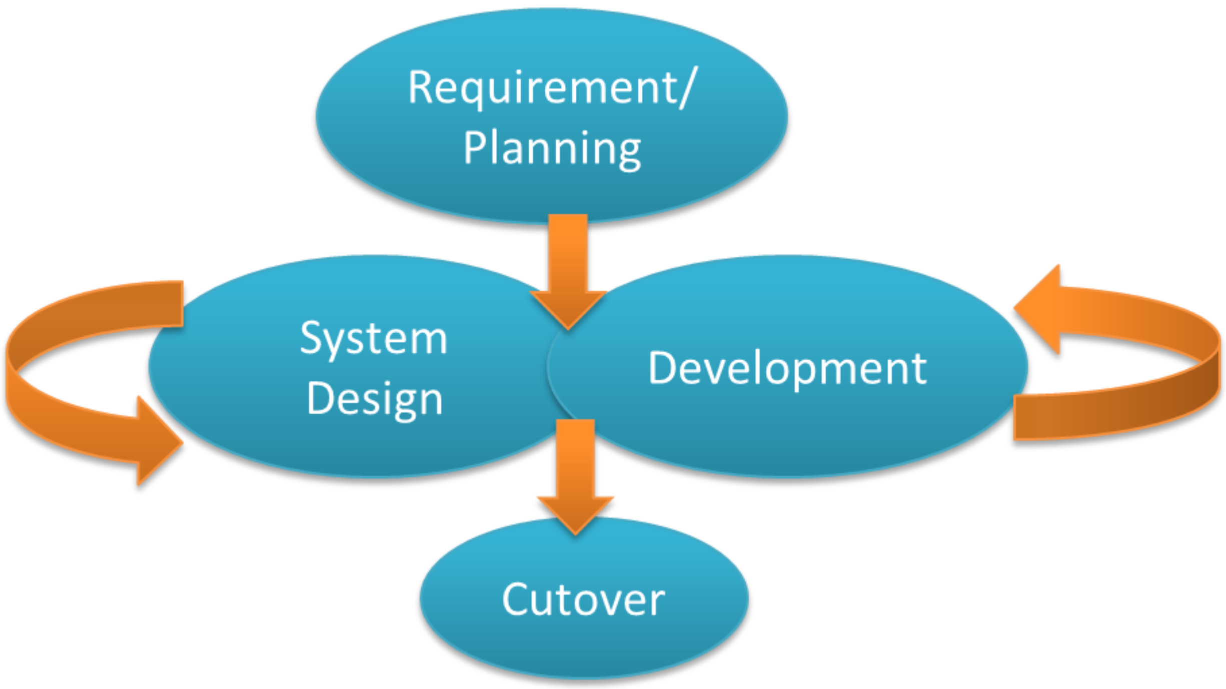


Figure 10: Rapid application development (Source: [17])

3.5.3 Scrum

Scrum is a project management methodology. To be more precise, it is an Agile methodology. Scrum is based on the idea of adding value to a software product in an iterative manner. The overall Scrum process is repeated – iterated – multiple times until the software product is considered complete or the process is otherwise stopped. These iterations are called sprints, and they culminate in software that is potentially releasable. All work is prioritized on the product backlog and, at the start of each sprint, the development team commits to the work that they will complete during the new iteration by placing it on the sprint backlog. The unit of work within Scrum is the story. The product backlog is a prioritized queue of pending stories, and each sprint is defined by the stories that will be developed during an iteration.

Agile is a family of lightweight software development methods that embrace the changing requirements of customers even as the project is in progress. Agile is a reaction to the failings of more rigidly structured practices. The Agile Manifesto exemplifies the contrast. It can be found on the web at www.agilemanifesto.org. Agile methods emphasize using UML as a sketch.

The Agile Manifesto was signed by 17 software developers. The Agile method has grown in influence in the intervening years to the extent that experience in an Agile environment is now a common prerequisite for software development roles. Scrum is one of the most common implementations of an Agile process [11, 16].

Agile methods also emphasize working software as the primary measure of progress. Combined with the preference for face-to-face communication, agile methods produce very little written documentation relative to other methods.

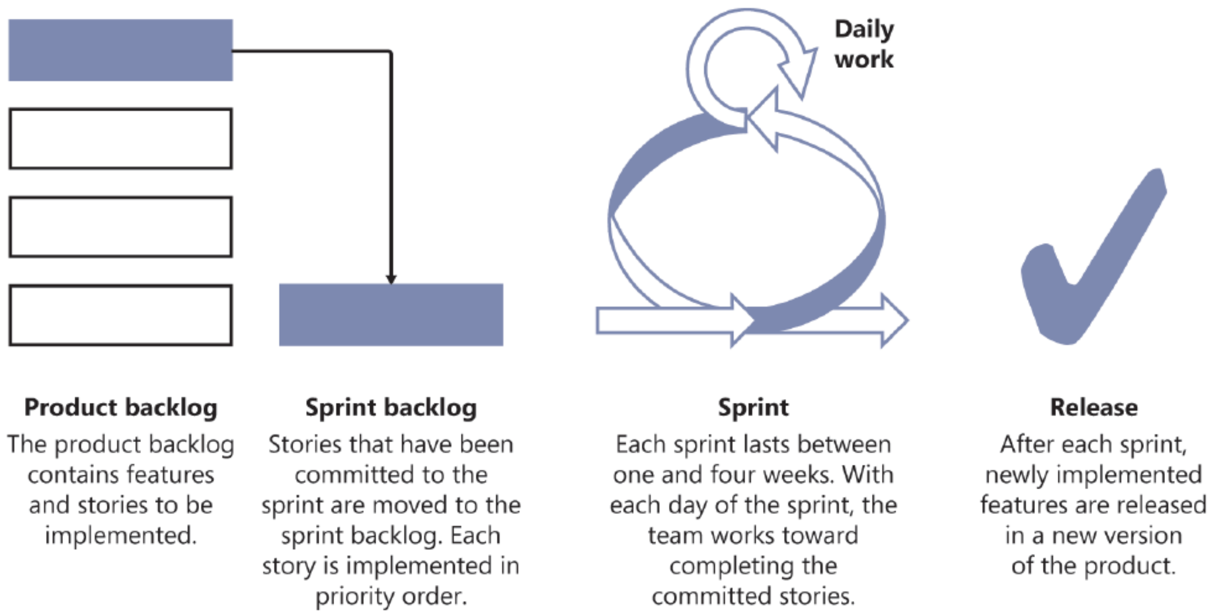

Daily work
Product backlog
The product backlog contains features and stories to be implemented.
Sprint backlog
Stories that have been committed to the sprint are moved to the sprint backlog. Each story is implemented in priority order.
Sprint
Each sprint lasts between one and four weeks. With each day of the sprint, the team works toward completing the committed stories.
Release
After each sprint, newly implemented features are released in a new version of the product.

Figure 11: Agile software development (Source: [11])

3.6 Cloud Computing

Cloud computing means storing and accessing data and programs over the Internet from a remote location or computer instead of our computer's hard drive. This so-called remote location has several properties such as scalability, elasticity etc., which is significantly different from a simple remote machine [7, 9]. Cloud computing is an (IT) platform that enables unlimited access to shared pools of configurable system resources. Cloud computing enables on sharing of resources to provision multiple scales of company infrastructure or services. Third-party cloud service providers enable organizations to focus or concentrate on their core businesses design structure instead of worrying about resources on computer infrastructure and maintenance. Cloud computing gives benefits that companies to avoid or minimize up-front IT infrastructure costs.

Service Models

Infrastructure as a service (IaaS): Provide physical computing infrastructure resources to the clients. For example- virtual machines, servers, storage, load balancer, network and so on.

Platform as a service (PaaS): Provide a service to deploy onto the cloud infrastructure the applications that have been created using programming languages or services. For example- database, web server, development tools, etc.

Software as a service (SaaS): Enable the customer or client to use the company applications running on a cloud infrastructure. The applications are accessible from various client devices such as a web browser. For example- Adventure camping management system, email, mobile app, etc.

Finally, the broad concept of Information Systems and UML discussed in brief. This section is an introductory or first part of the objectives explained above. Information Systems, Object-Oriented development, origin and history of UML, basic and key diagrams that used to design information systems, popular and widely used system development methodologies are the main components explained in this section. This is not enough for the topics listed above, it needs further study and research.

4. Design of Adventure Camping Management System

4.1 Adventure Camping Management System

Adventure camping is a supervised program for children, teenagers, and/or families conducted during vacation periods in most countries. The common type of camping is a woody place with hiking, campfires, and newer types of specialized activities. The primary business of Adventure Camping is selling camping holiday packages to the general public. A typical camping holiday package consists of one or more tents and few essential items for a reserved period of time, maintained by the company in their own camping sites and other privately-owned sites.

Typically, customers make their inquiries through the website on forthcoming holiday packages. Customer goes through the confirmed reservations to check the availability of the needed packages and the intended dates. If the required dates have already been reserved, the customer can go through the other package offers. If the interested package is available, the customer will be asked to log in or to register. Pending reservation will be made after customer provided his/her personal information and number of tents needed.

To confirm the pending reservation, the customer is requested to settle the full payment through online-payment at least seven days prior to the reservation date and must print the site-copy of the payment receipt. Otherwise, the pending reservation will be expired. Once paid, the system marks the corresponding pending reservation as “Paid”.

The Reservations Clerk goes through the available reservations in every morning to extract the paid and the expired reservations. For each paid reservation, he/she sends the corresponding holiday package with customer’s particulars to the list of confirmed reservation page who have a reservation for the same date. Then the expired pending reservations will be discarded. For each confirmed reservation, he/she creates booking-note, which consists of customer details profile, the number of tents needed and other special remarks.

The relevant booking-note is now accessible by the Site Supervisor by login with his/her account at the corresponding camping site to arrange the place with the given customer’s requirements. The customer can obtain above mentioned essential items provided by Adventure Camping by bringing the site-copy of the payment receipt to the Site Supervisor at the time of check-in to the camping site.

Occasionally, the Reservations Clerk receives new site information from the Management. Using such information, she prepares new camping holiday packages and makes them available for reservation. She also uses revisions sent by the Management to update the existing packages. She prepares a summary report of all refunds that is going to be sent to the Management at the end of the day.

Adventure camping host companies need to manage effectively the campground, follow daily weather forecasts, collecting camp fees, checking visitors, and other related tasks. Some adventure camping host companies have multiple campgrounds in different locations around the world. They might have many customers to serve, many employees to hire, and so many management tasks to be carried out. These companies need a real-time data about their customers, the weather forecast, and the resource utilization.

For these and other purposes adventure camping host companies need to implement a software application- in our case a web-based application. In order to implement this application, it needs to pass through in some of the design procedures. UML will be used for a design of a web-based application by covering all the aspects of an information system.

Adventure Camping Management System is a web-based application designed and developed to support the information system of a particular Adventure Camping host company, which has multiple campgrounds in different locations around the world. Adventure Camping Management System has many components and functionality to support the on-going activities of the company. These functionalities are online registration, customer management, billing system, transportation management, emergency medical information, employee management, reporting and analytics, vacancies and other extra custom features. We will see some of the key features in this document.

Online Registration: The Online Registration System allows customers, through Internet, to register, and create an account.

Customer Management: is a module to manage a company's interaction with current and potential customers. It uses data analysis about customers' history with a company to improve business relationships with customers, specifically focusing on customer retention.

Billing System: a module designed to handle time and billing tracking as well as invoicing customers for services and products.

Transportation Management: the management of transportation operations of all types, including tracking and managing every aspect of the vehicle, costing, routing and mapping, communications, traveler and cargo handling.

Weather Forecast Information: is a statement saying what the weather will be like the next day or for the next few days. This module is an integration with another forecasting company module. It uses API integration.

Emergency Medical Information: is a module that provides useful medical information and addition emergency service dedicated to providing out-of-hospital acute medical care.

Employee Management: is an information system that supports the relationship between a company and its employees. Employee management has focused on enabling employees to collaborate on typical managerial tasks with their employers.

Resource Management: is a module for efficient and effective development of an organization's resources when they are needed. Such resources may include inventory, production resources, or IT systems.

Reporting and Analytics: Reporting is the process of organizing data into informational summaries in order to monitor how different areas of a business are performing. Analytics is the process of exploring data and reports in order to extract meaningful insights, which can be used to better understand and improve business performance.

4.2 Overview of the System

The projected system will be available and online at any time, anywhere and display the all the necessary information needed.

4.2.1 Functional Requirements

Functional requirement defines a function of a system or its component and specifies particular results of the system. It also describes the interaction between the system and the users independent of its implementation. The functional requirements of the system are listed as follows:

- Register customer personal information,
- Register booking information,
- Process the payment based on the payment method,
- Display customers, users/employees, and information online,
- Enable administrators to download reports in pdf format
- The system should provide searching functionality.
- The system provides navigation to a different category.
- The system should authenticate the user and assign privileges according to the assigned rights.
- The system should provide the facility to change the password
- The system should display the administration page menu according to the assigned user’s right and so on.

4.2.2 Non-functional Requirement

The following list of non-functional requirements is expected from the system:

- The windows application part
  - should allow login to only authorized users i.e. users that have username and password,
  - should work in a networked environment,
  - should validate data during data entry,

- The web-based part
  - Should respond to requests in a reasonable period of time, i.e. at least before the session expires,
  - Should give only valid result, if no data is found with the specified criteria the system should not have to crash or give invalid response,

4.3 Analysis Model

4.3.1 Data Dictionary

- **Customer**: A person who reserves camping packages.
- **Reservation Clerk**: An employee who performs clerical work related to reservations.
- **Equipment**: Equipment rent along with the camping packages for the use of customers at a particular camping site. E.g Lantern, barbeque grill.
- **Tent**: Tent rent for customers with the camping packages for the use of customers at a particular camping site.
- **Booking Note**: A note created by Reservation Clerk for each confirmed reservation which consists of customer details profile or name in particulars with his phone number and address, number of tents/equipment needed and other special remarks. This is couriered to Site Supervisor.
- **Reservation**: A reservation for a camping holiday package.
- **Pending Reservation**: A temporary reservation for a camping holiday package.
- **Confirmed Reservation**: A confirmed (by payment) reservation for a camping holiday package.
- **Management**: People who manage the company.
- **Site Supervisor**: An employee who manages a camping site and its equipment and tents.
- **Package**: A camping holiday package. Consists of a combination of tents and equipment for a specific camping site for a specified number of days and for specified maximum number of people.

4.3.2 Class Model

Class Diagram

Figure 12 shows the general layout of putting all the elements (objects and classes) and their relationship into a class diagram. Objects and classes, attributes and links, the association between them have identified prior to design a class diagram. Figure 12 shows a class diagram by understanding the project description of adventure camping management system detail requirements explained in section 4.1, We can see here classes of packages, reservation clerk, confirmed reservation, customer, and so on. The relationship between classes and multiplicity also identified in the diagram.

As we can see from the figure 12 that package is a class and has a composition relationship with tent and equipment classes. Packages contain tents and equipment's which belong to one assembly. The parts in our case tents and equipment classes will disappear by disappearing the assembly which is in our case the package.

We can also see the association class which is created in between customer and package classes. The relationship between customer and package classes is to reserve a package for the holiday season. Customer reserve packages and packages are reserved by a customer. The reservation has its own class because Reservation Clerk mange all confirmed reservations and pending reservations. Confirmed reservations and pending reservations are part of the assembly class which is reservation class.

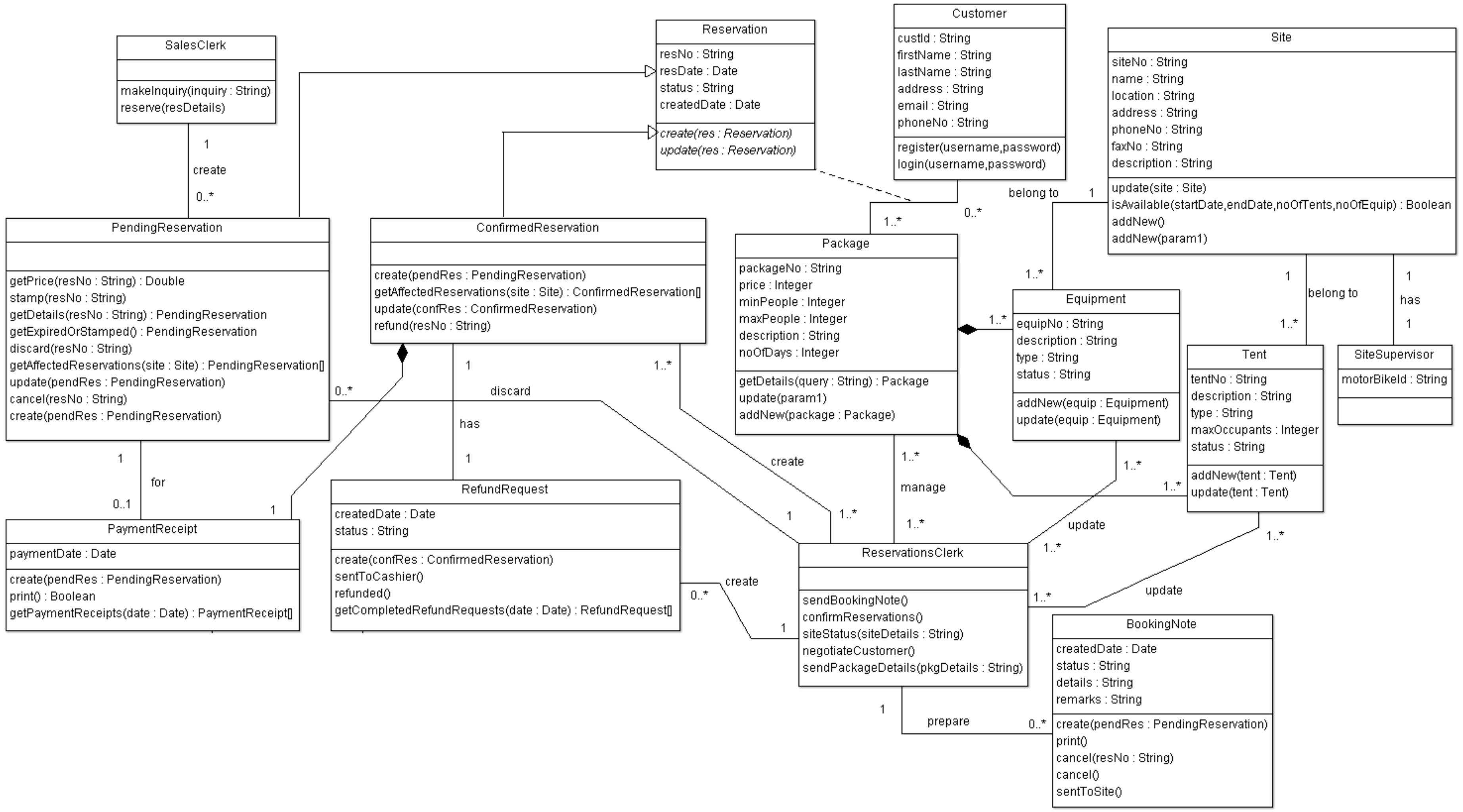


Figure 12: Class diagram of reservation system (Source: [author])

Relationship of the staff members of the company is shown below.

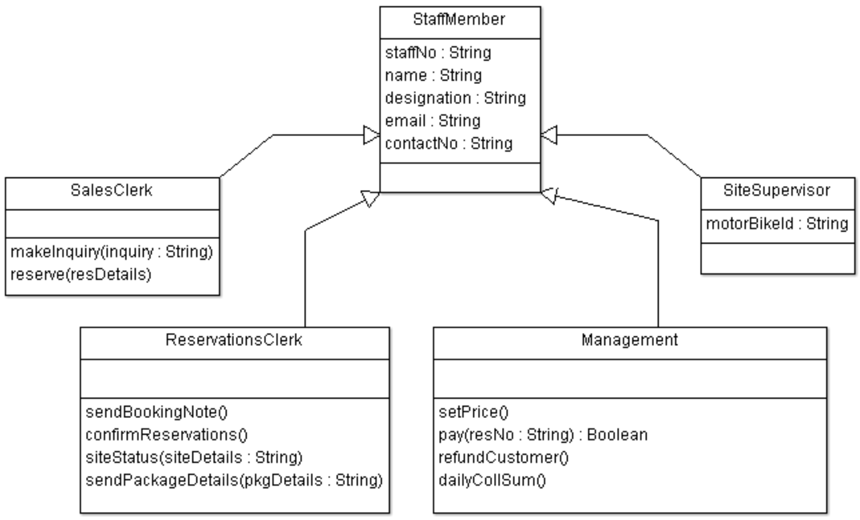


Figure 13: Relationship of staff members (Source: [author])

4.3.3 Interaction Model

Use Case Diagram

Customers can register to the management system portal and make an inquiry. They can check out and pay using a credit card or PayPal. Site Supervisor gets a login credential from the admin or management team in order to check the confirmed reservations and to prepare the site to the customer. Reservation clerk will check all the paid reservations and provide any updated information to the site supervisor. The management will check summary reports and administration. The general structure of system functionality can be shown in figure 12.

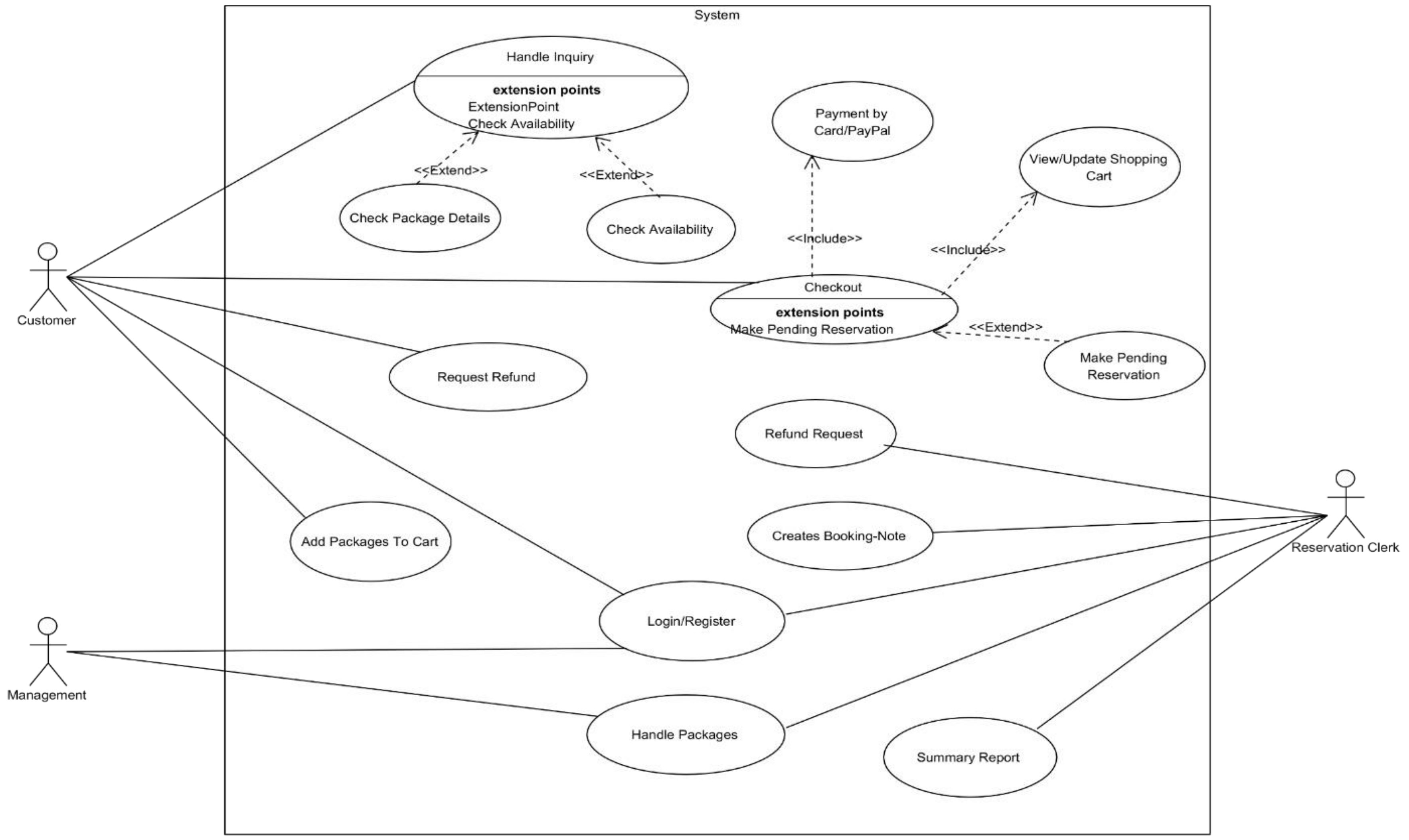


Figure 14: Use Case diagram of reservation system(Source: [author])

Sequence Diagrams

A sequence diagram linked with use case objects. It shows the interaction between participating objects in a given use case. Sequence diagrams are always presented in the form graphical design but are supported by a scenario which is a textual description of the model. We will look the following selected sequence diagrams as follow.

Sequence Diagram of Handle Inquiry

Figure 15 shows a sequence diagram for Handle Inquiry use case.

Scenario: Customer makes an inquiry.

Customer open the website to make an inquiry. The system checks the availability of the required package and quantity. The customer decides to depend on the result. Customer will be ready to make a pending reservation or to check out by the end of this process.

Sequence Diagram of Create Booking-Note

Figure 16 shows a sequence diagram for Create Booking-Note use case.

Scenario: Reservation clerk confirm reservations.

Reservation Clerk login to the management system with her credential. The Reservations Clerk goes through the available reservations in every morning to extract the paid and the expired reservations. For each paid reservation, he/she sends the corresponding holiday package with customer’s particulars to the list of confirmed reservation page who have a reservation for the same date. Then the expired pending reservations will be discarded. For each confirmed reservation, she creates booking-note, which consists of customer details profile, the number of tents needed and other special remarks.

Sequence Diagram of Handle Inquiry

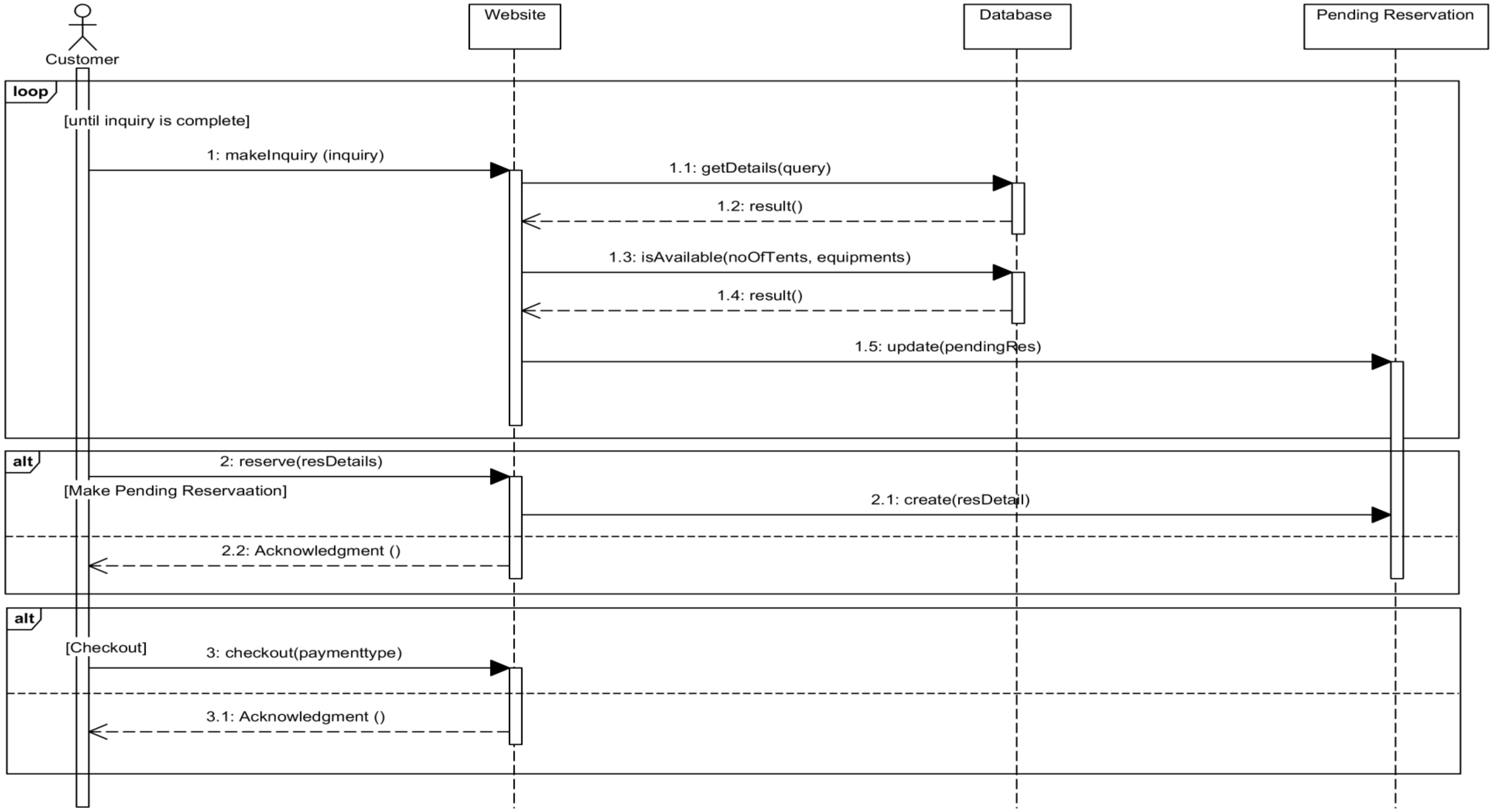


Figure 15: Sequence diagram of handle inquiry (Source: [author])

Sequence Diagram of Create Booking-Note

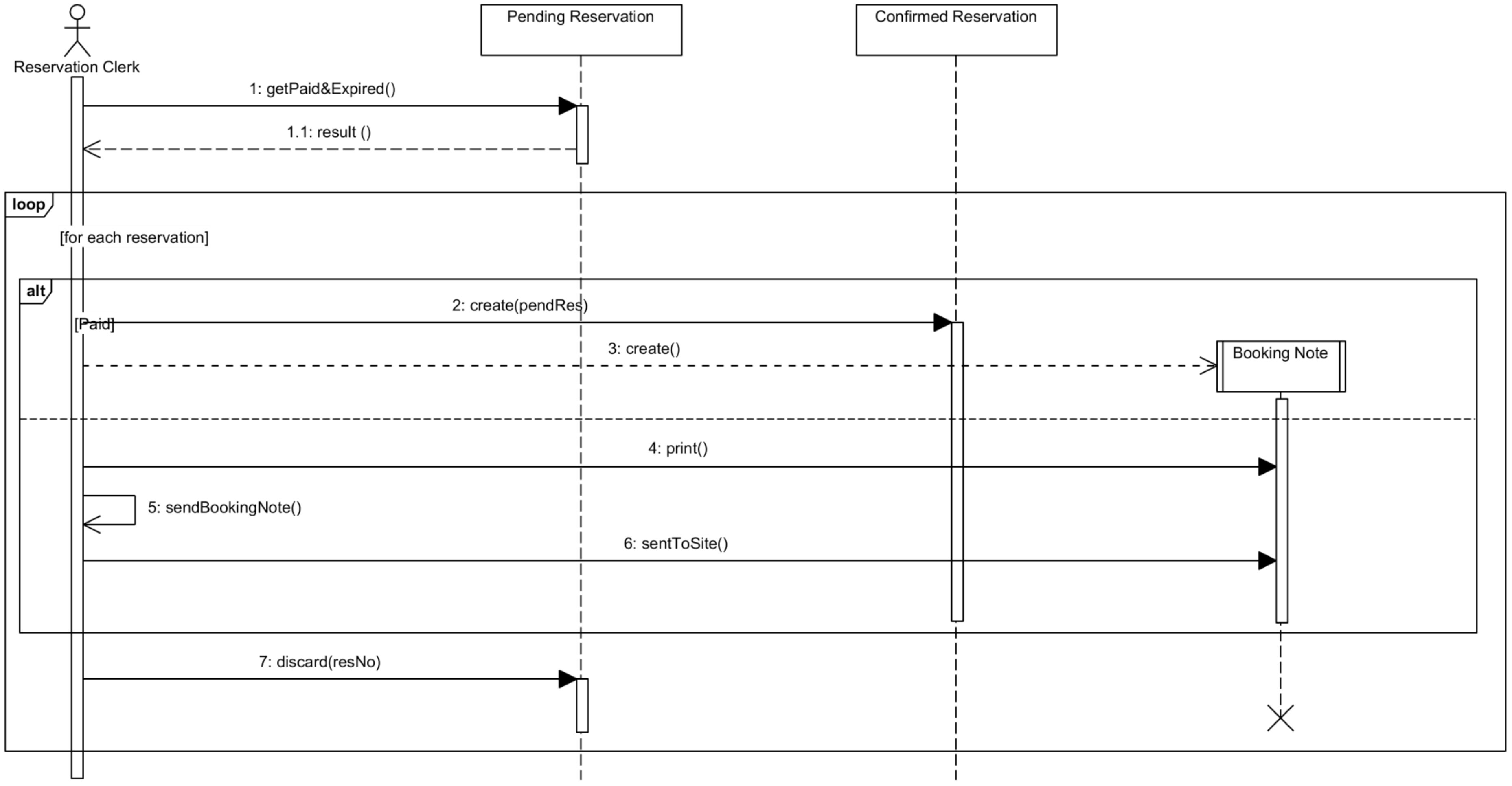


Figure 16: Sequence diagram of creating Booking-Note (Source: [author])

Sequence Diagram of Handle Packages

Figure 17 shows a sequence diagram for Handle Packages use case.

Scenario: Update the camping site information in the system.

The Reservations Clerk receives new site information from the Management. Using such information, she prepares new camping holiday packages and makes them available for reservation. She also uses revisions sent by the Management to update the existing packages.

We can see from the figure 17 that there are two alternative options that need to be carried out individually. If the information from the Management is to create new packages, the Reservation Clerk will fill all the relevant information including the packages, tents, and equipment on the website. The other option is to update the information on the existing packages. The Reservation clerk modifies the quantity, type, and size of the available packages, tents or equipment.

Sequence Diagram of Summary Report

Figure 18 shows a sequence diagram for Summary Report use case.

Scenario: Reservation Clerk sends a daily summary report to management.

Reservation Clerk prepares a summary report of all payment records and refunds that is going to be sent to the Management at the end of the day. This helps the management to improve their services and their system. A new summary report created every day and sent to Management. Management gets a printed version and emailed report from Reservation Clerk.

Sequence Diagram of Handle Packages

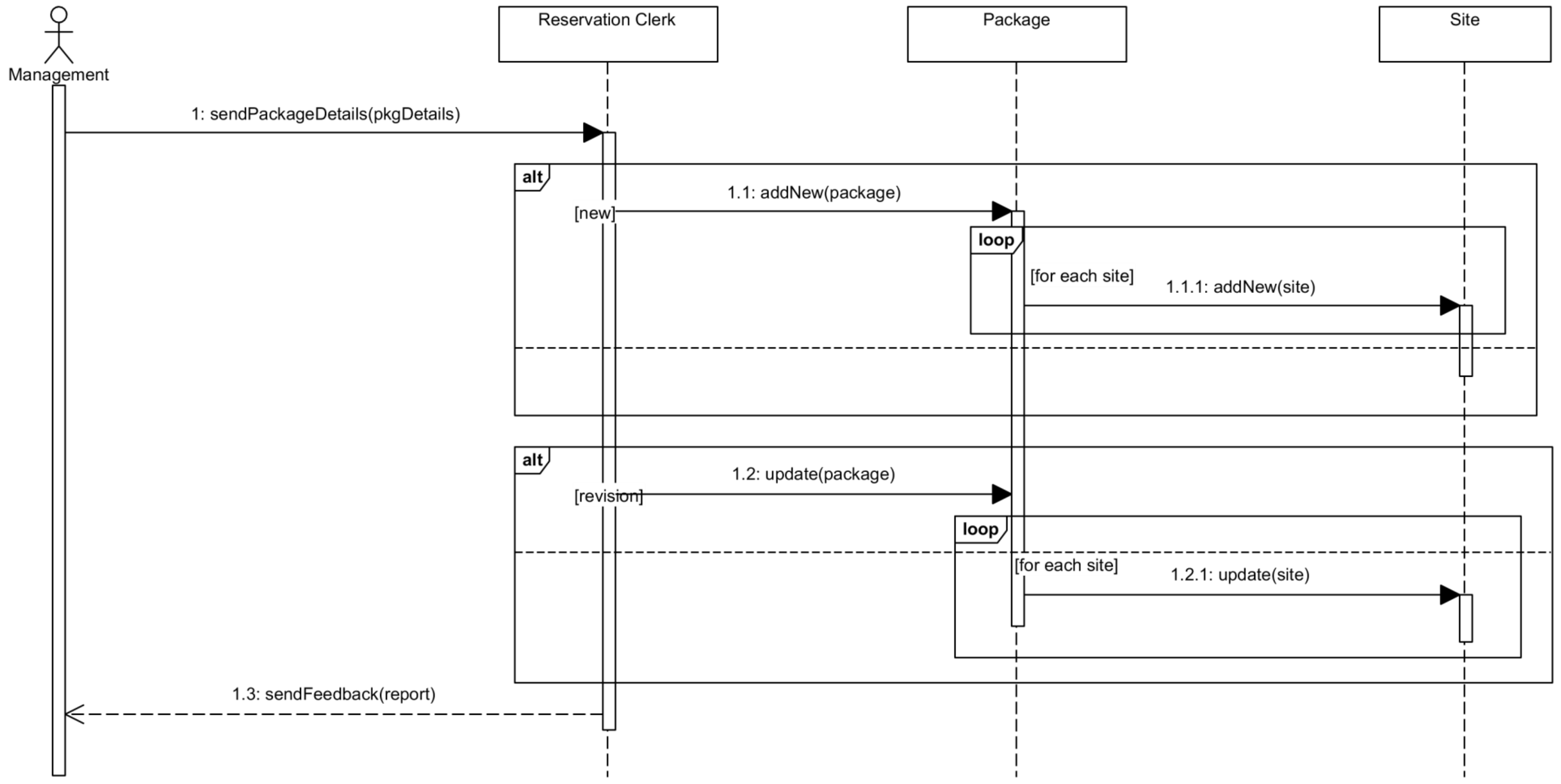


Figure 17: Sequence diagram of handle packages (Source: [author])

Sequence Diagram of Summary Report

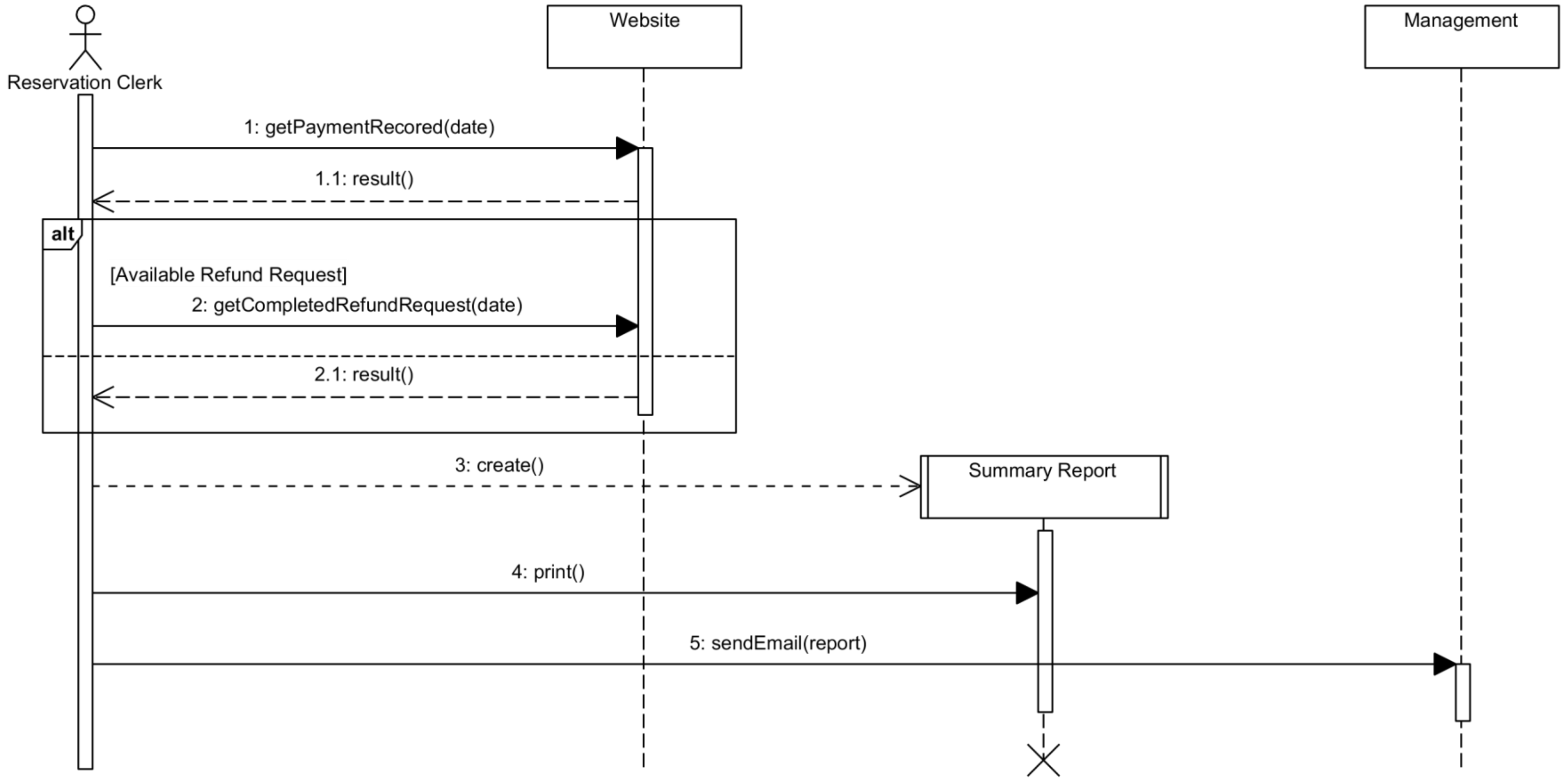


Figure 18: Sequence diagram of summary report (Source: [author])

Activity Diagrams

Activity Diagram of Handle Inquiry

Figure 19 shows an activity diagram for handle inquiry. The customer can decide for making pending or temporary reservations for the period of seven days. The payment should be completed within these days. A customer can check out also directly to reserve the packages for the given days.

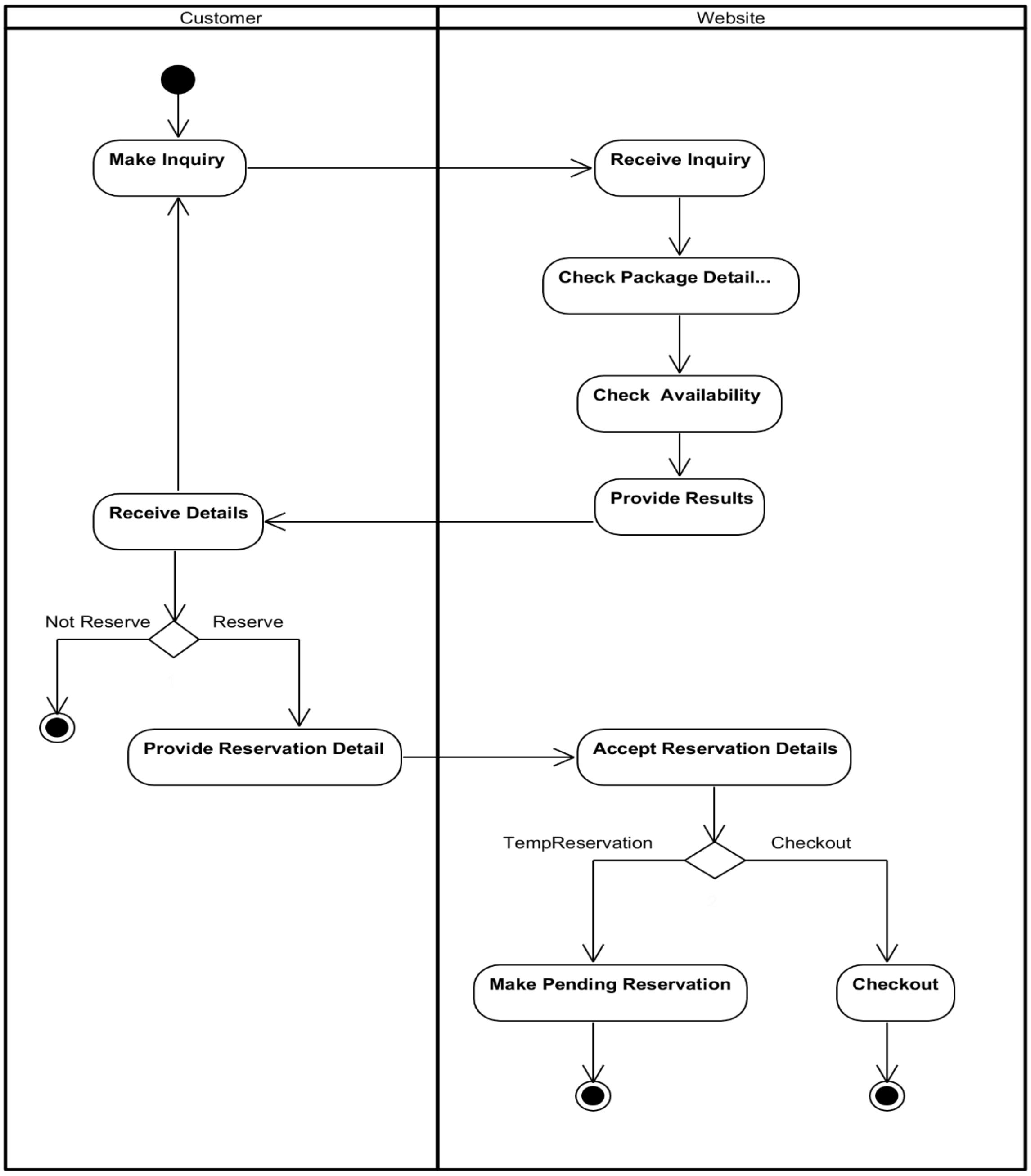


Figure 19: Activity diagram of handle inquiry (Source: [author])

Activity Diagram of Create Booking-Note

Figure 20 shows an activity diagram for creating a booking-note. This shows when to take alternatives and decisions. Reservation Clerk checks all the paid and expired reservations. The expired reservation will be discarded accordingly. She will send the prepared booking-note with customer details to Site Supervisor.

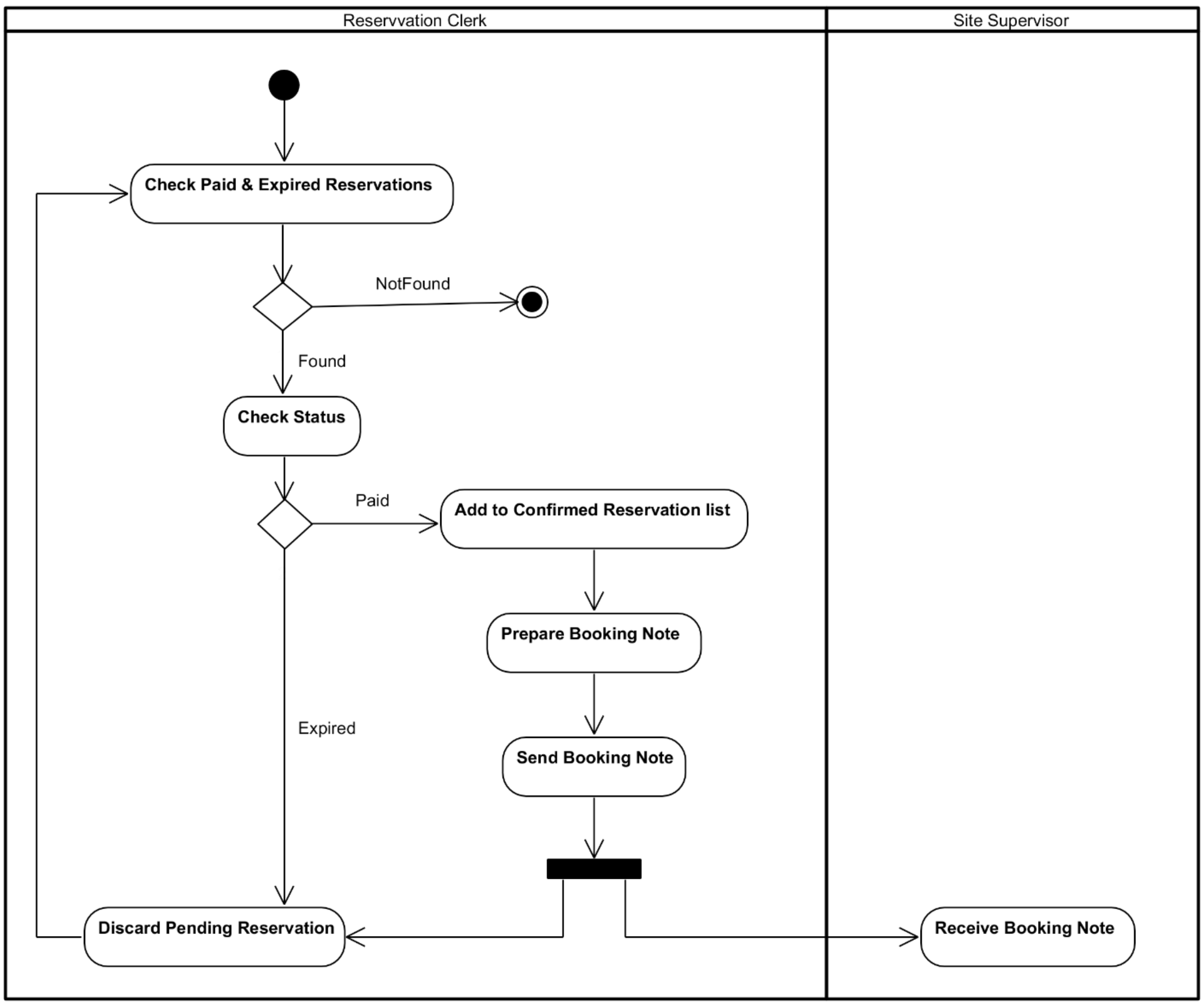


Figure 20: Activity diagram of creating Booking-Note (Source: [author])

Activity Diagram of Handle Packages

Figure 21 shows an activity diagram for handle packages. This also shows when to take alternatives and decisions. Management sends new or updates campground or site information to reservation clerk. The Reservation Clerk updates the website information accordingly and sends a confirmation to Management.

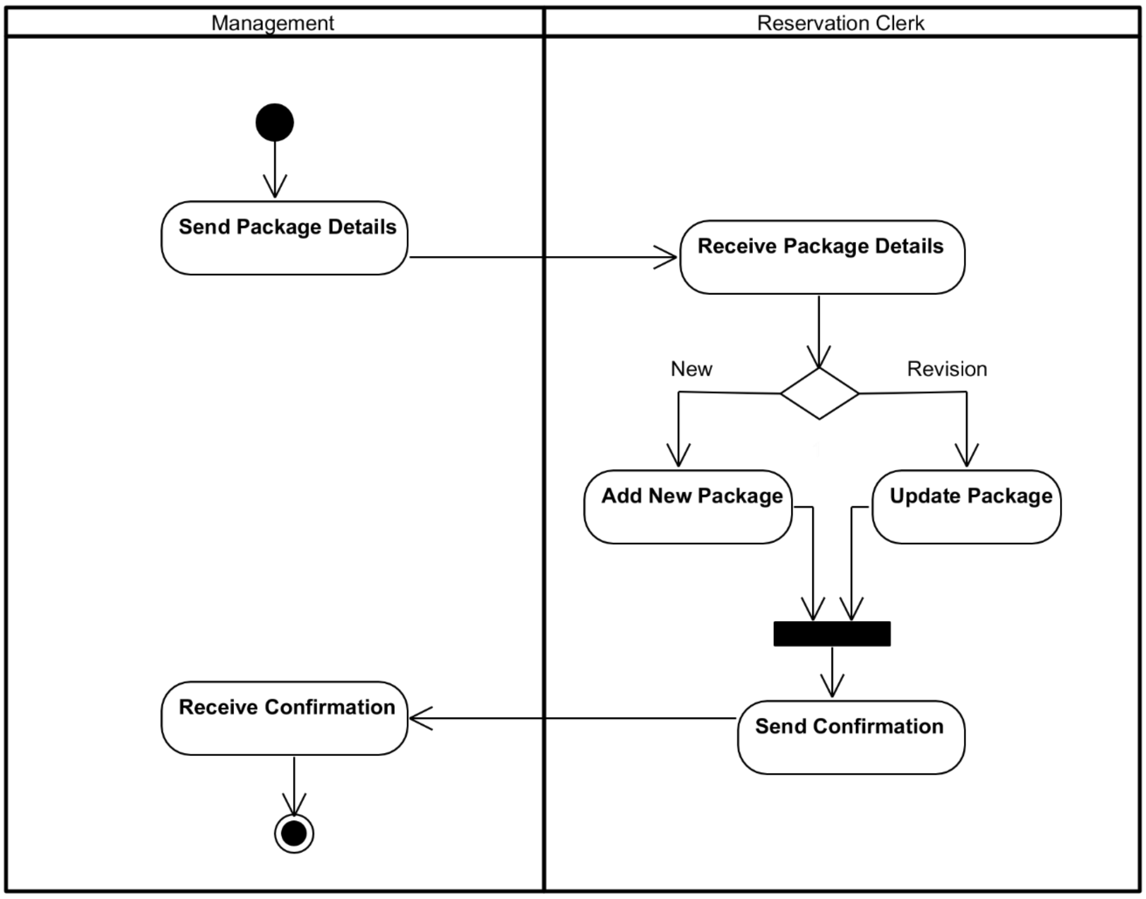


Figure 21: Activity diagram of handle packages (Source: [author])

Activity Diagram of Summary Report

Figure 21 shows an activity diagram for the summary report. Reservation Clerk collects all payment information and refund requests available. At the end of the day she will send it to Management. As we can see from the figure that both collection summery merged together to create a single report.

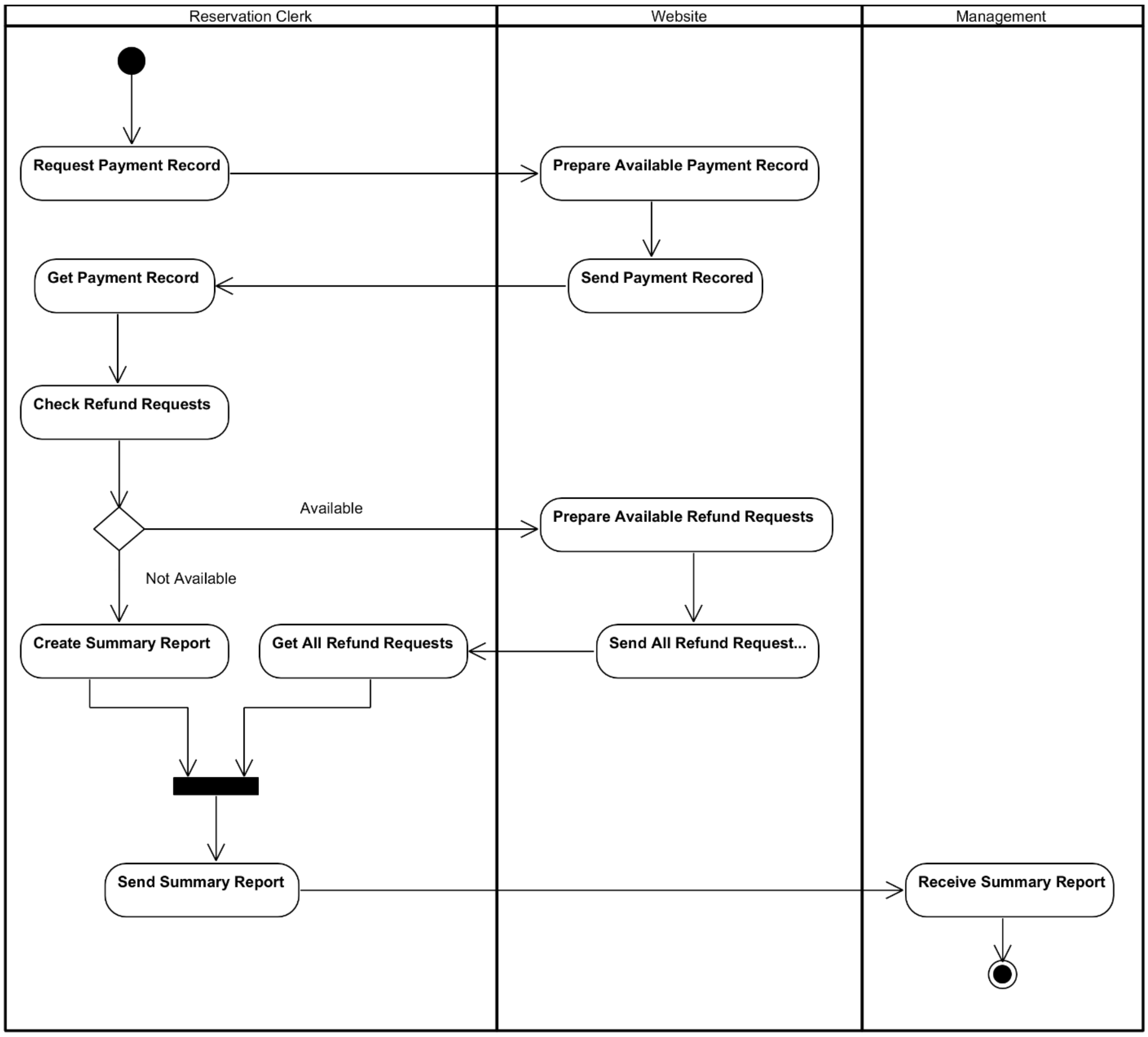


Figure 22: Activity diagram of summary report (Source: [author])

4.3.4 State Model

State Diagrams

State machine diagram is a graphical representation that describes the behavior of systems which is based on the concept of state diagrams. State diagrams depict the permitted states and transitions as well as the events that affect these transitions.

Handle Inquiry State Diagram

The following figure shows the state diagram of handle inquiry. The state starts with the main menu which uses a do activity. The customer makes inquiry and then add packages to the cart. The packages added to the cart is subjected to make pending reservation, checkout or cancel the reservation process. As we can see from the figure there are few conditions are associated with the transition.

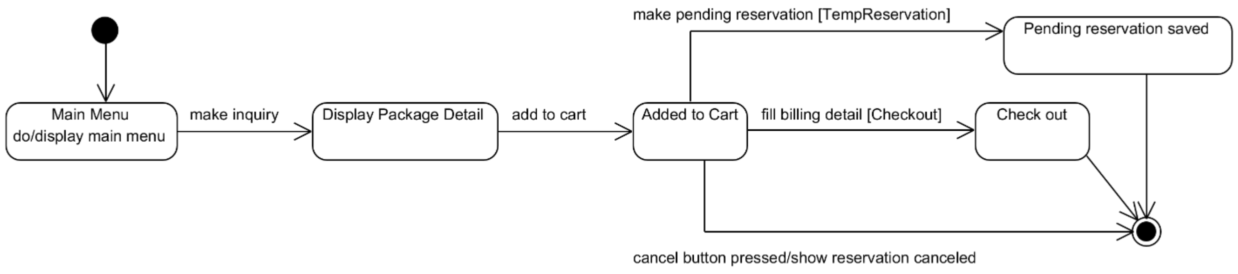
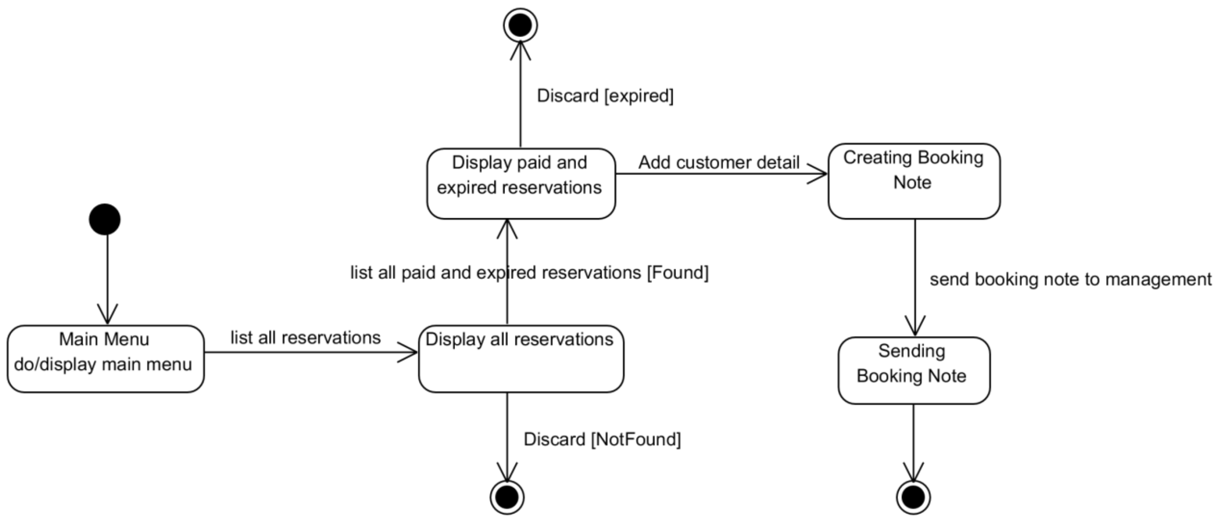

make pending reservation [TempReservation]
Pending reservation saved
Main Menu
do/display main menu
make inquiry
Display Package Detail
add to cart
Added to Cart
fill billing detail [Checkout]
Check out
cancel button pressed/show reservation canceled

Figure 23: Handle Inquiry state diagram (Source: [author])

Create Booking-Note State Diagram

Figure 24 shows the state diagram of creating a booking note. The state starts with the main menu as mentioned previously, which uses a do activity. The Reservation Clerk goes through a list of reservations available. If there are no reservations available the state will jump to the terminal state. States fired only in a case when the condition is satisfied at the moment when the event occurred [38].

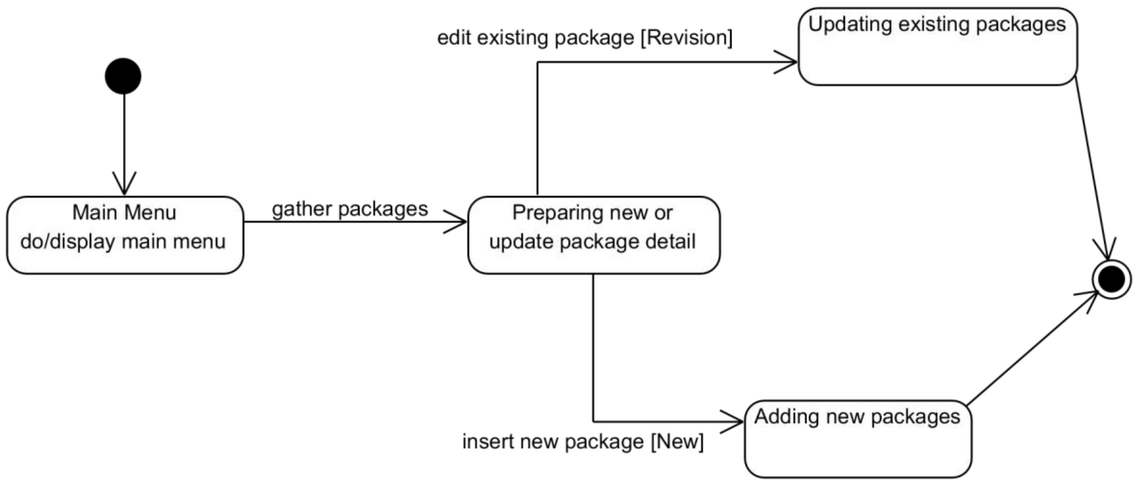


Discard [expired]
Display paid and expired reservations
Add customer detail
Creating Booking Note
list all paid and expired reservations [Found]
send booking note to management
Main Menu do/display main menu
list all reservations
Display all reservations
Sending Booking Note
Discard [NotFound]

Figure 24: Create Booking-Note state diagram (Source: [author])

Handle Packages State Diagram

Figure 25 shows the state diagram of handle packages. The Management collects all new information and send it to Reservation Clerk. Reservation Clerk check adds the relevant information to the system by adding new packages or by updating the existing packages.



edit existing package [Revision]
Updating existing packages
Main Menu do/display main menu
gather packages
Preparing new or update package detail
insert new package [New]
Adding new packages

Figure 25: Handle packages state diagram (Source: [author])

A case study has been selected and discussed with the support of various UML diagrams. This section primary discussed and explained to design a particular information system using UML. Visual Paradigm 15.0 used to design all UML diagrams in this document.

4.4 Architecture of the System

Design of an information system must follow standard design procedures. These standard procedures or design criteria are widely accepted and used in many software development applications. These design criteria are explained as follow:

Performance: The system should respond fast having high throughput, i.e. it should perform request processing in a time less than 1 minutes and when customers retrieve order information it should respond quickly.

Reliability: The system must have a backup plan to accommodate any service failure with minimum downtime. The backup strategy will also help to restore at any point in time to recover from any data loss.

Availability: The system should be available 27/7 for the whole year. The system should not have any downtime at any time. For this purpose, SLA must be put in place to recover quickly with minimum downtime if something worse happens.

Maintenance: The system should be easily maintainable periodically. This will boost the performance and the functionality of the system. Maintenance can be performed and tested by system developers.

Security: The security part is only for the authorized administration staffs as per their assigned rights. Some access limit can be granted to authorized users. The system also enabled to put all IP addresses into blacklist whenever there is a failed login attempt multiple times.

End User: The system should have a responsive graphical user interface such as forms and buttons which have descriptive names. It should be flexible to fit for mobile and desktop versions. The system also must be fast when processing the data and the requests.

4.4.1 Proposed System Architecture

The proposed subsystem will be implemented in high availability client-server structure. The database servers will be up all the time and are clustered for high availability features. This will give additional capability and performance when there is an outage or critical problems in the system. It is a preferred way to perform regular maintenance operations in the infrastructure. This functionality of the system will be transparent for the end users or customers. There will not be any downtime and unavailability of the services and application.

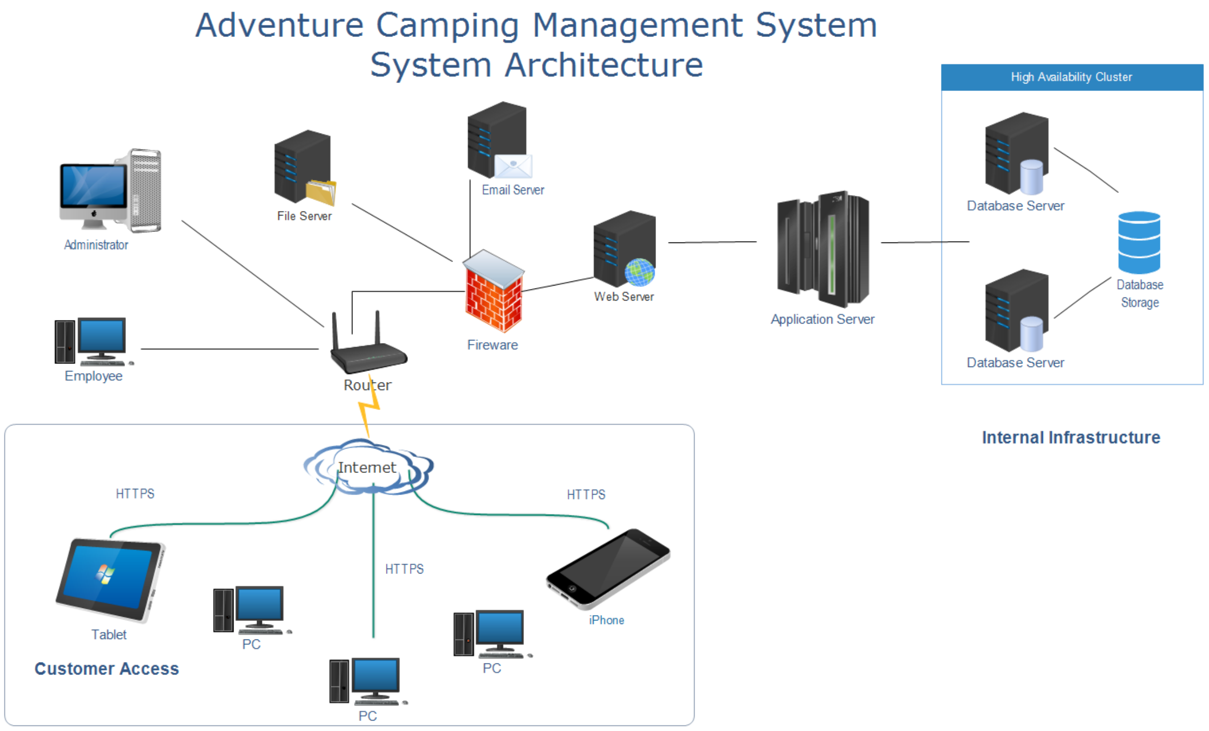


Figure 26: ACMS system architecture (Source: [author])

4.4.2 Subsystem Decomposition

The proposed system is constructed from seven main subsystems. The system is categorized according to the specialized task it performs to ensure high coherence. The level of high interactions between subsystems is an only exchange of data the following four subsystems will handle the major task.

1. Search Subsystem

It deals with the electronic and online global search functionality of the system in addition to dealing with new information captured from the users to meet the interactivity of the system. It provides search for different packages and services provided by the adventure camping management system.

2. Resources Subsystem

This group is a collection of different types of packages that will be available for any type of camping activities in the adventure camping. Customers will decide by selecting different types of items and their quantity. If the customers couldn't find the required packages on the watch list, they will contact adventure camping support team through a contact form available on the website.

3. Customers Subsystem

This subsystem includes the collection of all customer activities and interactions with the system and with adventure camping. This will help to identify the high potential customers and the services that will be delivered. The user accounts and the access level will be set accordingly.

4. Employee Subsystem

Employee subsystem includes the collection of all employee activities and interactions with the system and with adventure camping. This is designed for monitoring the activities of employees, determining active and passive employees, identifying a vacant space in the major activities by adventure camping and so on.

5. Security Subsystem

The security subsystem needs to have a high level of design and structure to handle any type of security issue. Security is a major component of many business architectures and needs to be considered seriously. A security breach will cause a lot of problems including losing the company. Therefore, each access level will be identified and set of permissions to grant and deny will be in place accordingly.

6. Weather Forecast Subsystem

This subsystem will collect all the necessary data that will be used for forecasting purpose. These data can be collected from a various organization who built their entire infrastructure on weather forecasting systems. The data is collected through API integration to adventure camping systems.

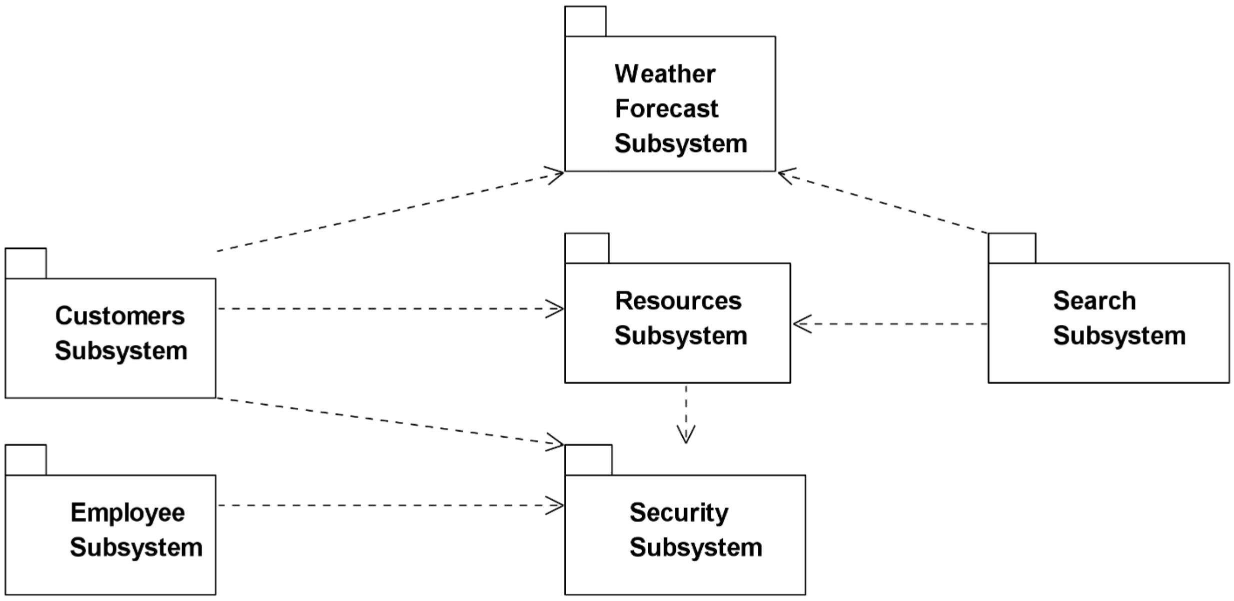


Figure 27: Subsystem decomposition (Source: [author])

This section discussed the design of the system, includes the architecture of the system and subsystem decomposition. This helps to construct a clear implementation process. Most of the implementation process takes too long because they don’t have a clear structure of the system and how modules organized and integrated. This section is a baseline for implementation which will be discussed in the next section.

5. Implementation of Adventure Camping Management System

5.1 Overview

Implementation is a process of making sure that the information system is working as designed. It involves creating a new system from scratch and/or creating a new system from the existing one. The implementation allows the users to work on the system and to evaluate. It involves training the users to make them working on the system and plan for a smooth conversion. Systems operators must get appropriate training such that they can handle all possible operations, both routine and extraordinary. The operators should be trained in what common malfunctions may occur, how to recognize them, and what steps to take when they come.

A prototype web-based application has been implemented for Adventure Camping Management System using ASP.NET [C#]. UML is a language to convey system ideas and concepts to other designers, stakeholders, coders, testers etc. UML can be used in many places in system development and can be used in any development methodology, structured, rational, or agile.

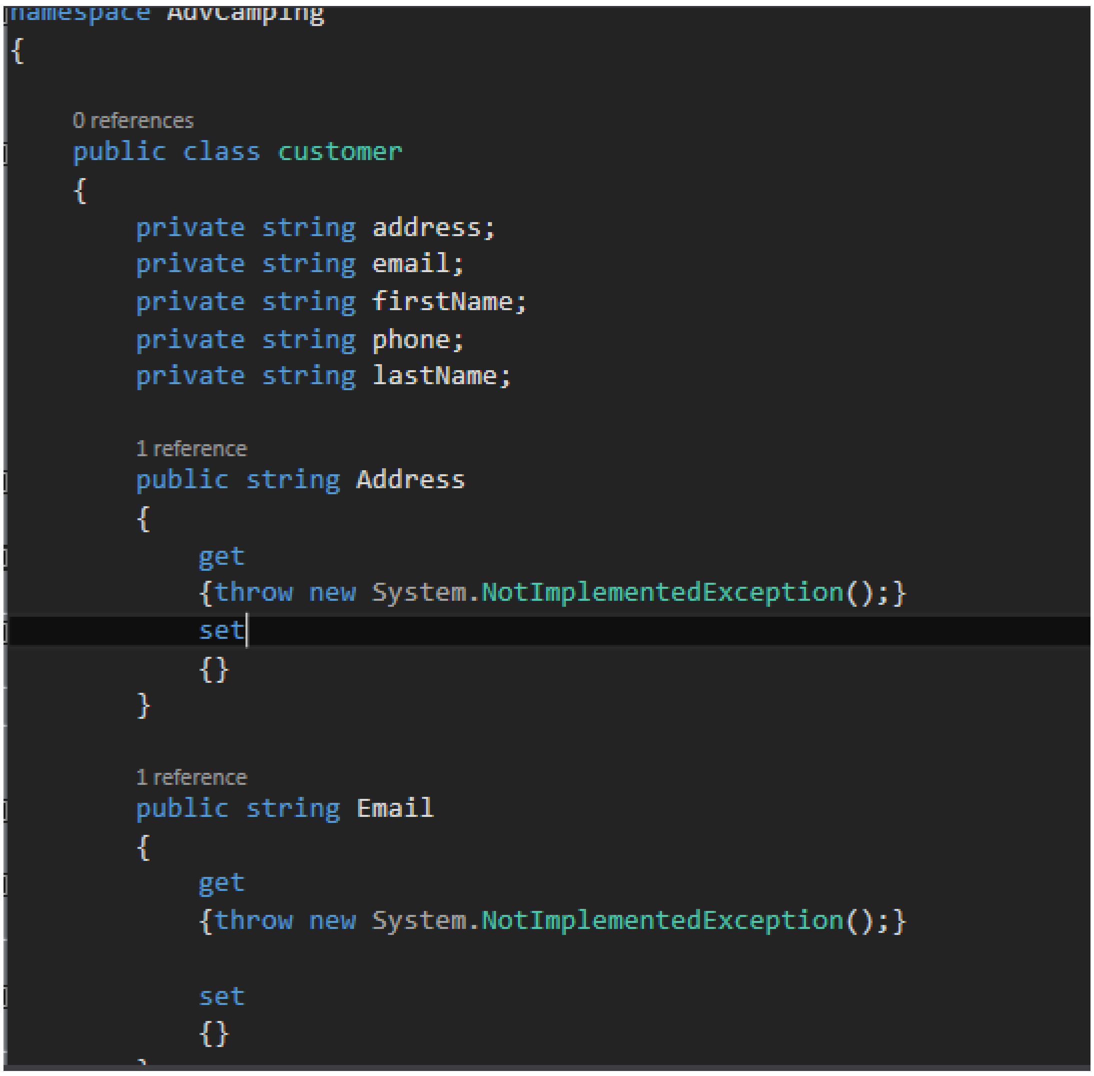
UML enables a precise, unambiguous and complete specification of the proposed system. Possible to generate some of the code in a specified computer language directly from the UML model. Therefore, UML used to generate a code for adventure camping management systems, which is a web-based application. This document discusses only the process of how a customer makes inquiries for the forthcoming holiday packages. The process has been explained well in section 3 and 4.

The development of adventure camping management system is based programming language code and concrete aspects of user interface (UI), such as fonts, and colors. Adventure camping management system have different modules but, in this document, the process of making a reservation has been designed and implemented to explain the use of UML. Each process and user interactions are explained well in this section.

Here we can see some code snippets from the implementation part. The implementation is based on ASP.NET [C#]. As ASP.NET is a server-side web application, it is composed of HTML tags, CSS and javascript. C# acts as a backend integrated with .aspx pages. Additional classes can be added and relationships can be performed as required.

ACMS C# Code for Customer Class

Figure 28 shows customer class definition with some attributes or fields. Customer provide these fields when they registered ant the registration page. These fields are very important to identify potential customers.

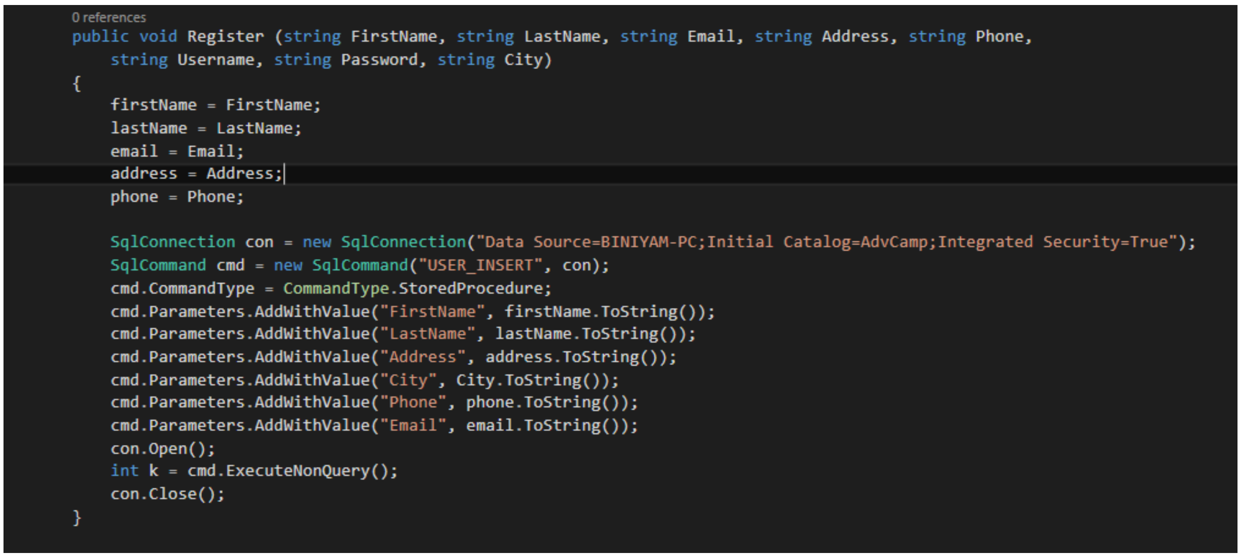


```
namespace AdvCamping
{

    0 references
    public class customer
    {
        private string address;
        private string email;
        private string firstName;
        private string phone;
        private string lastName;

        1 reference
        public string Address
        {
            get
            {throw new System.NotImplementedException();}
            set
            {}
        }

        1 reference
        public string Email
        {
            get
            {throw new System.NotImplementedException();}

            set
            {}
```

Figure 28: ACMS C# code for customer class (Source: [author])

ACMS C# Code for Registration

Customer class has two method the first is register and the second is login. Figure 29 shows the method and the code for registration. The code gets values from parameters. Values passed to parameters from the website textbox fields. Database connection created to insert values into the table. Business logic is created to add a tier between the application and the database. Which means a developer should use a stored procedure or views to communicate with a table. This will limit access to the tables directly from the application, security issue. Then data is inserted accordingly to the matching fields.

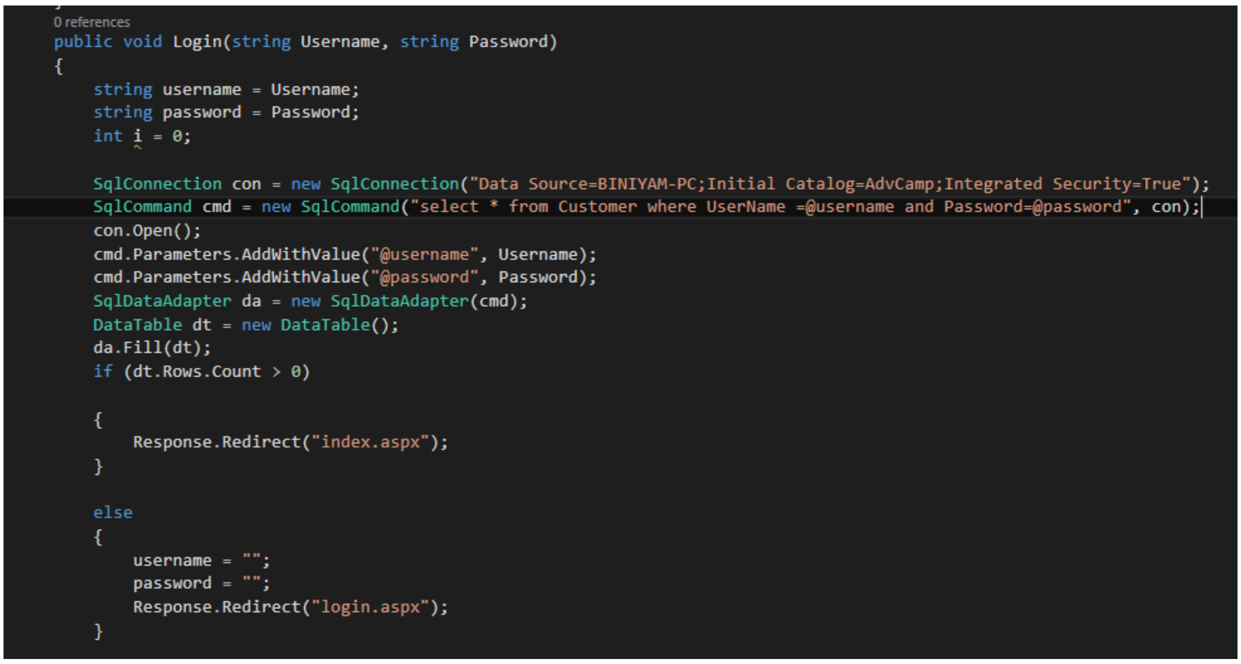


```
0 references
public void Register (string FirstName, string LastName, string Email, string Address, string Phone,
    string Username, string Password, string City)
{
    firstName = FirstName;
    lastName = LastName;
    email = Email;
    address = Address;
    phone = Phone;

    SqlConnection con = new SqlConnection("Data Source=BINIYAM-PC;Initial Catalog=AdvCamp;Integrated Security=True");
    SqlCommand cmd = new SqlCommand("USER_INSERT", con);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("FirstName", firstName.ToString());
    cmd.Parameters.AddWithValue("LastName", lastName.ToString());
    cmd.Parameters.AddWithValue("Address", address.ToString());
    cmd.Parameters.AddWithValue("City", City.ToString());
    cmd.Parameters.AddWithValue("Phone", phone.ToString());
    cmd.Parameters.AddWithValue("Email", email.ToString());
    con.Open();
    int k = cmd.ExecuteNonQuery();
    con.Close();
}
```

Figure 29: ACMS C# code for registration (Source: [author])

ACMS C# Code for Login

Figure 30 shows the method and the code for login. The code gets username and password from parameters. Values passed to parameters are from the website textbox fields. Database connection created to check the matching value for login credentials. If the matching value found the user is directed to the home page, if not the inserted fields will be cleared and requested to login again.

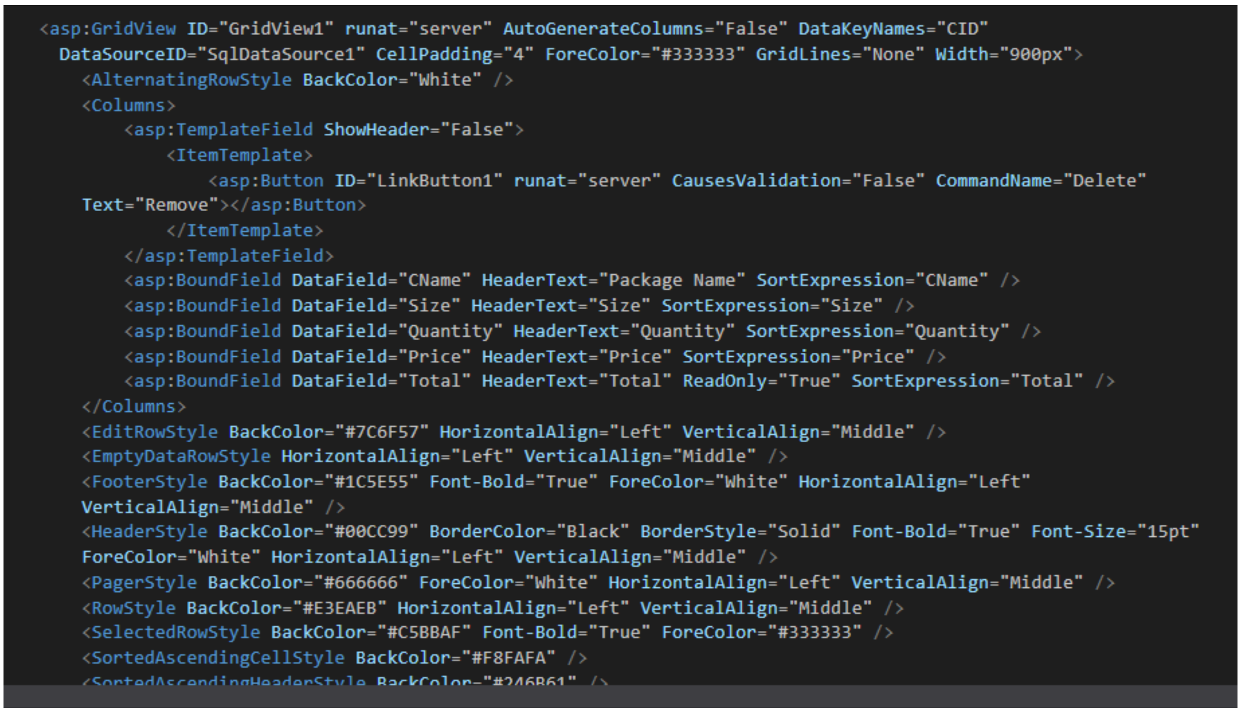


```
0 references
public void Login(string Username, string Password)
{
    string username = Username;
    string password = Password;
    int i = 0;

    SqlConnection con = new SqlConnection("Data Source=BINIYAM-PC;Initial Catalog=AdvCamp;Integrated Security=True");
    SqlCommand cmd = new SqlCommand("select * from Customer where UserName =@username and Password=@password", con);
    con.Open();
    cmd.Parameters.AddWithValue("@username", Username);
    cmd.Parameters.AddWithValue("@password", Password);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if (dt.Rows.Count > 0)

    {
        Response.Redirect("index.aspx");
    }

    else
    {
        username = "";
        password = "";
        Response.Redirect("login.aspx");
    }
```

Figure 30 ACMS C# code for login (Source: [author])

ACMS HTML for Creating a GridView

This is the snippets from cart.aspx that shows the list of items in a GridView. The GridView gives a dynamic feature that a customer can remove the selected packages. It is very elegant style, easy to use and understand.



```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="CID"
  DataSourceID="SqlDataSource1" CellPadding="4" ForeColor="#333333" GridLines="None" Width="900px">
    <AlternatingRowStyle BackColor="White" />
    <Columns>
        <asp:TemplateField ShowHeader="False">
            <ItemTemplate>
                <asp:Button ID="LinkButton1" runat="server" CausesValidation="False" CommandName="Delete"
    Text="Remove"></asp:Button>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="CName" HeaderText="Package Name" SortExpression="CName" />
        <asp:BoundField DataField="Size" HeaderText="Size" SortExpression="Size" />
        <asp:BoundField DataField="Quantity" HeaderText="Quantity" SortExpression="Quantity" />
        <asp:BoundField DataField="Price" HeaderText="Price" SortExpression="Price" />
        <asp:BoundField DataField="Total" HeaderText="Total" ReadOnly="True" SortExpression="Total" />
    </Columns>
    <EditRowStyle BackColor="#7C6F57" HorizontalAlign="Left" VerticalAlign="Middle" />
    <EmptyDataRowStyle HorizontalAlign="Left" VerticalAlign="Middle" />
    <FooterStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" HorizontalAlign="Left"
    VerticalAlign="Middle" />
    <HeaderStyle BackColor="#00CC99" BorderColor="Black" BorderStyle="Solid" Font-Bold="True" Font-Size="15pt"
    ForeColor="White" HorizontalAlign="Left" VerticalAlign="Middle" />
    <PagerStyle BackColor="#666666" ForeColor="White" HorizontalAlign="Left" VerticalAlign="Middle" />
    <RowStyle BackColor="#E3EAEB" HorizontalAlign="Left" VerticalAlign="Middle" />
    <SelectedRowStyle BackColor="#C5BBAF" Font-Bold="True" ForeColor="#333333" />
    <SortedAscendingCellStyle BackColor="#F8FAFA" />
    <SortedAscendingHeaderStyle BackColor="#246B61" />
```

Figure 31: ACMS HTML for creating a GridView (Source: [author])

ACMS has multiple navigation menus with links related to the functions and services provided by the adventure camping host company. The different navigation menu that is available in ACMS is explained here in brief.

Home: is the general introductory and welcome page to the end users. Whenever a user accesses the website from a remote location, the website redirects the user to the homepage. The home page has a search engine that is a global search engine. If a customer wants to access any component or functionality from the system, he can type in the field provided in the search bar. Then the populated list or the search result will provide a link to redirect the customer to the specified page on the website.

About: is the general information about the company history and background on the market. This will explain the adventure camping company formation, company locations, the company structure, their mission, and goals, etc.

Sale: is a page where customer finds all the available holiday packages. Figure 28 shows full packages, tents, back bags, and other different items. Customer on this page will be redirected to the details page to view more on the products and to select the quantity.

Detail: is the page that will explain the detail information about the package. Figure 29 shows detail page in ACMS website. This page gives a flexibility for customers to choose the size and the quantity of the item for their holiday seasons. This page enables customers to provide their reviews on the item or package listed. Customers can add many items as needed to their cart for their holiday trip. Some on-demand items that are not available on the sale list can be provided by adventure camping only on request by email or through a phone call to support desk.

Services: is a page that explains the services provided by the adventure camping host company. Some of these services are online support, onsite support, and call support. This is a page where customers find valuable information in the case of emergency or support at any time before or during the holiday seasons.

Web-based ACMS Sale Page

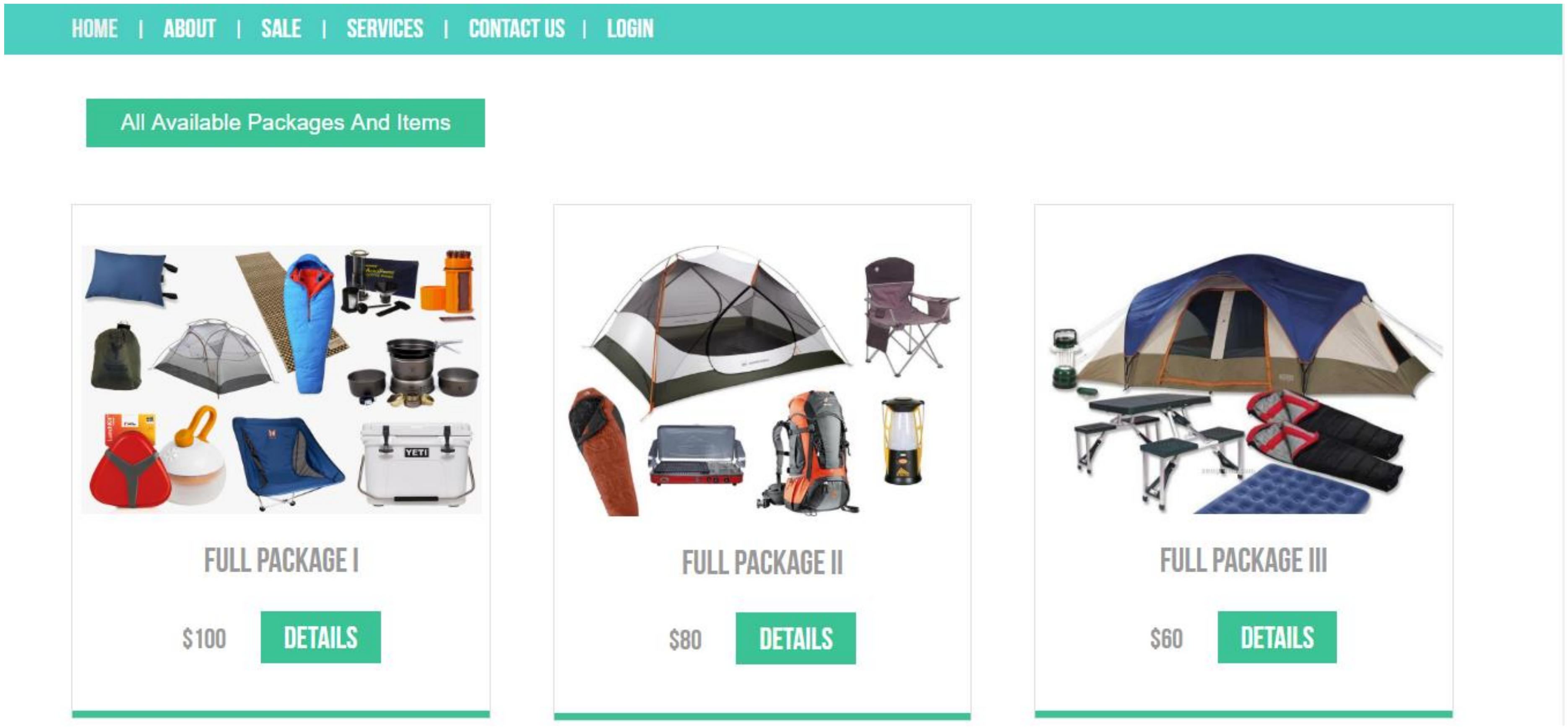
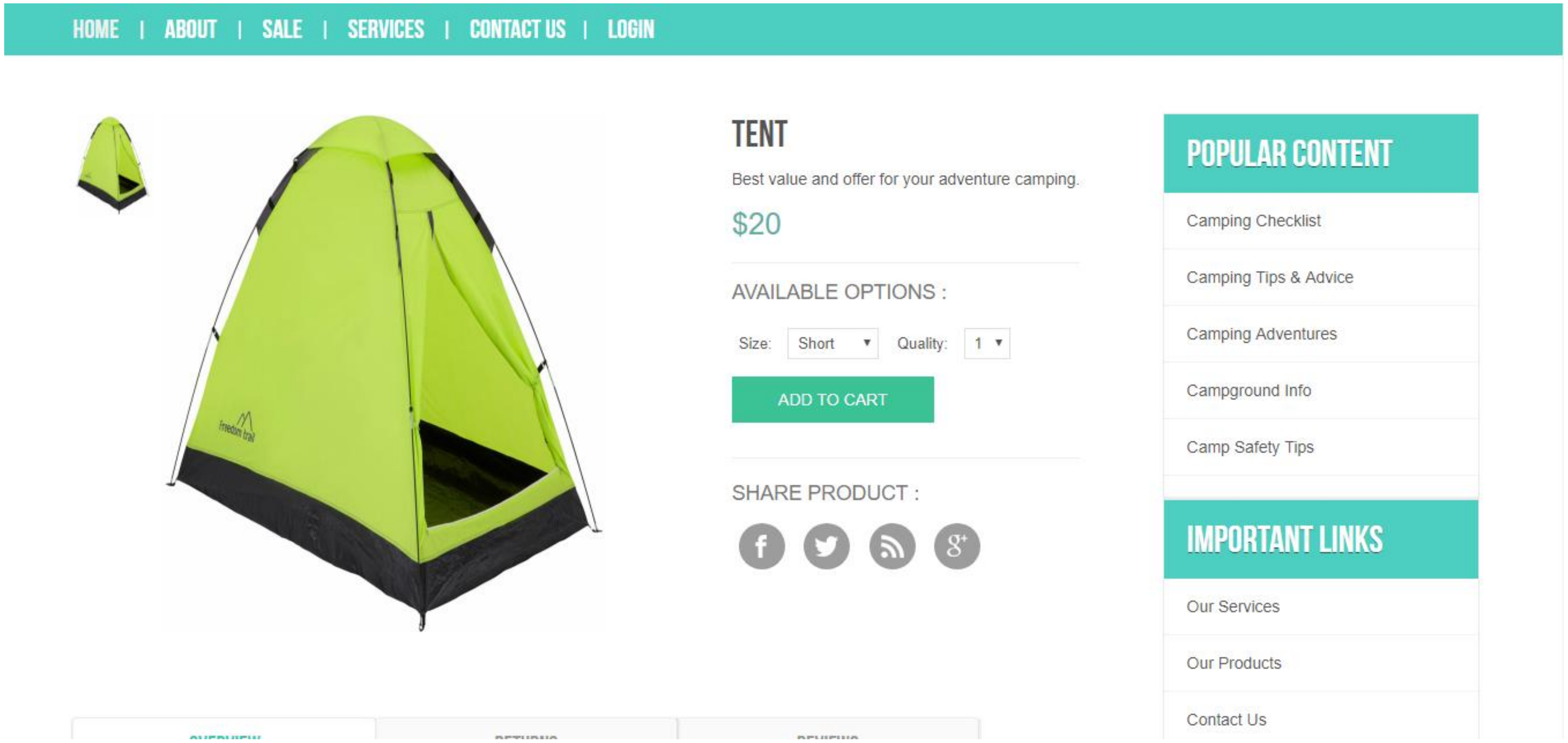

HOME | ABOUT | SALE | SERVICES | CONTACT US | LOGIN
All Available Packages And Items
FULL PACKAGE I
$100
DETAILS
FULL PACKAGE II
$80
DETAILS
FULL PACKAGE III
$60
DETAILS

Figure 32: Web-based ACMS sale page (Source: [author])

Web-based ACMS Detail Page



TENT

Best value and offer for your adventure camping.

$20

AVAILABLE OPTIONS :

Size: Short Quality: 1

SHARE PRODUCT :

POPULAR CONTENT

Camping Checklist

Camping Tips & Advice

Camping Adventures

Campground Info

Camp Safety Tips

IMPORTANT LINKS

Our Services

Our Products

Contact Us

Figure 33: Web-based ACMS detail page (Source: [author])

My Cart: is a page that lists the items and packages added to the cart. These items are displayed in a grid form that enables a customer to remove any items from the list. This page also includes to check out, to continue shopping or to save the list as a pending reservation.

Make Pending Reservation: this enables the customer to save the list of items for at least seven days, this will make the reservation as pending. This reservation must be paid within seven days otherwise the reservation will be expired and discarded from the list.

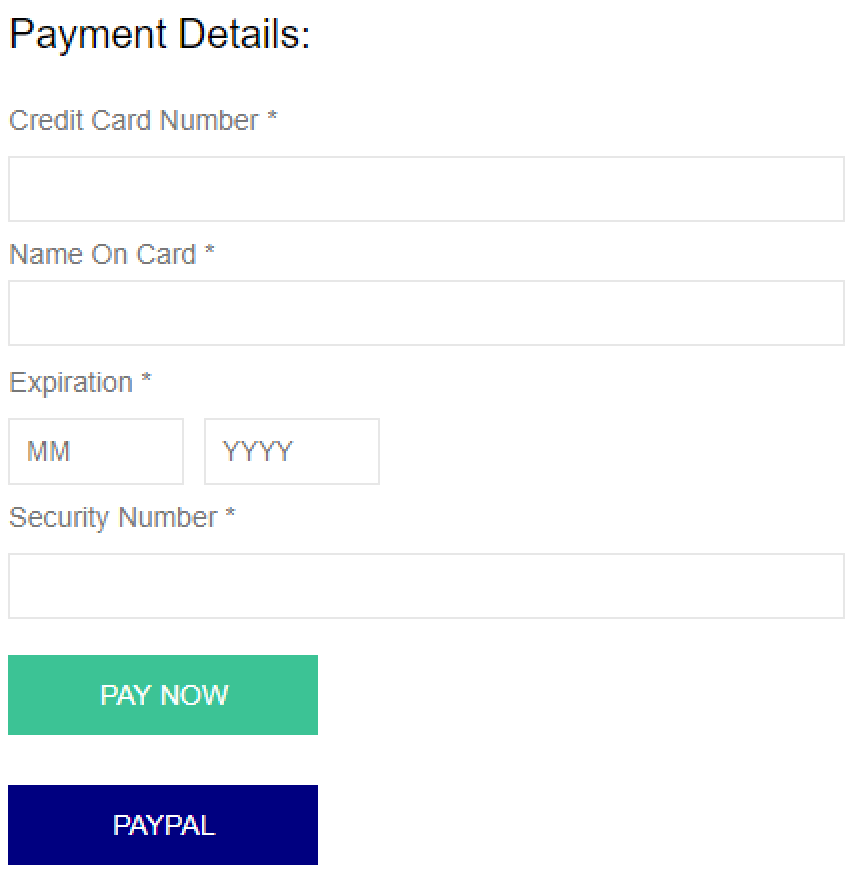
Checkout: This makes customers to check out the list of items in the cart. The customers will fill the billing details and the payment method available in the options. For this system there are only two types of payment method available, these are credit card and PayPal. PayPal payment will redirect to PayPal website to act as a gateway for credit card payment.

Register: is a page where customers register to track their order history, to make a pending reservation, and to make customize activity. Management uses registered customers to make the services more suitable and flexible. This also helps to manage a company's interaction with current and potential customers.

Login: is a page that enables administrators, employees, and customers to sign into the system. The administrators have ultimate control on the system. They can give access rights to employees and generate aggregated reports as needed. The customers need to register if they want to track the orders they have made. The employees need to login in order to perform their day to day tasks.

Contact Us: is where customers contact adventure camping host company by filling a form in detail and sending their request to support desk team. The support desk will reply to any of the customer requests as soon as possible. The contact us page is also a way to develop or improve the company services and products. This means it can be used for customers to provide their feedback and comments on their experience in adventure camping.

Web-based ACMS Checkout Page


Payment Details:

Credit Card Number *

Name On Card *

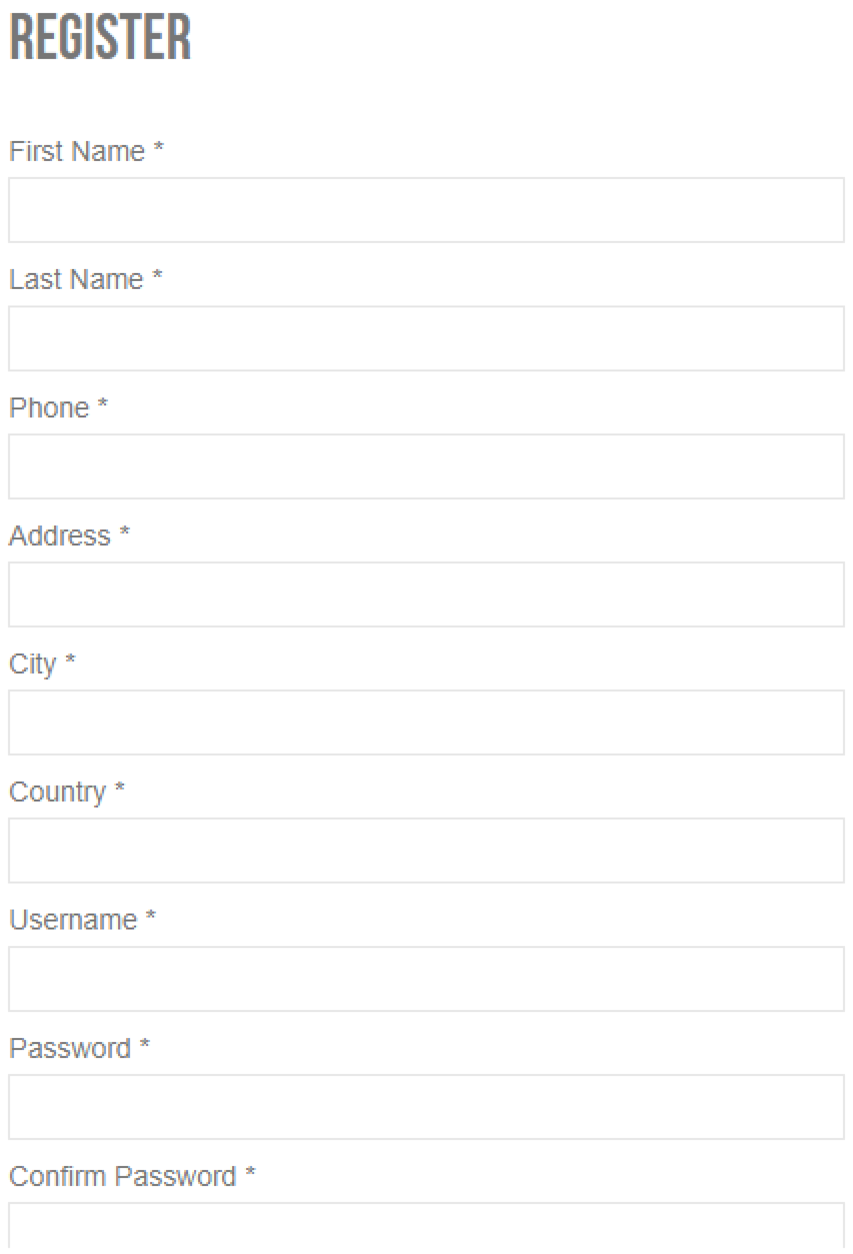
Expiration *

MM YYYY

Security Number *

Figure 34: Web-based ACMS checkout page (Source: [author])

Web-based ACMS Registration Page



REGISTER

First Name *

Last Name *

Phone *

Address *

City *

Country *

Username *

Password *

Confirm Password *

Figure 35: Web-based ACMS registration page (Source: [author])

Web-based ACMS Admin Page

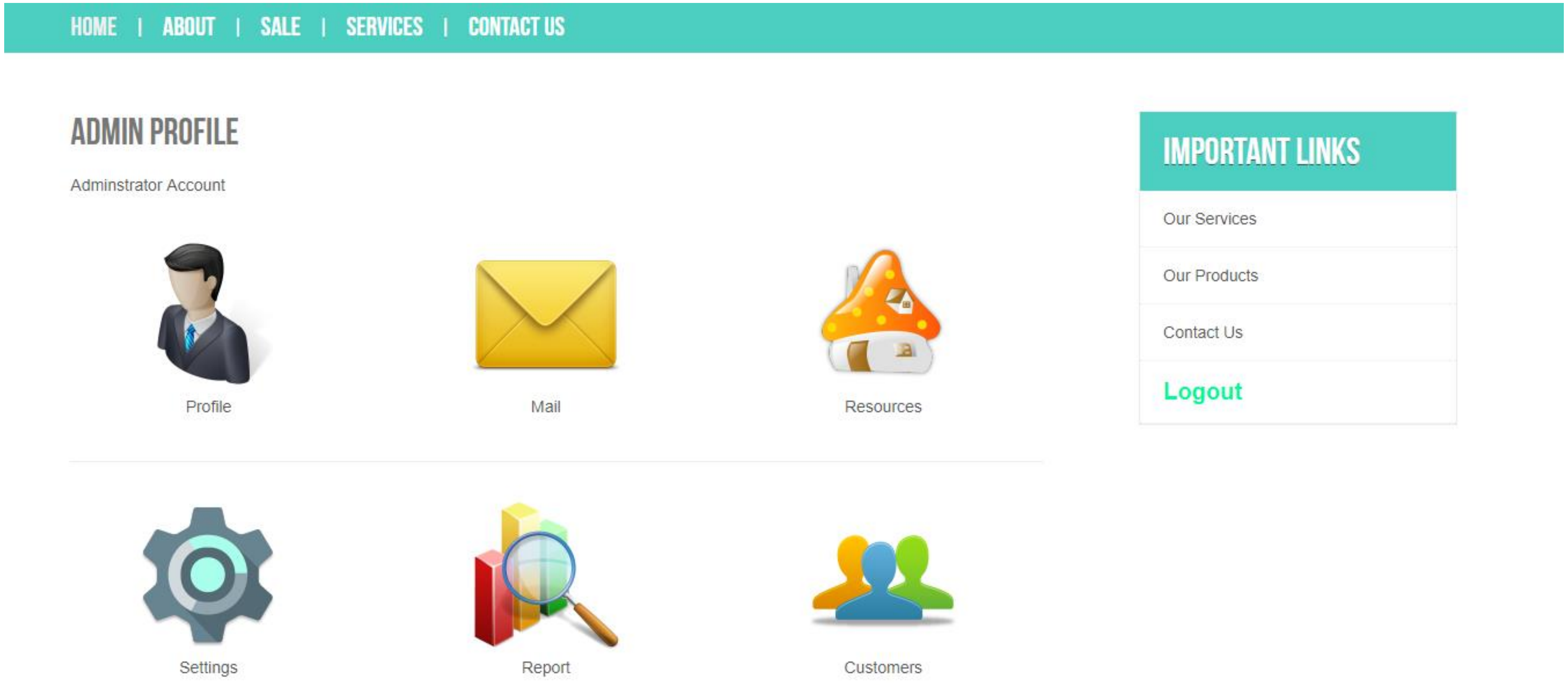


Figure 36: Web-based ACMS admin page (Source: [author])

In this section system implementation has been discussed. The implementation of Adventure Camping Management System done using ASP.NET Web Application [C#]. MSSQL Server 2016 database server used to store the data. The implementation shows how the UML design makes it easy for coding and putting all the process for the real user interaction with the system. The next section discusses how we migrate or deploy this application to the web server. This is the next process to go live and to start using the application in the real-world scenarios.

6. Evaluation and Discussion

6.1 Evaluation

This document explains in detail about information system design in UML. The key UML components have been addressed in brief with some illustration. UML enables a precise, unambiguous and complete specification of the proposed system. Possible to generate some of the code in a specified computer language directly from the UML model. For this purpose, a case study has been selected to use the modeling feature of UML and further coding using C# for implementation of a proposed system. The selected case study design and implementation are called adventure camping management system.

Adventure camping management system has nine modules. This document has discussed only the process of how a customer makes inquiries for the forthcoming holiday packages. The process starts with a customer makes an inquiry into the system. The reservation inquiries can be marked as confirmed reservation if the reservation is paid or can be marked as pending reservation if it is saved for payment within seven days. The reservation clerk goes with all reservation and creates a booking note for confirmed reservations. The booking note will be sent to site supervisor in order to prepare the requested packages for the customer. Additional update information will deliver to reservation from management.

Basic UML diagrams used for the analysis of adventure camping management system. Each phase of the modeling process enabled to capture what should be done and how to do it. Modeling accelerates achieving of consensus among developers and experts. Use case and activity diagram used in the first phase of requirement analysis to depict the system behavior.

This document shows general use case diagram for the whole reservation process, which represent the main functionality. Conceptual model has followed next for the web-based adventure camping management system. Conceptual model provides a visual specification of the domain relevant information for the web-based application. Sequence diagrams used to model the sequence of behavior from a perspective of a system user. Activity diagrams also included showing a control flow between individual steps of computation. Finally, state diagrams used to show the relationship between events and states.

A website has been implemented for adventure management system using ASP.NET [C#]. This implementation helps a customer to reserve their holiday packages. They can save the reservations as pending reservations. There is also a registration page if a customer wants to follow their orders and to keep other personal details and manages the reservations. Reservation clerk is an employee, which will get login credentials from the administrator. The administrator has the ultimate role of managing users, customers, reports, and so on. This is a prototype application that shows all the life cycle that UML supports.

Finally, UML modeling makes all the information system life cycle easy, flexible and understandable. The UML is a language used to make software blueprints. UML is not a programming language but tools can be used to generate code in various software developments. The direct relationship between UML and object-oriented analysis and design makes it easier and more flexible.

6.2 Discussion

In computing, a web application or web app used a lot in nowadays technological advancement. Most organization shifting or moving their entire infrastructure to cloud in a client-server computer program. The client accesses any of the services of a company in the web using a web browser. A web app is a contrast with traditional desktop applications, which are installed and configured on a local machine.

Many organizations claim benefits from their use of UML, ranging from improved communication among developers to productivity gains using code generation from UML models. UML is a good language to describe software systems at a higher level than code. Properly used UML can improve productivity and quality. This document explains well in showing the whole process models linked with the implementation. UML diagrams helped to capture the general overview of the system, its boundaries and the interconnection between them. In simple words, Adventure Camping Management System implemented accordingly or based on the models created prior to the implementation phase. As the functionality of UML is broad and huge, the key diagrams and implementation process used to design a particular information system.

Best practice is so important in various aspects, especially in IT. I would recommend using the emerging cloud technologies as a best practice for implementation and deployment of Adventure Camping Management System. Organizations are moving their entire infrastructure from on-premises to public or hybrid clouds. This is helping them a lot by saving their up-front and other related costs. No organization wants to pay excess for the resources they are not used, that is the cloud comes into play. Pay for what you used. It may be infrastructure, or platform, or software services used in the cloud, it gives more flexibility. Vendors like Microsoft uses a full collection set that helps developers in making applications and deployment for a particular software project. I used Visual Studio 2015 and SQL Server 2016 but it is also possible to use both tools from Microsoft Azure without installing any application. The URL provided by Azure makes it easy to go live with the completed software application.

Finally, this document has discussed the current technological advancement in the information system and how we can use UML with the integration of these technologies. In an organization like adventure camping, host company can use cloud computing using SaaS service tier by only focusing the business architecture of the management system. The rest can be handled by the third-party cloud service providers. This will help the company to provide the maximum services to the end users.

7. Conclusion

This thesis document has covered a major concept that plays a vital role in the current technological world, this is Information System Design in UML. An information system has been covered before going into UML and its features. An information system is the core part of any particular organization. All organization activities depend on the data and information flows within the intranet or outside to the public. For this purpose, it needs to design and model an information system using a standard modeling tool, this known as Unified Modeling Language (UML).

Users and vendors want UML to be more expressive, simple, not difficult to integrate with other modeling and programming languages, and more significant to today’s technologies. For example, some users want to use UML to drive the interactive behavior of a mobile application. Increasingly, users would like to use UML to model applications deployed in the Cloud like Software as a Service (SaaS). To address these needs, the OMG and UML vendors are working together towards making UML smarter and more agile.

One of the biggest complaints about UML is that it is too large and too complex. Typically, a project that uses UML only uses 20% of the specification to help build 80% of the code and other deliverables. To address this complaint, UML should be described differently for different user groups. There are ordinary users such as analysts, designers, website builders, database designers, developers, operators, architects, and testers, each bringing a different but valid perspective that uses different but overlapping subsets of UML.

Key UML diagrams explained and used to model a case study that has been selected for this thesis. Adventure camping management system is a web-based application that supports the ongoing day-to-day activities of the adventure camping host company. The UML model has been used for coding and implementation using ASP.NET [C#]. The document has focused on designing and implementing how a customer makes inquiries for the forthcoming holiday packages.

Finally, this document discussed all the major components listed in the objective part of this thesis document. This document also answered some of the important questions like What is UML? Why UML modeling? What is the road ahead for UML? How to use UML specifications and diagrams for coding and development of deliverables? What system development methodologies used for software development? This research document covered and addressed all the concepts in detail. But as the technologies in the IT world are moving fast forward, further research and studies need to be carried out in order to use the best out of it. Organizations need to integrate their entire system with the new technologies and tools to provide effective and efficient services to customers. This can be viewed from many perspectives. Security and performance are the top listed elements. This document can be used as a reference for further research and study in these areas, Information Systems, Object-Oriented, UML and System Development.

8. Bibliography

[1] Almendros-Jimenez, J. M. and Iribarne, L., 2006. Describing Use-Case Relationships with Sequence Diagrams. *The Computer Journal, 50*(1), p. 116-128

[2] Ambler, S. W., 2004. *The Object Primer: Agile Model-Driven Development with UML 2.0,* Cambridge, England: Cambridge University Press.

[3] Arlow, J. and Neustadt, I., 2005. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design,* (2nd ed.). Boston, MA: Addison Wesley.

[4] Bennett, S., McRobb, S. and Farmer, R., 2005. *Object-Oriented Systems Analysis and Design: Using UML,* (3rd ed.). New York, NY: McGraw-Hill.

[5] Booch, G., 2007. *Object-Oriented Analysis and Design with Applications*. (3rd ed.). Boston, MA: Addison Wesley.

[6] Booch, G., Rumbaugh, J. and Jacobson, I., 1998. *The Unified Modeling Language User Guide*, Boston, MA: Addison Wesley

[7] Buyya, R., Broberg, J. and Goscinski, A., 2011. *Cloud Computing: Principles and Paradigms,* Hoboken, NJ: John Wiley & Sons, Inc.

[8] Cantor, M., 1998. *Object-Oriented Project Management with UML,* Hoboken, NJ: John Wiley & Sons, Inc.

[9] Chandrasekaran, K., 2015. *Essentials of Cloud Computing*, Boca Raton, FL: CRC Press

[10] Conallen, J., 2002. *Building Web Applications with UML*, Boston, MA: Addison Wesley

[11] Delamater, M. and Boehm, A., 2016. *Murach's ASP.NET 4.6 Web Programming with C# 2015*, Fresno, CA: Mike Murach & Associates, Inc.

[12] Erl, T., Mahmood, Z. and Puttini, R., 2013. *Cloud Computing: Concepts, Technology & Architecture,* Upper Saddle River, NJ: Prentice Hall

[13] Fowler, M. and Scott, K., 1999. UML Distilled: *A Brief Guide to the Standard Object Modeling Language*, (2nd ed.). Boston, MA: Addison Wesley

[14] Hall, G. M., 2014. *Adaptive Code via C#: Agile Coding with Design Patterns and Solid Principles*, Redmond, WA: Microsoft Press

[15] Hamilton, K. and Miles, R., 2006. *Learning UML 2.0*, Boston, MA: O'Reilly

[16] Hansson, D. H. and Thomas, D., 2005. *Agile Web Development with Rails: A Pragmatic Guide (Pragmatic Programmers)*, Raleigh, NC: Pragmatic Bookshelf

[17] Hubbell, T. *Top 4 Software Development Methodologies*[online]. [Accessed 14 October 2017]. Available at: https://blog.blackducksoftware.com/top-4-software-development-methodologies

[18] Jacobson, I. and Cook, S. *The Road Ahead for UML* [online]. [Accessed 10 January 2018]. Available at: http://www.drdobbs.com/architecture-and-design/the-road-ahead-for-uml/224701702

[19] Jakimi, A. and Koutbi, M. E., 2009. An Object-Oriented Approach to UML Scenarios Engineering and Code Generation. *International Journal of Computer Theory and Engineering*, *1*(1), 1793-8201.

[20] Karim, A. J., 2011. The Significance of Management Information Systems for Enhancing Strategic and Tactical Planning. *Journal of Information Systems and Technology Management*, *8*(2), p. 459-470.

[21] Larman, C., 2001. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, (2nd ed.). Upper Saddle River, NJ: Prentice Hall

[22] Lee, S., 2012. Unified Modeling Language (UML) for Database Systems and Computer Applications. *International Journal of Database Theory and Application*, 5(1), p. 157-163

[23] Miller, M., 2009. *Cloud Computing: Web-Based Application that Change the Way You Work and Collaborate Online*, Indianapolis, IN: Que Publishing

[24] MIT Press. *Information Systems: Introduction and Concepts* [online]. [Accessed 13 September 2017]. Available at: https://mitpress.mit.edu/sites/default/files/titles/content/9780262015387_sch_0001.pdf

[25] Nazir, M., 2012. Cloud Computing: Overview & Current Research Challenges. *IOSR Journal of Computer Engineering*, *8*(1), pp. 14-22

[26] O'Brien, J.A. and Marakas, G.M., 2011. *Management Information Systems*, (10th ed.). New York, NY: McGraw-Hill.

[27] OMG, 2017. *OMG Unified Modeling Language (OMG UML): Infrastructure specification, Version 2.5.1*, formal/2017-12-05, s.l.: Object Management Group.

[28] Penberthy, W., 2016. *Beginning ASP.NET for Visual Studio 2015*, Hoboken, NJ: John Wiley & Sons, Inc.

[29] Peppard, J. and Ward, J., 2016. *The Strategic Management of Information Systems: Building a Digital Strategy,* (4th ed.). Hoboken, NJ: John Wiley & Sons, Inc.

[30] Perkins, B., Hammer, J. V. and Reid, J. D., 2016. *Beginning C# 6 Programming with Visual Studio 2015*. Fresno, CA: Mike Murach & Associates, Inc.

[31] Pilone, D. and Pitman, N., 2005. *UML 2.0 in a Nutshell*, Boston, MA: O'Reilly

[32] Rajagopal, D., and Thilakavalli, K., 2017. A Study: UML for OOA and OOD. *International Journal of Knowledge Content Development & Technology*, *7*(2), p. 5-20.

[33] Rosenberg, D. and Stephens, M., 2007. *Use Case Driven Object Modeling with UML: Theory and Practice*, New York, NY: Apress.

[34] Rumpe, B., 2016. *Modeling with UML: Language, Concepts and Methods*, Cham, Switzerland: Springer.

[35] Tewabe, W. *The downside of UML as a Complete Modeling Tool for Software Design* [online]. [Accessed 13 January 2018]. Available at: https://www.academia.edu/5149888/The_downside_of_UML_as_a_Complete_Modeling_Tool_for_Software_Design

[36] Tutorials Point (I). *Object Oriented Analysis & Design* [online]. Available at: https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_tutorial.pdf [Accessed 22 November 2017].

[37] Visual Paradigm. *What is Unified Modeling Language (UML)?* [online]. [Accessed 30 September 2017]. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#use-case-diagram

[38] Vrana, I., 2016. *Projecting of Information Systems with UML*, Prague, Czechia: Faculty of Economics & Management, CULS Prague.

[39] Wazlawick, R. S., 2014. *Object-Oriented Analysis and Design for Information Systems: Modeling with UML, OCL, and IFML*, Boston, MA: Elsevier

[40] Wrycza, S., 2016. *Information Systems: Development, Research, Applications, Education*, Cham, Switzerland: Springer.

[41] Xue, C. T., and Xin, F. T., 2016. Benefits and Challenges of The Adoption of Cloud Computing in Business. *International Journal on Cloud Computing: Services and Architecture*, *6*(6), 1-15