PAVEMENT PERMIT SYSTEM INFRASTRUCTURE: UML BASED DESIGN

BY

Rajesh Prasad

BACHELOR OF TECHNOLOGY IN CIVIL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, 1990

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Engineering
IN CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE
Massachusetts Institute of Technology
JUNE 2001

SIGNATURE OF AUTHOR: [illegible]
RAJESH PRASAD
DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
MAY 11, 2001

CERTIFIED BY: [illegible]
GEORGE KOCUR, PH. D.
SENIOR LECTURER, CIVIL AND ENVIRONMENTAL ENGINEERING
THESIS SUPERVISOR

CERTIFIED BY: [illegible]
ORAL BUYUKOZTURK, PH. D
CHAIRMAN, DEPARTMENTAL COMMITTEE ON GRADUATE STUDIES

PAVEMENT PERMIT SYSTEM INFRASTRUCTURE: UML BASED DESIGN

By
RAJESH PRASAD

Submitted to the Department of Civil and Environmental Engineering Department on May 11, 2001,
in partial fulfillment of the requirements for the degrees of Master Of Engineering in Civil and Environmental Engineering

Abstract

Distributed systems have complex interactions that are difficult to understand. This is true of any large application. Vague system specifications and poor designs are a still major problem even in today's Internet era. A failure to follow specifications or poor design can introduce errors with very severe results. Therefore there is a great need for using efficient specifications and design methods.

UML, the Unified Modeling Language, is viewed as the answer to this problem. It simplifies the complex process of software design, making a "blueprint" for construction. In this thesis, I apply the UML to the modeling of requirements and design for a Pavement Permit System project.

The Pavement Permit System is a web-based application for automating the permit application and issuance process for the Public Works Department in the Town of Arlington, MA. This system allows utilities, such as gas or phone companies, as well as independent contractors, to obtain permits over the Internet to do work in public streets. It performs a variety of checks before granting a permit. Besides this it provides other capabilities like restricting streets, generating billing data and performing other manipulations of administrative data.

This thesis explores the issues related to the use of UML as applied to the Pavement Permit System design and development. As a case study we applied UML based techniques to the requirements specification and design of the Permit System. There were significant benefits derived from this. It provided for a better understanding of the requirements leading to a clearer requirements specification and subsequent design. In the end the software developed for the application reflected the design very well and there was a relatively smooth integration of the components and timely release of the product.

Thesis Supervisor: G. Kocur
Title: Senior Lecturer

Acknowledgments

I would like to thank Dr. George Kocur for initiating the fabulous Pavement Permit System Project for our Master of Engineering Project. The project not only helped us understand large projects, but also provided a strong basis for this thesis. He constantly provided us his helping hand during the course of execution of the project and also during the writing of this thesis. I am very grateful for his help and encouragement, without which, I am certain, we would not have been able to bring the project and thesis to a successful culmination.

I would also like to thank Mr. Ron Santosuosso and the other staff of the Department of Public Works, Arlington, for their help and cooperation during the Pavement Permit System project. They made a great contribution towards the completion of our project.

I am very grateful to the professors and the teaching assistants in our department for their inputs during the year, in making me an IT professional. I would also like to thank my colleagues in the M.Eng. 2001, for their help during the course of the year.

Finally, I would like to thank my parents, my wife, my little daughter, Akansha, my brother, his wife, my in-laws and also my uncle and his family for their great support during this year. Without their love and affection, I would not have successfully completed the course.

Table of Contents

1 Introduction 13

1.1 Overview 13

1.2 Preamble 13

1.2.1 The Problem statement 13

1.2.2 Object-oriented view 14

1.2.3 The UML and e-World 15

1.3 Motivation 16

1.4 Thesis Overview 17

2 Object Oriented Paradigm 19

2.1 Object-Oriented Concepts And Principles 19

2.2 Object-Oriented Analysis 20

2.3 Object-Oriented Design 21

2.4 Conventional Vs. OO Approaches 22

3 UML: A Theoretical Perspective 25

3.1 UML-Overview 25

3.2 UML: Modeling Concepts 25

3.3 UML Meta-Model 26

3.4 UML Meta-Model: Terminology 28

3.4.1 Use cases 28

3.4.2 Use case Diagram 29

3.4.3 Use Case Relationships 31

3.4.4 Generalization 31

3.4.5 Extend 31

3.5 Class Diagrams 32

3.5.1 Association 34

3.5.2 Attributes 34

3.5.3 Operations 34

3.5.4 Constraints Rules 36

3.5.5 When to Use Class Diagrams 36
3.6 Interaction Diagrams 37
3.6.1 Sequence Diagrams 37
3.7 Component Diagram & Deployment Diagram 39
3.8 Collaboration Diagram 42
3.9 Activity Diagram 42
3.10 State Diagrams 44
4 Pavement Permit System 46
4.1 Problem Description 46
4.2 Requirements of the Proposed System 47
4.2.1 Contractors and Utility Companies (External Users) Perspective 47
4.2.2 The internal user's perspective 48
4.2.3 General Perspective 49
5 Pavement Permit System – Use Cases 50
5.1 Overview 50
5.2 Use Cases 51
5.2.1 Use Case for Accessing Permits menu – Internal User 51
5.2.2 Use Case for making a new application- Internal User 52
5.2.3 Use-Case for searching permits by Internal User 54
5.2.4 Use Case to view permits issued within last 30 days by Internal User 55
5.2.5 Use Case for editing an existing permit 55
5.2.6 Use Case for deleting an existing permit 56
5.2.7 Use-Case on street restrictions administration 56
5.2.8 Use-Case on street restriction addition: 56
5.2.9 Use-Case on street restriction editing 57
5.2.10 Use-Case on street restriction deleting: 57
5.2.11 Use case on listing all current street restrictions 58
5.2.12 Use case on listing all street restrictions in the entire history 58
5.2.13 Use case on Companies Administration 58
5.2.14 Use-Case on company/utility company addition: 59

5.2.15 Use-Case on contractor/utility company editing ... 59
5.2.16 Use-Case on company/utility company deleting: ... 60
5.2.17 Use case on listing all contractor/utility company: ... 60

6 UML Application To Pavement Permit System ... 61

6.1 Use Case Diagram ... 61
6.2 Class Diagrams ... 62
6.2.1 Permit Class ... 63
6.2.2 Company Class ... 65
6.2.3 Restriction Class ... 66
6.2.4 Holiday Class ... 66
6.2.5 Option Class ... 67
6.2.6 Report Class ... 68
6.2.7 FAQ Class ... 69
6.2.8 Class Diagram for Pavement Permit System ... 70
6.3 Sequence Diagram ... 73
6.3.1 Sequence Diagram For Adding A Street Restriction ... 73
6.3.2 Sequence Diagram For Deleting Data On A Holiday ... 74
6.3.3 Sequence Diagram For Permit Issuance ... 75
6.4 State Diagrams ... 75
6.5 Component Diagrams ... 76

7 Conclusions ... 78

8 Appendix ... 79

9 References ... 98

List of Figures

Figure 3-1 UML Meta Model 27
Figure 3-2 Use Case Diagram - Features 29
Figure 3-3 Use Case Diagram - Example 30
Figure 3-4 Actor In Use case 31
Figure 3-5 Use Case: Extends Diagram 32
Figure 3-6 Class Diagram Concepts (Source: UML™ Quick Reference for Rational Rose) 33
Figure 3-7 Class Diagram Interactions - Example 33
Figure 3-8 Class Example 35
Figure 3-9 Class Diagrams – Other Concepts (Source: UML™ Quick Reference for Rational Rose) 37
Figure 3-10 Sequence Diagram Concepts (Source: UML™ Quick Reference for Rational Rose) 38
Figure 3-11 Sequence Diagram Example 39
Figure 3-12 Component and Deployment Diagram (Source: UML™ Quick Reference for Rational Rose) 40
Figure 3-13 Component Diagram Example 41
Figure 3-14 Deployment Diagram Example 41
Figure 3-15 Collaboration Diagram 42
Figure 3-16 Activity Diagram Example (Based on UML Distilled, Martin Fowler) 43
Figure 3-17 State Diagram Example (Based on UML Distilled: Martin Fowler) 44
Figure 4-1 Paper Based Permit 46
Figure 5-1 Use-case Diagram: Access Permits Menu 51
Figure 5-2 Apply For New Permit Use-Case Diagram 52
Figure 6-1 Use Case Diagram: Permit Application 61
Figure 6-2 Permit Class 64
Figure 6-3 Company Class 65
Figure 6-4 Restrictions Class 66

Figure 6-5 Holiday Class........67
Figure 6-6 Option Class........68
Figure 6-7 Report Class........69
Figure 6-8 FAQ Class........70
Figure 6-9 Class Diagram-Pavement Permit System........71
Figure 6-10 Pavement Permit System Data-Model (Source: Web Application Development Using Open Source Technologies and Java by Wolfgang Andreas Klimke)........72
Figure 6-11 Sequence Diagram For Adding A Street Restriction........73
Figure 6-12 Sequence Diagram For Editing Company Data........74
Figure 6-13 Sequence Diagram For Deleting Data On A Holiday........74
Figure 6-14 Sequence Diagram For Permit Issuance........75
Figure 6-15 Permits: State Diagram........76
Figure 6-16 Component Diagram........77
Figure 8-1 Pavement Permit System: Internal User Main........79
Figure 8-2 Companies Main Menu........80
Figure 8-3 Permits Main menu........81
Figure 8-4 Restrictions Main Menu........82
Figure 8-5 Holiday Restrictions Main Menu........83
Figure 8-6 Miscellaneous Options........84
Figure 8-7 Export Billing data to MS-Excel........85
Figure 8-8 Reports on Permits........86
Figure 8-9 FAQ Menu........87
Figure 8-10 Street Opening Permit System Help........88
Figure 8-11 About The Software........89
Figure 8-12 Complete Permit Application........90
Figure 8-13 Login Screen for External Users........91
Figure 8-14 List of Companies........92
Figure 8-15 Add a Restriction........93
Figure 8-16 Edit a Holiday Information........94
Figure 8-17 Delete the FAQ........95

Figure 8-18 Billing Data 96
Figure 8-19 Reports on Permits 97

1 Introduction

1.1 Overview

This thesis examines the application of the UML based design techniques to software development. We use the Pavement Permit System as a case study for the application of the UML techniques. This case study helps us better understand the abstract concepts of object-oriented design and of UML.

1.2 Preamble

1.2.1 The Problem statement

Worldwide, there is an insatiable demand for software. The advent of the Internet has fuelled the demand further. The future of software can be reasonably predicted from the visible trends that it's going to be more complex and it's going to be far more distributed. It's going to be far more complex mainly because of demand-pull and supply-push, to use an economic analogy. User expectations for what software can do are exceedingly high. Furthermore, what is possible is far greater now than just a few years ago. Although the cost of computing has plummeted, yet the cost and complexity of software development have continued to increase. Future systems will be far more distributed for similar reasons.

The new e-world brings a lot of power and has created exciting times for professional software developers, but that power comes at a high development cost. Concurrent, distributed systems have extremely complex interactions that can be hard to understand, let alone predict. Vague specifications are a major problem. In the past, the specifications for a monolithic system only affected the single system, and if it didn't work exactly as specified, the impact was limited to just that system. But now a business system may have to interoperate with another system halfway around the world, both of which may be written by people who have never heard of each other. A failure to follow specifications can introduce errors that propagate around the world. One cannot take a snapshot of a distributed system for backup or reboot it if part of it fails. The entire system must keep going in spite of failures, errors, or data corruption of some of its parts. Most systems are now real-time (online rather than batch) systems. Timing concerns matter a great deal to customers and partners. On top of everything else, performance of complex systems is often nonlinear and cannot be predicted by simple extrapolation.

Although it has created exciting times for the professional software developers, the trends show that no amount of heroic programming will ever suffice to meet this demand. Furthermore, as software continues to weave itself deeply into the fabric of society, the stakes have become higher. Moreover, it is still very true that building complex software of quality and of scale is still fundamentally a hard problem. Simply put, this means that, all the latest technology trends notwithstanding, deploying quality software is still an engineering problem. Moreover, unfortunately, software bugs are still considered just a normal part of the territory, but now, they may manifest themselves in the fall of a business or even worse, in the loss of a human life.

The challenge then is what can one do about it? There is a powerful school of thought that believes that one can use the same kinds of approaches available to engineers in every field: modeling before construction, architecture based on experience, a process based on best practices, building with reusable components, and the use of tools to leverage the developer's time and skill. This is manifested in the need for a practical and effective tool to help the designers to take on the challenges of the modern day software development. It is here that the modern day concept of object-oriented approach and UML based design comes in very handy.

1.2.2 Object-oriented view

Object-oriented analysis and design, through its concepts and methods, provides many advantages to today's software development environment. The most obvious advantage of object-oriented design and analysis is its ability and ease of reuse of objects from other designs. Structural or functional design techniques fail miserably in the category of reuse of code and design. The reason being that real world problems differ substantially in structure and function. However, many real world problems are composed of the same objects and those objects are expected to maintain their properties and behavior no matter the domain.

Object-Oriented Software Architecture (OOSA) changes the way one thinks about systems. The OOSA way of thinking is more natural for most people than the techniques of structured analysis and design. End users and business people think naturally in term of objects, events, and triggers. We can create Object-Oriented (OO) diagrams that they can relate to and thus make the design process more comprehensible and effective

In object-oriented styles, architectures are composed of objects that contain and manage their own internal data and provide well-defined interface services that allow other objects to use them. Object-oriented diagrams show collections of objects and the way each object makes use of the services provided by other objects.

1.2.3 The UML and e-World

The Unified Modeling Language is designed to be distributed, concurrent, and connected. It is well suited to the new demands of the brave new e-world. It is based on objects. Objects are distributed -- each one maintains its own state, distinct from all others. They are concurrent -- each one can potentially execute in parallel with all others. Also they are connected -- each one can send messages to others through a Web of links. UML is not associated with one platform or programming language; therefore it is well suited to bridge networks of different systems. UML was designed with extensibility in mind, so it can adapt to new issues as and when they arise.

The UML as a modeling language is evolutionary and general-purpose, has broad applicability, is tool-supported, and is now the industry-standard for specifying, visualizing, constructing, and documenting the artifacts of a system-intensive process. The language is broadly applicable to different types of systems software and non-software, domains i.e. business versus software, and methods and processes. The UML enables and supports (but does not require nor mandate) a use-case-driven, architecture-centric, iterative, and incremental process that is object oriented and component based.

The UML enables the capturing, communicating, and leveraging of knowledge. The models capture knowledge i.e. semantics, the architectural views organize knowledge in accordance with guidelines that express idioms of usage, and diagrams depict knowledge of syntax- for communication.

1.3 Motivation

We are developing a web-based application for the Public Works Department of the town of Arlington, MA, the "Pavement Permit System", which essentially aims to automate the street opening permit application and processing system for the department. This system allows utilities, such as gas, electric or phone companies, as well as independent contractors, to obtain permits over the Internet that allow them to do work in public streets. It performs a variety of checks on whether a permit should be granted. It provides the capability to restrict streets and times where permits are issued; generate the data required for billing pertaining to permits that were issued and allows management of the overall permits process.

It is a very typical software application that is representative of the modern day trends in the software development. The application is of a reasonable size and had to be developed in a fairly short span of time by a small team.

As discussed earlier, proper requirements specification and design is essential to the development of any software application and UML is a very important tool towards that end.

For me, this project was the first real world large software assignment. I understand the project very well. Therefore, I felt that I should study the application of the principles of UML based design for this software application, which will greatly help me understand the bigger picture of the application of object oriented design and application of UML techniques.

From my work experience in the oil industry, I know that formalized planning and design are crucial to the successful execution of any project. Although software projects are a relatively new venture for me, I believe that the basic premises mentioned above do remain the same for the software projects too. Therefore, in order to be able to take future challenges in the software sphere, it was necessary that I understand the application of object-oriented paradigm and application of UML. These reasons combined with the factors mentioned above motivated me to write the thesis on this topic.

1.4 Thesis Overview

There are a total of seven chapters in this thesis. We provide a brief introduction to each of the chapters below.

Chapter One is an introduction to the thesis. In this chapter, we provide an overview of the thesis, along with the motivation for writing it. We also present the structure of the thesis.

Chapter two deals with a theoretical input on the abstract concepts of the object-oriented paradigm. Since, this forms a very important background for the use of UML, we discuss the paradigm in some detail therein

Next in chapter three, we provide the theoretical principles of UML itself. We discuss the concepts that form the UML to provide a framework for the case study.

In chapter four, we introduce the case study itself, i.e. the Pavement Permit System, the problem it solves, its requirements premises and so on. The intent is to provide the background for application of the UML techniques to it.

In chapter five we provide the representative use-cases for the Pavement Permit system, as derived from the requirements specification.

In chapter six, we actually apply the principles and techniques of UML to the Pavement permit system. The chapter contains various diagrams that were derived by applying the UML techniques to the case study, along with explanations.

In chapter seven, we provide our conclusions for the thesis. We also provide an appendix, which shows the screen shots of representative portions of the actual system.

2 Object Oriented Paradigm

The UML based design techniques are basically derived from the object oriented concepts and principles. Here we give a brief description on the object-oriented paradigm. (The material in this chapter is drawn, in part, from Software Engineering A Practitioner's Approach: Roger S Pressman)

2.1 Object-Oriented Concepts And Principles

There are several ways to look at problems that have to be solved using a software-based solution. One approach that is widely used for problem solving is the object-oriented viewpoint. The objects are manipulated with a collection of functions (called methods, operations, or services) and they communicate with each other through a messaging protocol. Objects are categorized into classes and subclasses.

The definition of object consists of describing its attributes, behaviors, operations, and messages. An object encapsulates both data and the processing that is applied to the data. This characteristic is very important as it enables classes of objects to be built and therefore inherently leads to libraries of reusable classes and objects. As reuse is a critically important attribute of modern software engineering, the object-oriented paradigm has become very is attractive to many software development organizations. Additionally, the software components derived using the object oriented paradigm show design characteristics (e.g. functional independence, information hiding) that are essential ingredients of high-quality software.

Object-oriented software engineering follows the same steps as conventional approaches. Analysis identifies objects and classes that are relevant to the problem domain: design provides the architecture, interface, and component-level detail: implementation (using an object-oriented language) transforms design into code; and testing exercises the object-oriented architecture, interfaces and components.

The product of this effort is a set of object-oriented models. These models describe the requirements, design, and code and test process for a system or product. In order to ensure the correctness of the process, at each stage, object-oriented work products are reviewed for clarity, correctness, completeness and consistency with customer requirements and with one another.

2.2 Object-Oriented Analysis

Before one can build an object-oriented system, one needs to define the classes (objects) that represent the problem to be solved, in the manner in which the classes relate to and interact with one another, the inner workings i.e. attributes and operations of objects, and the communication mechanisms i.e. messages that allow them to work together. All of these things are achieved during object-oriented analysis (OOA).

The definition of an object oriented analysis model consists of a description of the static and dynamic characteristics of classes that describe the system or product.

OOA is important because one cannot build software (object-oriented or otherwise) until we have reasonable understanding of the system or the product. OOA provides with a concrete way to represent ones understanding of requirements and then test that understanding against the customer's perception of the system to be built.

OOA usually begins with a description of use-cases – a scenario based description of how actors (people, machines, other systems) interact with the product to be built. "Class-responsibility-collaborator modeling "or other methods translate the information contained in use–cases into a representation of classes and their collaborations with other classes. The static and dynamic characteristics of classes are then modeled using the Unified Modeling Language (or, more rarely, some other method).

The work product of OOA is an analysis model. The analysis model is consists of graphical or language-based representations that define the attributes of classes, relationships between classes, their behaviors as well as inter class communication and a description of class behavior over time.

At each stage the element of the object-oriented analysis model are reviewed for clarity, correctness, completeness and consistency with customer requirements and with one another.

Object oriented design transforms the analysis model created using object-oriented analysis into a design model that serves as a blueprint for software construction.

2.3 Object-Oriented Design

The design of object oriented software requires the definition of a multi-layered software architecture, the specification of subsystems that perform required functions and provide infrastructure support, a description of objects (classes) that form the building blocks of the system, and a description of the communication mechanisms that allow data to flow between layers, subsystems and objects. Object-oriented design accomplishes all of these things.

An object oriented system draws upon class definitions that are derived from the analysis model. Some of these definitions will have to be built from scratch, but many others can be reused if appropriate design patterns are recognized. OOD establishes a design blueprint that enables a software engineer to define the OO architecture in a manner that maximizes reuse, thereby improving development speed and ending product quality.

OOD is divided into two major activities: system design and object design. System design creates the product architecture, defining a series of layers that accomplish specific system functions and identifying the classes that reside at each layer. In addition, system design considers the specification of three components: the user interface, data management functions, and task management facilities. Object design focuses on the internal detail of individual classes, defining attributes, operations and message detail.

An OO design model encompasses software architecture, user interface description, data management components, task management facilities, and detailed descriptions of each class to be used in the system.

At each stage, the elements of the object oriented design model are reviewed for clarity, correctness, completeness, and consistency with customer requirements and with one another.

2.4 Conventional Vs. OO Approaches

Conventional approaches to software design apply a distinct notation and set of heuristics to map the analysis model into a design model. Each element of conventional analysis model maps into one or more layers of the design model. Like conventional software design, OOD applies data design when attributes are represented, interface design when a messaging model is developed, and component –level (procedural) design for the design of operations. It is important to note that the architecture of OO design has more to do with the collaborations among objects than flow of control between components of the system.

Within the object-oriented model, the subsystem design is derived by considering overall customer requirements (represented with use-cases) and also by the events and states that are externally observable (the object behavior model). Class and object design is mapped from the description of attributes, operations, and collaborations contained in the model. The object-relationship model enables message design, and responsibilities design is derived using the attributes, operations, and collaborations described in the model.

Fichman and Kemerer [FIC92] suggest ten design modeling components that may be used to compare various conventional and object-oriented design methods:

1. Representation of hierarchy of modules.
2. Specification of data definitions.
3. Specification of procedural logic.
4. Indication of end to end processing sequence.
5. Representation of object states and transitions.
6. Definition of classes and hierarchies.
7. Assignment of operations to classes.
8. Detailed definition of operations.
9. Specification of message connections.
10. Identification of exclusive services.

Because many conventional and object-oriented design approaches are available, it is difficult to develop a generalized comparison between the two methods. It can be stated, however, that modeling dimensions 5 through 10 are used more in the object oriented paradigm and are less used in the structured design or its derivatives.

The OOD landscape consists of a wide variety of object-oriented analysis and design methods that were proposed during the 1980s and 1990s. These methods established the foundation for modern OOD notation, design heuristics, and models. Some of these methods include the "Booch Method", "Rumbaugh Method", the "Jacobson Method", the "Coad and Yourdon Method".

Although the terminology and process steps for each of these OOD methods differ, the overall OOD processes are reasonably consistent. To perform object-oriented design, a software engineer should perform the following generic steps:

1. Describe each subsystem and allocate it to processors and tasks.
2. Choose a design strategy for implementing data management, support for interface, and management of task.
3. Design a suitable control mechanism for the system.

4. Make object design by creating a procedural representation for each operation and data structures for class attributes.
5. Perform message design using collaborations between objects and object relationships.
6. Create the message model.
7. Review the design model and iterate as required.

It is important to note that the design steps discussed above are iterative. This means that the steps can be executed incrementally, along with additional OOA activities, till a completed design is produced.

As discussed above, there were various methods that evolved in helping the system designers in achieving the above stated tasks. However, over a period of time, Booch, Rumbaugh and Jacobson combined the best features of their individual object-oriented analysis and design methods into a unified method. The result, called the ***Unified Modeling Language (UML)*** has become widely used in the industry.

3 UML: A Theoretical Perspective

3.1 UML-Overview

UML is a third-generation method for “specifying, visualizing, and documenting the artifacts of an object-oriented system under development”. UML represents the unification of the earlier Booch, Objectory, and OMT methods, and in addition it also incorporates ideas from a number of other methodologists. By unifying these three object-oriented methods, UML provides method that is common, stable, and expressive. (The material in this chapter is drawn, in part, from UML Distilled, Second Edition: A Brief Guide to the Standard Object Modeling Language; The Addison-Wesley Object Technology Series by Martin Fowler, Kendall Scott.)

UML was designed by intentionally making it quiet on process. However, UML does enable a process that is use-case driven, architecture-centric, and both incremental and iterative. In can be stated that in many ways, UML tries to codify the best practices that software people have encountered in successful object-oriented projects worldwide. Therefore the authors of UML were in many ways, not really the "inventors" of anything radically new. But, the value that UML brings is that its proponents have observed what works and what doesn't in the world of object-oriented software development, and then they packaged that up in the form of a modeling language that scales to systems of complexity.

3.2 UML: Modeling Concepts

The UML specifies a modeling language that incorporates the object-oriented community's consensus on core modeling concepts. The developers of the UML had the following objectives in mind during its development:

- Provide sufficient semantics and notation to address a wide variety of contemporary modeling issues in a direct and economical fashion.
- Provide sufficient semantics to address expected future modeling issues, specifically related to component technology, distributed computing, application frameworks, and executability.
- Provide extensibility mechanisms so individual projects can extend the meta-model for their application at low cost. It doesn't want users to have to adjust to the UML meta-model itself.
- Provide extensibility mechanisms so that future modeling approaches could grow on top of the UML
- Provide sufficient semantics to facilitate model interchange among a variety of tools.
- Provide sufficient semantics to specify the interface to repositories for the sharing and storage of model artifacts.

3.3 UML Meta-Model

The purpose of a model is to represent the semantics, physical details, and visualization of analysis, design, or implementation. The UML meta-model introduces concepts that are generally useful across many problems and many languages. This is always a balancing act and lines must be drawn, so there is always something that a method cannot model directly, no matter how complicated the method is. Agreeing on a common meta-model has had a great impact on flexibility and compatibility of further developments. The UML, in its current state, defines a notation and a meta-model. The notation it defines is the graphical elements displayed in the models. It is the syntax of the modeling language, for example, the class diagram notation defines how items and concepts such as class, association, and multiplicity are represented.

Formal methods usually conform to the idea of rigorous specification and design languages. However, since design is all about seeing the key issues in the development, formal methods often lead to getting bogged down in minor details. Further, formal methods are hard to understand and manipulate and often harder to deal with than the programming languages itself. However, most of the OO methods have very little rigor and their notation appeal to our intuition rather than to formal definition. On the whole this does not seem to have much harm, the methods may be informal but many people still find them useful that is what counts.

Having said that, it is also to be mentioned that there is a continuous effort on the part of the OO community to improve the rigor of the methods without sacrificing the usefulness. One way of achieving this is to define a meta-model i.e. a diagram that defines the notation. The diagram below is a small piece of UML meta-model that shows the relationship among associations and generalizations.

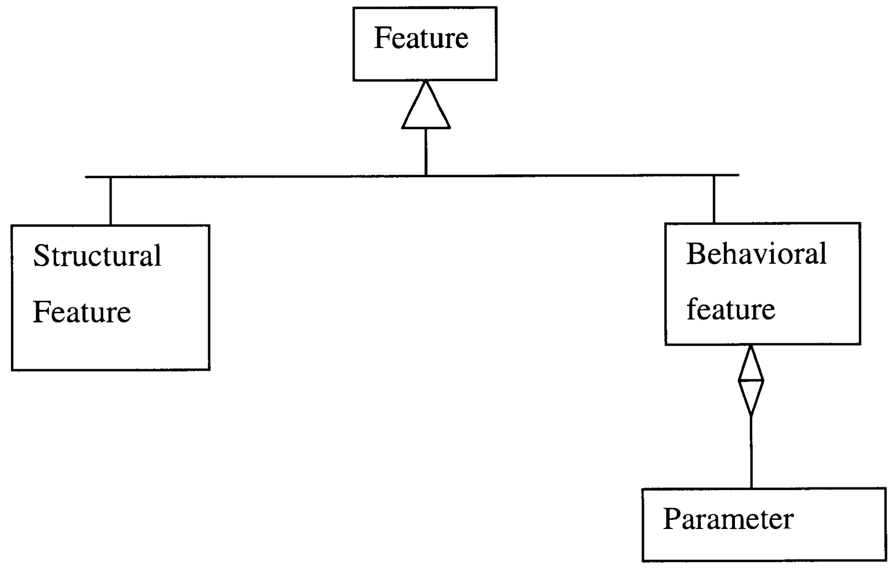


Figure 3-1 UML Meta Model

3.4 UML Meta-Model: Terminology

3.4.1 Use cases

Before defining a use case we need to define a scenario. A scenario is a sequence of steps that describes an interaction between a user and a system. So if we have a Web-based on-line store, we might have a "Buy a Product" scenario that would say this:

> "The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up email."

This scenario is one thing that can happen. However, the credit card authorization may fail. This would be a separate scenario.

A use case therefore is a set of scenarios tied together by a common user goal. A simple format for capturing a use case involves describing its primary scenario as a sequence of numbered steps and the alternatives as variations on the sequence.

There is a lot of variation as far as how one might describe the contents of a use-case; the UML does not specify any standard. One can even add additional standards. For example one can add a line for pre-conditions, which are things that must be true when the use case can start. However, it is also to be noted that the amount of detail one needs depends on the risk involved in the use-case: The more the risk the more detail one needs. (System size is the largest determinant: in a large system, no use case will have the detail that a small system will.)

3.4.2 Use case Diagram

A use case diagram is that diagram that captures the interactions between use cases and actors. It describes the functional requirements of the system and also how outside things like actors interact at the system boundary, and also what the system does in response.

The diagram below illustrates some features of Use Case Diagrams.

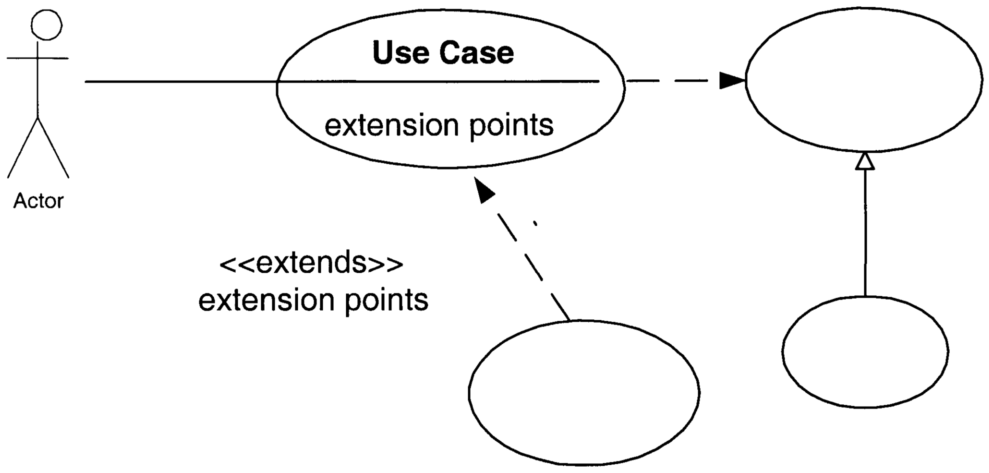


Figure 3-2 Use Case Diagram - Features

The diagram below shows a typical use-case diagram for a customer who has to log in to register with a bookshop.

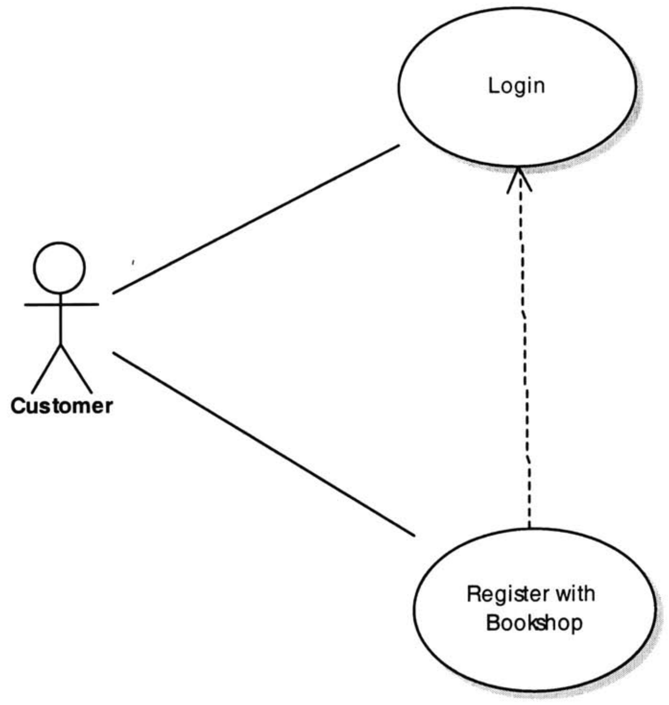


Figure 3-3 Use Case Diagram - Example

In addition to introducing use cases as primary elements in software development, Jacobson also introduced a diagram for visualizing use cases. The use case diagram is now an integral part of UML. The important elements of a use case diagram are discussed here:

3.4.2.1 Actors

An Actor is an entity that is the user of the system: either a human user, or machine or even another system. It is anything that interacts with the system from the outside or system boundary. Actors are typically associated with Use Cases. The actors may use the system either through a graphical user interface, or through a batch interface or through some other media. An actor's interaction with a use case is documented in a use case scenario and it details the functions a system must provide to satisfy the user requirements. The actor is represented as shown in the diagram below.

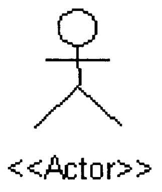


Figure 3-4 Actor In Use case

3.4.3 Use Case Relationships

The interaction between the use cases and the actors are shown as links. In addition to the links among actors and use cases, one can show several kinds of relationships between use cases.

3.4.3.1 Include

An element often includes the functioning of another. The include type of relationship in the use case models indicate that one use case (always) includes the behavior of another. It is used to avoid having the same subset of behavior in many use cases.

3.4.4 Generalization

One can use ***use-case generalization*** kind of relationship when one has one use case that is similar to another use case but does a bit mot more. In effect this is another way of capturing alternative scenarios.

3.4.5 Extend

A third relationship is called ***extend***. An element extends the behavior of another. As used in use case models it indicates that one use case (optionally) extends the behavior of another. Often, they are used to express alternate flows. Essentially it is similar to generalization but with more rules to it. With this construct the extending use case may add behavior to the base use case, but the base use case must declare certain "extension points", and the extending use case may add additional behavior only at those extension points. A use case may have many extension points, and an extending use case may extend one or more of this extension points. One can indicate which ones on the lines between the use cases on the diagram.

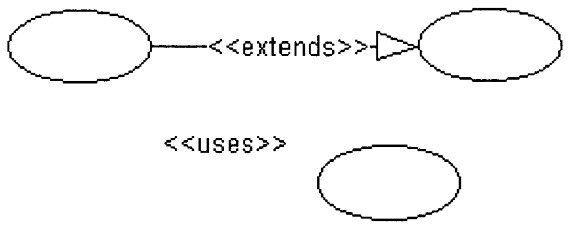


Figure 3-5 Use Case: Extends Diagram

Some of the rules that could be applied to the use of these relationships are the following:

- *Include* should be used when one is repeating oneself in two or more separate use cases and to avoid repetition.
- *Generalization may be used* when one is describing a variation on normal behavior and a more controlled form is needed, declaring ones extension points in ones' base case.

3.5 Class Diagrams

The class diagram technique has become truly central within object-oriented methods. The class diagram captures the logical structure of the system i.e. the classes and associated things that make up the model. It is a static model; i.e. it describes what exists and what attributes and behavior it has, rather than how something is done. Most of the object-oriented methods have included some variation of this technique.

There are two principal kinds of static relationships:

- **Associations** (for example a customer may rent a number of videos)
- **Subtypes** (a nurse is a kind of person)

Class diagrams also show the attributes and operations of a class and constraints that apply to the way objects are connected. A typical class diagram is shown below. This figure shows class diagrams from the theoretical standpoint.

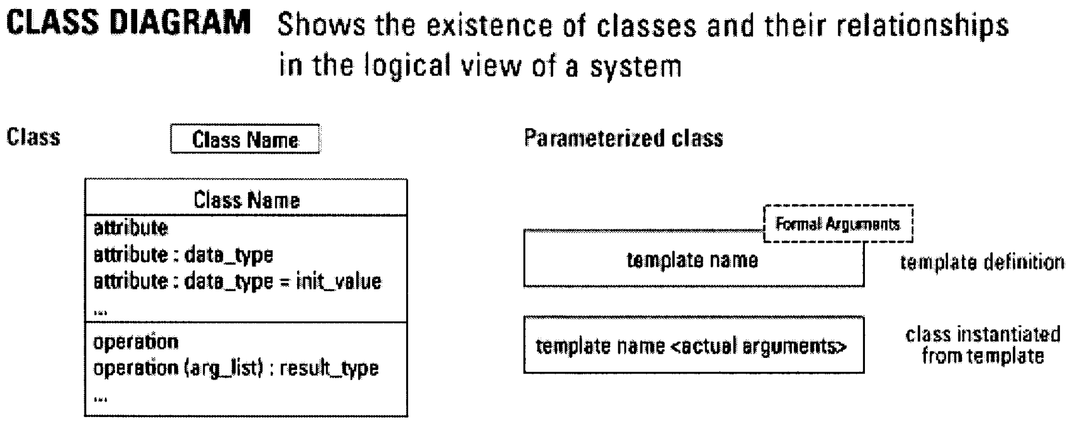

CLASS DIAGRAM Shows the existence of classes and their relationships in the logical view of a system
Class
Class Name
Class Name
attribute
attribute : data_type
attribute : data_type = init_value
...
operation
operation (arg_list) : result_type
...
Parameterized class
Formal Arguments
template name
template definition
template name <actual arguments>
class instantiated from template

Figure 3-6 Class Diagram Concepts (Source: UML™ Quick Reference for Rational Rose)

Below we show the various classes and their interaction for a bookstore. It is self-explanatory.

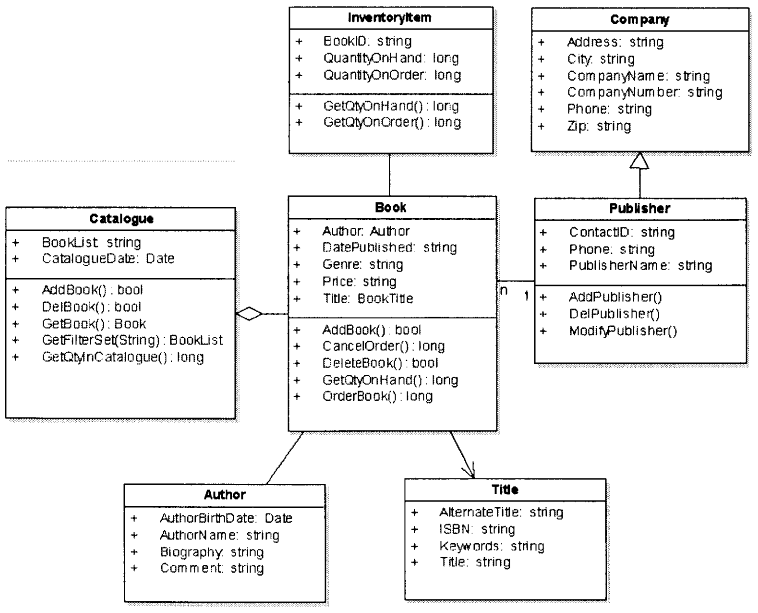


Figure 3-7 Class Diagram Interactions - Example

3.5.1 Association

Associations represent relationships between the instances of classes. For example a person works for the company; a company has a number of offices. Each association has two **association ends and** each of the ends is attached to one of the classes in the association. The ends can be explicitly named with a label, which is called a **role name**. An association end also has **multiplicity**, which indicates that how many objects may participate in any given relationship. In general, the multiplicity indicates lower and upper bounds for the participating objects. The * represents the range *0..Infinity.* The most common multiplicities in practice are 1, *, and 0.. 1.

The arrows on the association lines show **navigability**. If navigability exists in only one direction, we call the association a unidirectional association. Similarly, a bi-directional association contains navigability in both directions. As per UML specification one should treat associations without arrows to mean that either the navigability is unknown or the association is bi-directional. Bi-directional associations also include extra constraints, which is that two navigations are inverses of each other.

3.5.2 Attributes

An attribute is a named slot within a class that describes a range of values that instances of the class may hold. Depending on the detail in the diagram, the notation for an attribute can show the attribute's name, type and default value. The UML syntax is *visibility name: type=default Value,* where *visibility* is the same as for operations.

3.5.3 Operations

An operation is that use which can be requested from an object to affect some behavior. The operations have signature, which describes the actual parameters that are possible (including possible return values). A "method" is the implementation of an operation. It stipulates the procedure that affects the results of an operation. Thus operations are the processes that a class knows to carry out.

Full UML syntax for operations is

Visibility Name Parameter-list): Return-type-expressions {property-string}

Where

- *Visibility is* + (public), # (protected), or – (private)
- *Name* is a string
- *Parameter-list* contains comma separated parameters whose syntax is similar to that for attributes: *direction name: type=default value.* The element *direction* is used for input (*in*), output (*out*), or both (*inout*) with the default value being *in.*
- *Return-type-expression* is a comma-separated list of return types. Most people use only one return type, but multiple types are allowed.
- *Property-string* indicates property values that apply to a given operation.

The diagram below shows the attributes and operations of the Customer class for a typical business application. It shows all the elements discussed above.

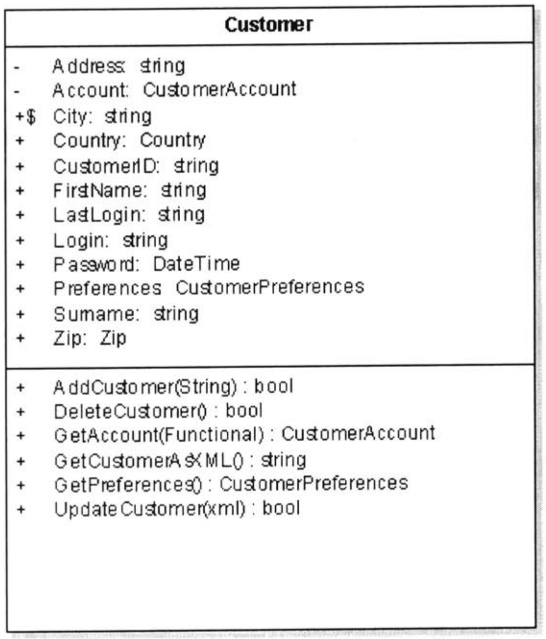

Customer
- Address: string
- Account: CustomerAccount
+$ City: string
+ Country: Country
+ CustomerID: string
+ FirstName: string
+ LastLogin: string
+ Login: string
+ Password: DateTime
+ Preferences CustomerPreferences
+ Surname: string
+ Zip: Zip
+ AddCustomer(String) : bool
+ DeleteCustomer() : bool
+ GetAccount(Functional) : CustomerAccount
+ GetCustomerAsXML() : string
+ GetPreferences() : CustomerPreferences
+ UpdateCustomer(xml) : bool

Figure 3-8 Class Example

3.5.4 Constraints Rules

The basic constructs of association, attribute, and generalization do specify important constraints, but they indicate every constraint. These constraints still need to be captured; and is generally done in the class diagram. The UML allows one to use anything to describe constraints with the only rule being that one should put them in braces ({ }).

3.5.5 When to Use Class Diagrams

Class diagrams are the backbone of nearly all OO methods. But the richness of the class diagrams does pose a problem of being overwhelming to use. Some tips on their effective use are provided below.

- One should start with simple elements like classes associations, attributes, generalization, and constraints and not try to use all the notations available. Similarly, other notations should be introduced only when needed.
- One should not draw models for everything but should instead concentrate on the key areas as it is better to have a few diagrams that one can use and keep up to date than to have many forgotten, obsolete models.

The diagram below shows typical representations of Association classes, Role names and derived associations, Aggregation, navigability and multiplicity and also constraints from the theoretical standpoint.

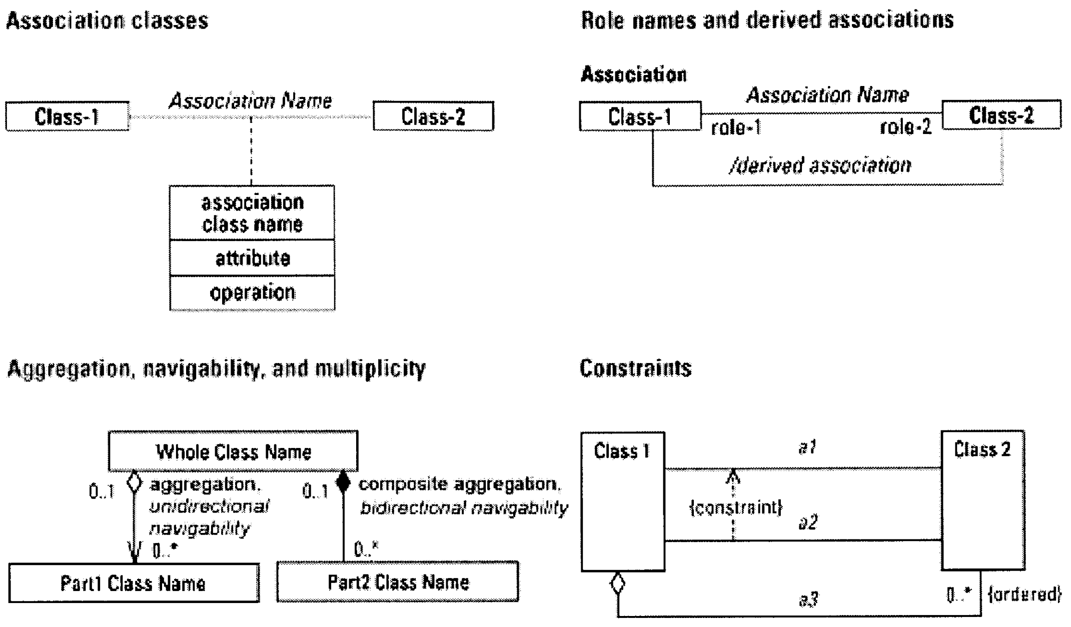


Figure 3-9 Class Diagrams – Other Concepts (Source: UML™ Quick Reference for Rational Rose)

3.6 Interaction Diagrams

Interaction diagrams are models that describe how groups of objects collaborate in some behavior. Typically, an interaction diagram captures the behavior of a single use case. These diagrams show a number of example objects and the messages that are passed between these objects within the use case.

There are two kinds of interaction diagrams: sequence diagrams and collaboration diagrams.

3.6.1 Sequence Diagrams

A sequence diagram is defined as an ordered depiction of behavior as a progression of sequential steps over time. It is used to depict workflow, message passing and how elements in general co-operate over time to achieve some result. The diagram below shows the theoretical representation of the interaction diagrams.

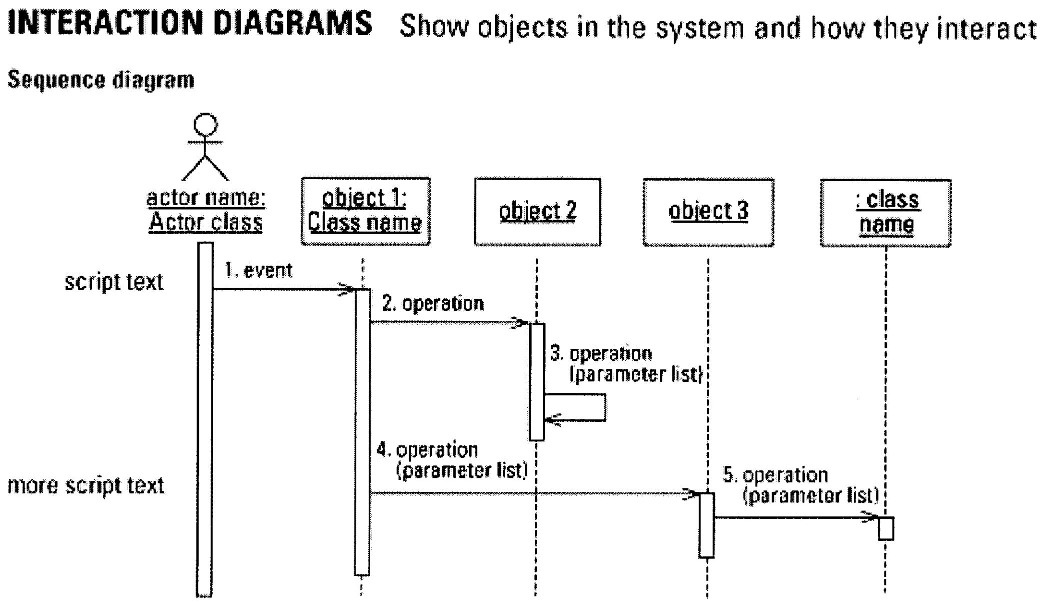

INTERACTION DIAGRAMS Show objects in the system and how they interact
Sequence diagram
actor name: Actor class
object 1: Class name
object 2
object 3
: class name
script text
1. event
2. operation
3. operation (parameter list)
4. operation (parameter list)
more script text
5. operation (parameter list)

Figure 3-10 Sequence Diagram Concepts (Source: UML™ Quick Reference for Rational Rose)

The diagram below shows the sequence diagram for the various activities involved in the validation of the customers in any operation as an example.

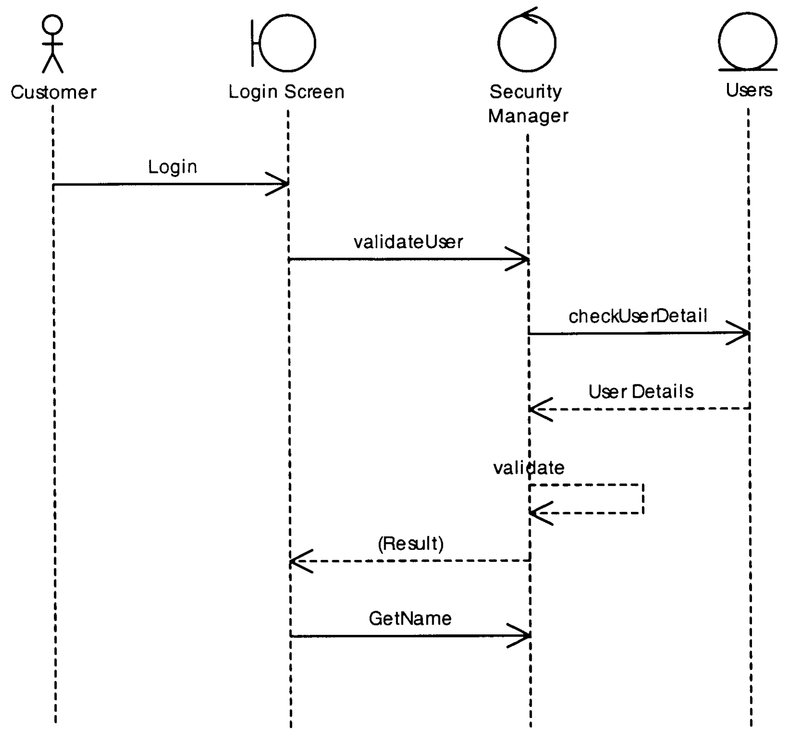


Figure 3-11 Sequence Diagram Example

Within a **sequence diagram** an object is shown as a box at the top of a vertical line. This vertical line is called the objects **lifeline.** The lifeline represents the objects life during the interaction. An arrow between the lifelines of two objects represents each message. The order in which these messages occur is shown top to bottom of the page. Each message is labeled at minimum with a message name; one can also include the arguments and some control information. One can also show a self-call, a message that an object sends to itself, by sending the message arrow back to the same lifeline.

3.7 Component Diagram & Deployment Diagram

A component diagram illustrates the pieces of software and embedded controllers etc. that will make up the system. It is a level up from the class level. Usually one or more classes implement a component (or objects at runtime).

A deployment diagram shows how and where the system will be deployed. It displays physical machines and processors, and the components that run on them.

The diagram below show a typical component diagram and a typical deployment diagram.

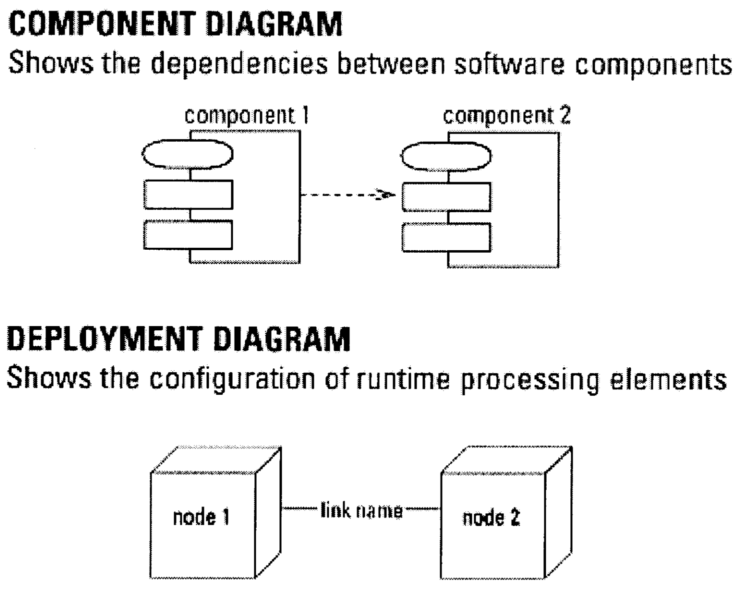

COMPONENT DIAGRAM
Shows the dependencies between software components
component 1
component 2
DEPLOYMENT DIAGRAM
Shows the configuration of runtime processing elements
node 1
link name
node 2

Figure 3-12 Component and Deployment Diagram (Source: UML™ Quick Reference for Rational Rose)

The diagram below shows a typical component diagram for a web application. It shows the various components that form the application and their inter-relationships.

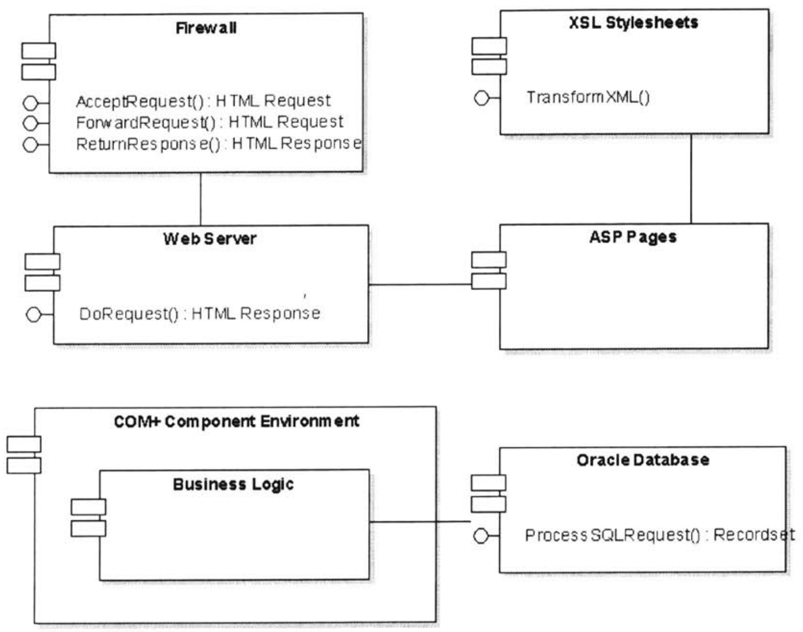

Firewall
AcceptRequest() : HTML Request
ForwardRequest() : HTML Request
ReturnResponse() : HTML Response
XSL Stylesheets
TransformXML()
Web Server
DoRequest() : HTML Response
ASP Pages
COM+ Component Environment
Business Logic
Oracle Database
ProcessSQLRequest() : Recordset

Figure 3-13 Component Diagram Example

The diagram below shows the deployment diagram for a simple application with four servers, two for the application, one for the database and one for the warehouse.

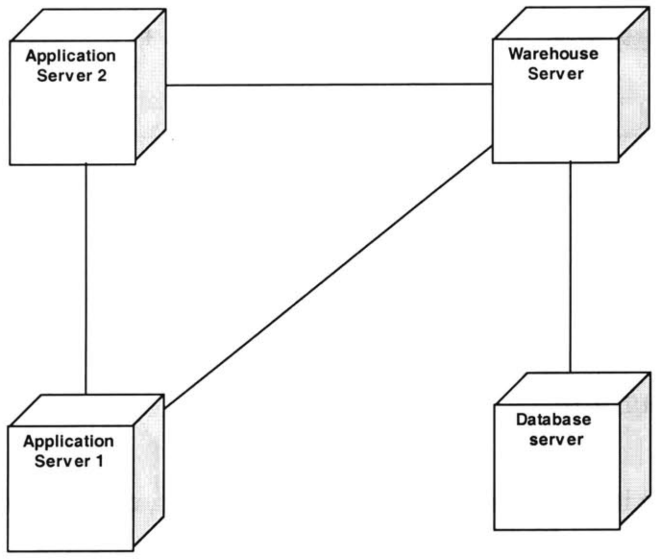


Figure 3-14 Deployment Diagram Example

3.8 Collaboration Diagram

A collaboration diagram is also a type of Interaction diagram. Within the collaboration diagram, the example objects are shown as icons. As on a sequence diagram, arrows indicate the messages sent within the given use case. However, in collaboration diagrams, the numbering on the messages indicates their position in the sequence of activities. The diagram below shows the sequence in the collaboration diagram for a printing request on a computer network.

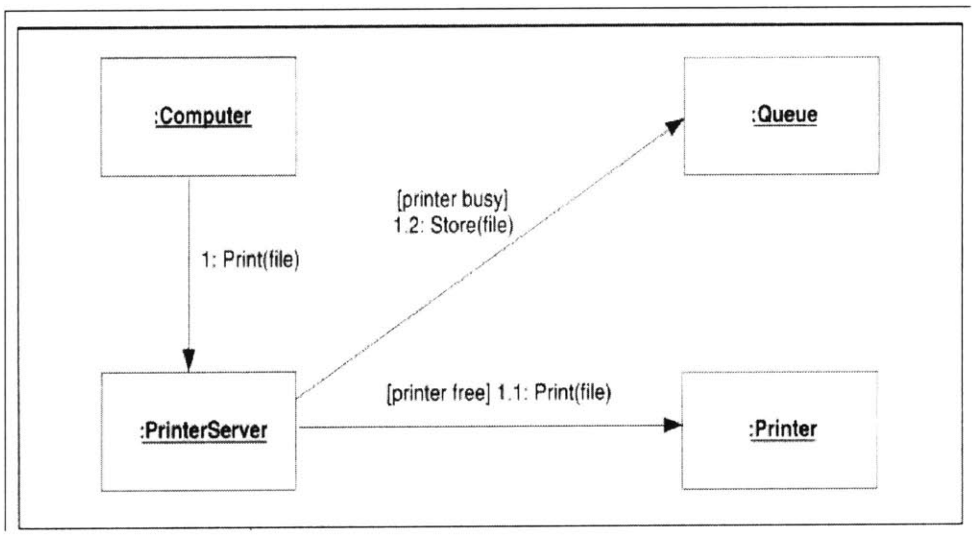


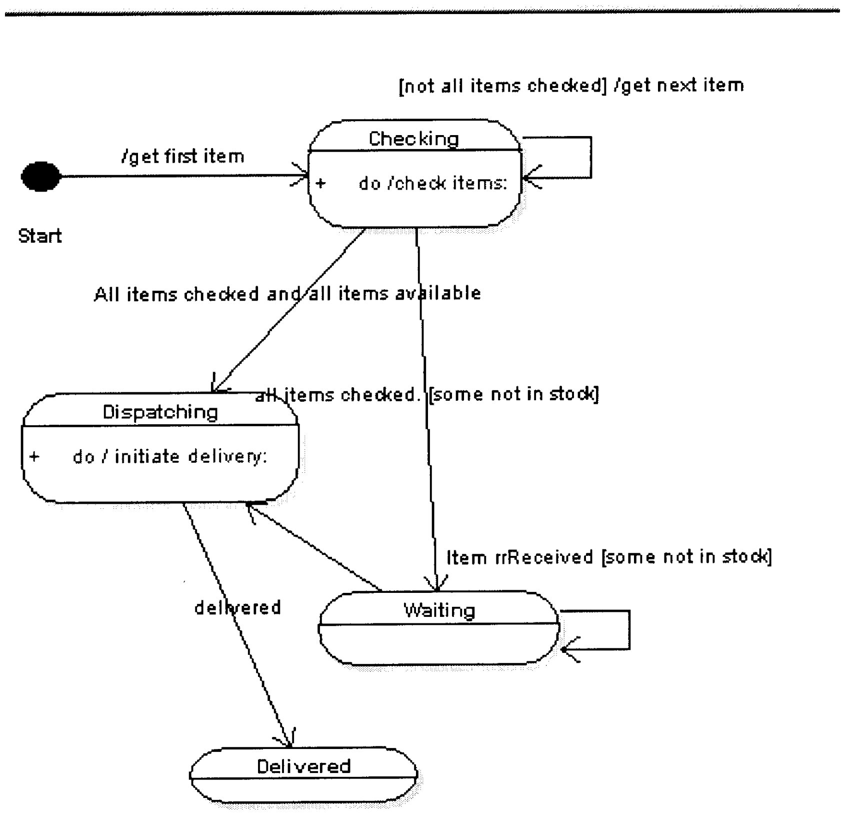
Figure 3-15 Collaboration Diagram

3.9 Activity Diagram

An activity diagram is used to show activities, workflows, conditions to processing and how work is done in general in the organization. The diagram below shows the activity diagram for the order processing related activities. This diagram allows one to choose the order in which to do things. In other words it just states the essential sequencing rules that should be followed. The strength of the activity diagrams lies in the fact that they support

3.10 State Diagrams

A state diagram is used to show how an element (usually a class) changes state over time and also the transitions that are allowable along with the conditions for the transitions. They are familiar techniques to describe the behavior of the system. They describe all the possible states that a particular object can get into and how the object's state changes as a result of the event that reaches the object.



[not all items checked] /get next item
Checking
+ do /check items:
/get first item
Start
All items checked and all items available
all items checked. [some not in stock]
Dispatching
+ do / initiate delivery:
Item rrReceived [some not in stock]
delivered
Waiting
Delivered

Figure 3-17 State Diagram Example (Based on UML Distilled: Martin Fowler)

3.10 State Diagrams

A state diagram is used to show how an element (usually a class) changes state over time and also the transitions that are allowable along with the conditions for the transitions. They are familiar techniques to describe the behavior of the system. They describe all the possible states that a particular object can get into and how the object's state changes as a result of the event that reaches the object.

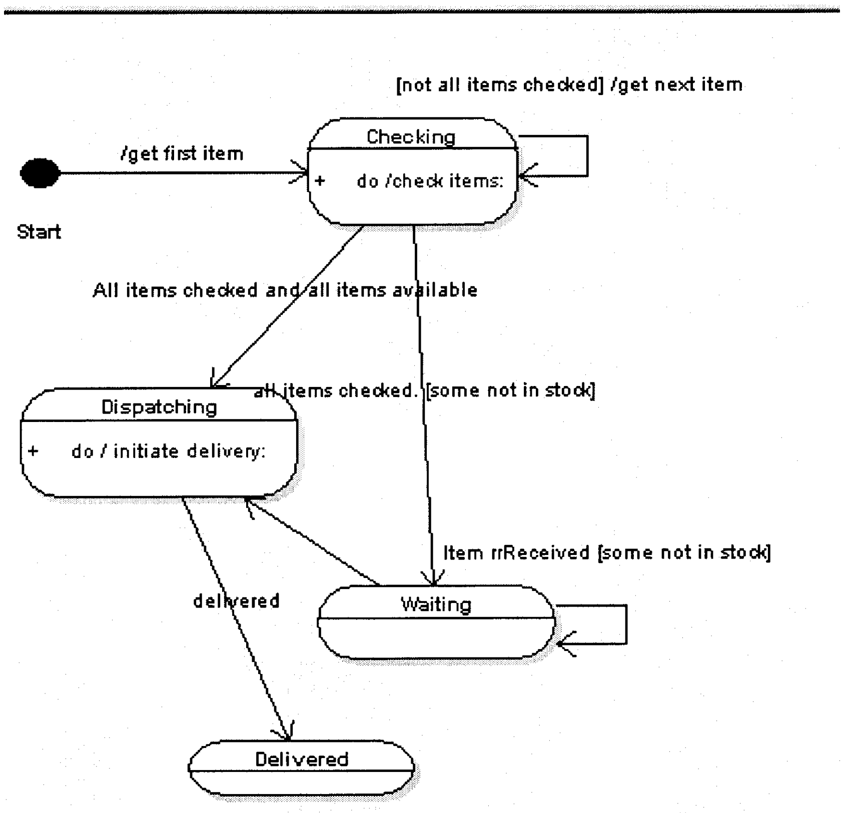

[not all items checked] /get next item
Checking
+ do /check items:
/get first item
Start
All items checked and all items available
Dispatching
+ do / initiate delivery:
all items checked. [some not in stock]
Item rrReceived [some not in stock]
delivered
Waiting
Delivered

Figure 3-17 State Diagram Example (Based on UML Distilled: Martin Fowler)

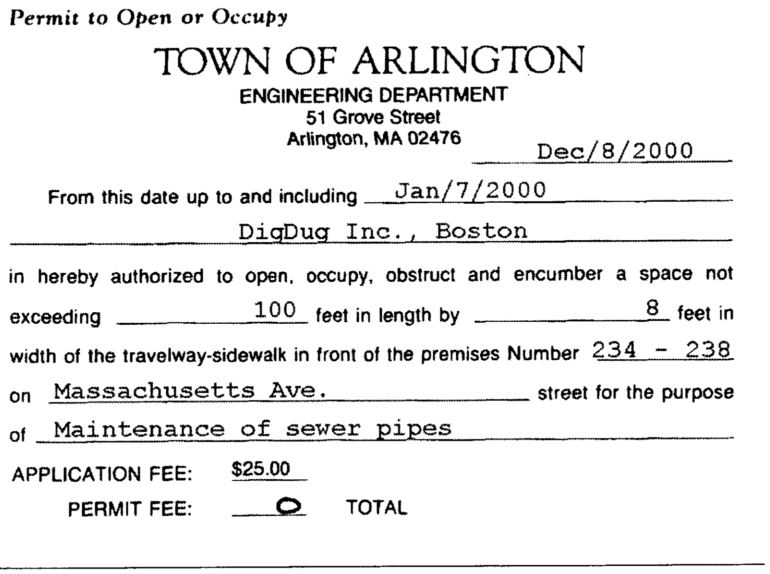
The diagram above shows the state diagram for an order in the order processing activity. The diagram indicates the various states of an order. The state diagrams are good at describing the behavior of an object across several use cases.

4 Pavement Permit System

4.1 Problem Description

Road construction work in the town of Arlington needs to be approved by the public works department. The major part of road construction work is the opening of streets, as required by utility companies (telco, gas, cable TV) and by individual contractors.

At present, utility companies and contractors have to apply for a permit for street opening in written form i.e. on a standardized form. They apply for the permit by filling out the standard form and submitting it to the department. The Department then processes the form. The town officials verify the application regarding location, proposed time, and occupied space, validity of the contractor registration, his permissions and so on. They also check against any restrictions that apply on the proposed work. After collection of the application fee, the permit is finally issued. The current paper form is shown in part below.


Permit to Open or Occupy

TOWN OF ARLINGTON

ENGINEERING DEPARTMENT
51 Grove Street
Arlington, MA 02476

Dec/8/2000

From this date up to and including Jan/7/2000

DigDug Inc., Boston

in hereby authorized to open, occupy, obstruct and encumber a space not exceeding 100 feet in length by 8 feet in width of the travelway-sidewalk in front of the premises Number 234 - 238 on Massachusetts Ave. street for the purpose of Maintenance of sewer pipes

APPLICATION FEE: $25.00
PERMIT FEE: 0 TOTAL

Figure 4-1 Paper Based Permit

This system has its own disadvantages. It is time intensive for both the permit applicants and the department personnel. The applicants have to fill out the application and wait for the department officials to process it. The department officials also have to input a substantial amount of time to review the application in terms of the conditions that apply relating to the parameters of the application. Since, on the average 500 applications are received by the department every year, they have to station at least one person in the office to take care of the applications. Moreover, the issues of reports and permit search are also very cumbersome for the paper based application system.

Considering this, it appeared logical to apply modern technology and provided for an automated system that will be accessible to all concerned and would process the applications on its own. Therefore, a web based application using the Internet technology seemed to be the best option to solve the problem.

4.2 Requirements of the Proposed System

Based on the discussions that our team had with the officials at Arlington PWD, Engineering Department, the following system requirements were identified. We are categorizing these identified requirements based on the various users of the proposed system.

4.2.1 Contractors and Utility Companies (External Users) Perspective

The requirements from the contactors and utilities that form the "external users" of the system are as under:

- The system should be easy to use. This is essential, as the department does not perceive most of the external users of the system and their agents as very proficient in the use of the computers.
- The system should be accessible from the web for ubiquitous access.
- The system should provide for a form, which is similar to the currently used form.

- The system should allow the external users the fill out the form and provide them with instant message on the processing results and also the facility to print the permit if granted.
- The system should provide the users with the reasons for rejection of their application, when applicable.
- The system should provide the users with the requisite fee for processing of their application.
- The system should allow the users a search facility, where they should be able for their own permits issued in the past.
- The system should provide the necessary help / Frequently asked questions section, to help the external users.

4.2.2 The internal user's perspective

- The system should allow the internal users to specify the valid companies and relevant information on them as also updating them removing them as needed.
- The system should allow the internal user to specify the restricted streets / sections and the details thereof.
- The system should allow the internal user to specify other restrictions related to holidays, valid time of application, etc.
- The system should allow the internal user to be intimated by e-mail when an application is granted.
- The system should allow the internal user to generate billing information from the system, which would allow him to prepare the billing summary meeting the current practice.
- The system should provide help section where information about system's use is documented.
- The system should allow the internal user to generate reports regarding the permits issued by the system.

- The system should allow the internal user to apply for a permit on a contractor or companies' behalf and override system decision when rejected, to meet emergency conditions.
- The system, before issuing a permit should make an exhaustive check on the restrictive conditions that apply during the proposed time of work, including for companies' validity for registration, permission on types of work, restrictions on streets / sections, other restrictions related to holidays and so on.

4.2.3 General Perspective

- The system should allow the internal users to set some miscellaneous setting applicable to the system as a whole, like, the timings for the week ends, whether e-mail notification be done, the e-mail id where notification be done, time for notification / demarcation before work can be undertaken by the contactor, fee structure etc.
- The system should be integrate with the Pavement Management system and provide that system with information related to manholes etc. that are added to the streets, so that the Pavement Management system, can take that into account in owns own application. The data on streets should be common for both the applications.
- The system should provide for separate entry points for the external and internal users.
- The system should also provide for login protection to the application.
- The system should provide for feedback by the users, both internal and external, so as to improve upon the application and also to help in its maintenance.
- The application configuration should allow for using less expensive components and as far as possible, freeware / shareware software should be used. This necessary considering the constraints of the budgetary provisions of the department.

5 Pavement Permit System – Use Cases

5.1 Overview

From the requirements stated for the pavement permit system in chapter four, several use cases can be derived. These use cases would essentially contain all the expected user interaction with the system. The use cases from the internal users' (users from within the department) perspective would relate to the following elements.

- Companies
- Permits
- Street Restrictions
- Holiday Schedule
- Miscellaneous settings and options
- Billing data transfer
- Reports
- Frequently Asked Questions (FAQs)
- Help and Instructions

Similarly, from the external users' (contractors and utility companies) perspective, the use cases would relate to mainly the Permits menu. Further, both the groups would also have login related use cases too.

In view of the fact that a complete listing of all the use cases possible would be beyond the purpose of the thesis, we are enunciating some of the more important use cases that we derived.

Each of the use-cases describes the sequence of activities that would occur for that use-case. We also provide the representative use-case diagrams for some of the use-cases as examples. The use-case diagrams for the other use-cases are similar. However, in chapter six also we provide some use-case diagrams for this application.

5.2 Use Cases

Below we provide the use case diagram for the

5.2.1 Use Case for Accessing Permits menu – Internal User

Below we provide a simple use case representation of the use case diagram:

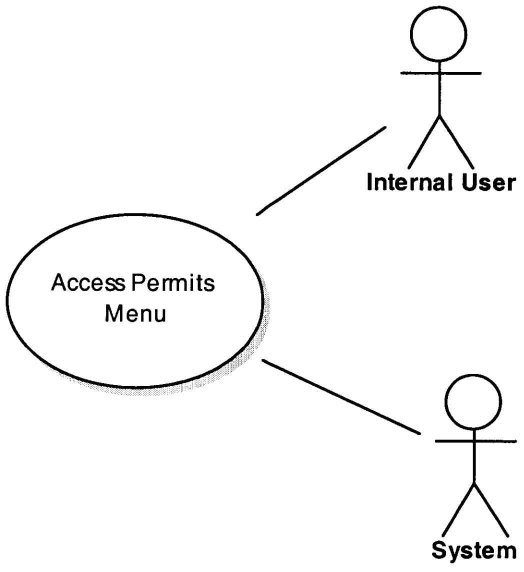


Figure 5-1 Use-case Diagram: Access Permits Menu

As seen in the diagram above, there is a basic use case with two actors, namely the internal user and the system itself. Both the actors interact with the basic use-case, "Access the Permits Menu" to complete the stated use-case. The complete sequence of the activities with the main use-case is enumerated below.

1. The internal user selects "permits menu" from the internal user home page.
2. The system displays the permits main page. The options provided on the main page include:

   a. "View all permits issued within the last 30 days",
   b. "Search for Permits",
   c. "Apply for a new permit",
   d. "Edit an existing permit" and
   e. "Delete an existing permit"
3. Depending on the user selection, the system takes the next action.

5.2.2 Use Case for making a new application- Internal User

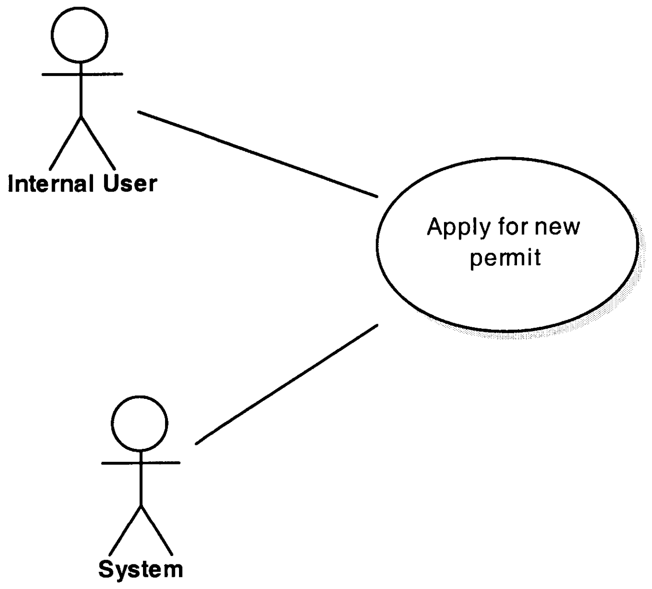


Figure 5-2 Apply For New Permit Use-Case Diagram

As seen in the diagram above too, there is a basic use case with two actors, namely the internal user and the system itself. Both the actors interact with the basic use-case, "Apply for new permit" to complete the stated use-case. The complete sequence of the activities with the main use-case is also enumerated below.

1. The internal user chooses the option to fill out a new application for permit from the Internal User main menu for Permits;

2. The system displays the form for the internal user to fill out for making the permit application;
3. The internal user fills out the following information;
    - Company name
    - Type of work the company does
    - The company's "DigSafe" number
    - Proposed first day of work
    - Proposed last day of work
    - Type of the proposed work
    - Street name where work is proposed
    - Premises, from and to where work is proposed on the above mentioned street
    - Length and width of the working area
    - Description of the purpose of the work
    - Upload any necessary drawing file
    - Information on number of openings that are added to or removed from the street, if any along with the type i.e. telephone manholes, Water Manholes, Other Openings, Water Gates, Electric Manholes, Sewer Manholes, Catch Basins, Gas Gates.
    - User is also shown important standard information on the application.
4. The internal user submits the application to the system.
5. The system displays the information submitted by the internal user for confirmation.
6. On confirmation by the internal user in 5. above, the system makes following checks on the application
    - Whether the proposed company is allowed to do the type of work that is proposed
    - Whether a "DigSafe number" is provided
    - Whether first day of work is at least 48 hours (from the database) later than the current time.

- Whether the last day of work is at most 30 days later than the first day of work, and that it is later than or equal to the first day of work.
- Whether the proposed street is on the currently restricted list
- Whether the proposed range of premises are overlapping a restricted range of premises on the street
- Whether the proposed date range for work overlaps with the restricted date ranges on the street
- Whether the proposed work falls on a holiday
- Whether the application is being made on a holiday
- System obtains the information on fee from the database, computes the application fee, and displays it.

7. If none of the checks fail, the system displays a "permit granted" message along with the permit itself and with a request to the internal user to get a printable version to print. The system also updates the database on the permit by adding this issued permit.
8. The internal user prints the permit.
9. If one or many check fails then the system displays a message on denial of permit along with the reason for denial. However, it also gives the internal user to override the system decision and have the permit issued, in which case the steps in 8 and 9 above are repeated.
10. The internal user may also reapply with changed parameters.

5.2.3 Use-Case for searching permits by Internal User

1. Internal user selects "Search for Specific Applications/Permits" from the permits menu.
2. System displays "search" page.
3. The system presents the internal user to search the permits in various ways. The user could select / provide the following parameters:
    - ➢ Contractor/Utility Company name,
    - ➢ Permit Number,

- Permit Issue Dates; with 'From date' and 'To date'
- Street Name.

4. System searches the database based on the combination of information on the search provided by the user on the search from and displays the results on a separate page. This display would be including the important parameters for the permit.
5. For the entire details the user could click on the hyperlink on the permit number provided in the display and then the system would then display the details about the particular permit.

5.2.4 Use Case to view permits issued within last 30 days by Internal User

1. The internal user selects the option on "viewing permits issued within the last 30 days".
2. System searches the database for all permits issued within the last 30 days and displays the results on a separate page. This display would include the important parameters of the permit.
3. For the entire details the user could click on the hyperlink on the permit number provided in the display and then the system would then display the details about the particular permit.

5.2.5 Use Case for editing an existing permit

1. The internal user provides a permit number to edit on the permits main page.
2. The system presents the requested permit in a separate editable form.
3. The user makes the necessary changes to the permit.
4. The system updates the database and gives a message on "success / error" to the user. The user can then go back to the permits main menu for further action.
5. In case of error, the database is not updated and the user can go back to the permits main menu.

5.2.6 Use Case for deleting an existing permit

1. The internal user provides a permit number to edit on the permits main page.
2. The system presents the requested permit in a separate page with details.
3. The internal user confirms deletion of the permit.
4. The system updates the database by deleting the said permit and gives a message on "success / error" to the user. The user can then go back to the permits main menu for further action.
5. In case of error, the database is not updated and the user can go back to the permits main menu.

5.2.7 Use-Case on street restrictions administration

1. Internal user selects option to administer street restrictions from the Internal users main page;
2. The system displays the restrictions main menu;
3. The restrictions main menu includes options on adding a new restrictions, editing a restriction, deleting a restriction, listing all current restrictions, listing all restrictions ever;
4. User chooses an option from above and system displays the relevant page for further action.

5.2.8 Use-Case on street restriction addition:

1. Internal user selects "Add a New Street Restriction" option from street restrictions main menu;
2. System displays street restrictions addition form;
3. Users fills out data on the form
    - ➢ Street name,
    - ➢ Date of start for the restriction,
    - ➢ Date when restriction ends,
    - ➢ Reason for the restriction,
    - ➢ Status of restriction on it being active or inactive,

- ➢ The type of restriction i.e. the whole street or a section only)

4. Based on the user selection of the type of restriction the system would take further action;
5. If the user selects the "whole street" option, the system would add the entered data on the restrictions table in the database;
6. If the user selects the "section only" option, the system would display an additional page with a list of valid premises within the selected street and prompt the user to enter the start and end premises for the section. Then when the user provides it, the system would add the user provided data to the database.
7. The system then displays the confirmation ("success/error") page.
8. The confirmation page would have option to go back to the main restrictions menu.
9. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.9 Use-Case on street restriction editing

1. On the restrictions main menu, internal user is given a list of all restrictions from which he chooses the one to edit.
2. System displays the data on the selected restriction in an editable form.
3. Users makes the changes to the data on the restriction
4. The system updates the database;
5. The system then displays the confirmation ("success/error") page
6. The confirmation page would have option to go back to the main restrictions menu.
7. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.10 Use-Case on street restriction deleting:

1. On the restrictions main menu, internal user is given a list of all restrictions from which he chooses the one to delete.

2. System displays the data on the selected restriction for user to confirm deletion.
3. Users confirms to delete
4. The system deletes the restriction from the database;
5. The system then displays the confirmation ("success/error") page
6. The confirmation page would have option to go back to the main restrictions menu.
7. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.11 Use case on listing all current street restrictions

1. The internal user selects option to list all current restrictions from the main menu;
2. The system lists all "active" status restrictions to the user from the database.

5.2.12 Use case on listing all street restrictions in the entire history

1. The user selects option to list all restrictions in the history from the main menu;
2. The system lists all "active" status restrictions to the user from the database.

5.2.13 Use case on Companies Administration

1. Internal user selects option to administer information on companies from the Internal users main page;
2. The system displays the companies main menu;
3. The Companies main menu includes options on adding a new contractor/utility company, editing a contractor/utility company, deleting a contractor/utility company, listing all contractor/utility company,
4. User chooses an option from above and system displays the relevant page for further action.

5.2.14 Use-Case on company/utility company addition:

1. User selects "Add a New company/utility company" option from Companies Main menu;
2. System displays company/utility company addition form;
3. Users fills out data on the form
   - Company Name,
   - Company Type,
   - License Number,
   - Address information (Street, City, State, Zip, Phone, Fax, e-mail),
   - Types of Work the contractor does (Gas or Electric or Telco or Cable TV or Drain/Water/Sewer or Driveway or Other),
   - Types of Street Work permitted (i.e. Street Opening Only or Sewer Only or Water Only or Sewer And Water or Other),
   - The Status of the contractor for the Department (Active or Inactive),
   - Contact Information for three contact persons as optional (Name, Phone, Pager, Email
4. The system then adds the user provided data to the database.
5. The system then displays the confirmation ("success/error") page.
6. The confirmation page would have option to go back to the companies' main menu.
7. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.15 Use-Case on contractor/utility company editing

1. On the Companies main menu, user is given a list of all company/utility company from which he chooses the one to edit.
2. System displays the data on the selected contractor/utility company in an editable form.
3. Users makes the changes to the data on the contractor/utility company
4. The system updates the database;

5. The system then displays the confirmation (“success/error”) page
6. The confirmation page would have option to go back to the companies’ main menu.
7. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.16 Use-Case on company/utility company deleting:

1. On the companies’ main menu, internal user is given a list of all contractor/utility company from which he chooses the one to delete.
2. System displays the data on the selected contractor/utility company for user to confirm deletion.
3. Users confirms to delete
4. The system deletes the company/utility company from the database;
5. The system then displays the confirmation (“success/error”) page
8. The confirmation page would have option to go back to the companies’ main menu.
9. In case of error, the database is not updated and the user can go back to the Restrictions main menu.

5.2.17 Use case on listing all contractor/utility company:

1. The user selects option to list all contractor/utility company from the main menu;
2. The system lists all company/utility company to the user from the database, in a concise format, with links so that the user can click on it to get the full details on a separate page.

6 UML Application To Pavement Permit System

The UML design techniques that were discussed in chapter three can be successfully applied to any project. Below, we apply these techniques to a representative part of the Pavement Permit System that we outlined in chapter four. We also provide the necessary explanations for the diagrams.

6.1 Use Case Diagram

The use-case diagram technique can be applied very effectively to the Pavement permit system. As a representative application, below we draw the use-case diagram for the use case on permit applications. The diagram captures the basic elements in the permit application use case.

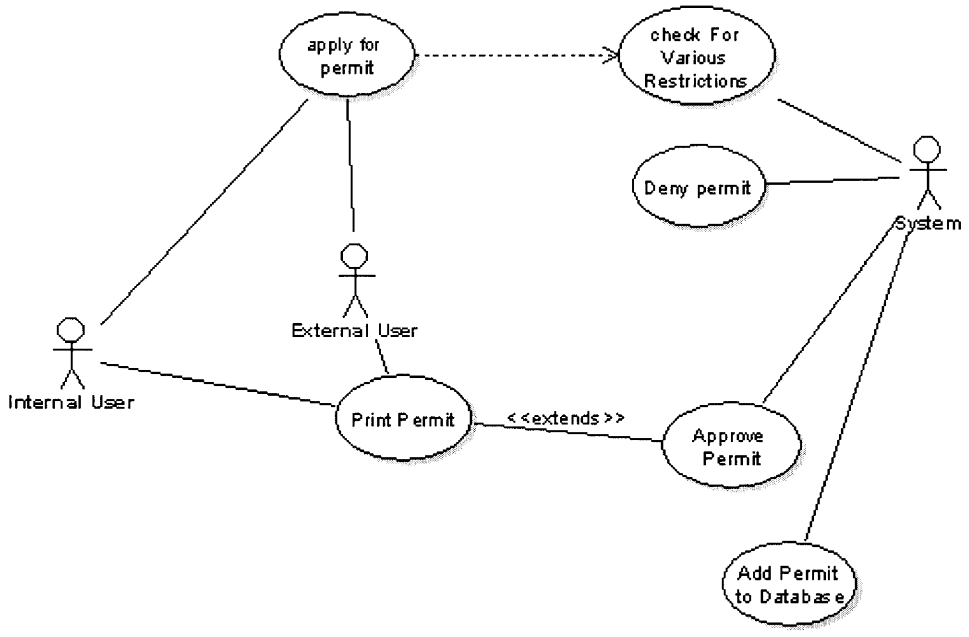


Figure 6-1 Use Case Diagram: Permit Application

There are three actors namely, the Internal User, the System and the External User. There are six basic use-cases namely:

- Apply for the permit
- Check for various restrictions
- Approve permit
- Deny permit
- Add permit to database
- Print permit

The various relationships between the actors and the use-cases are shown in the diagram. This diagram is basically a pictorial representation of the use-case sequence described above in the "Use-Case" section for the permit application process.

Moreover, this is a representative diagram for this particular use-case. The same diagrammatic technique can be applied to all the use cases described above. It is a very powerful tool and captures the essence of the use-cases in a very simple but effective manner.

6.2 Class Diagrams

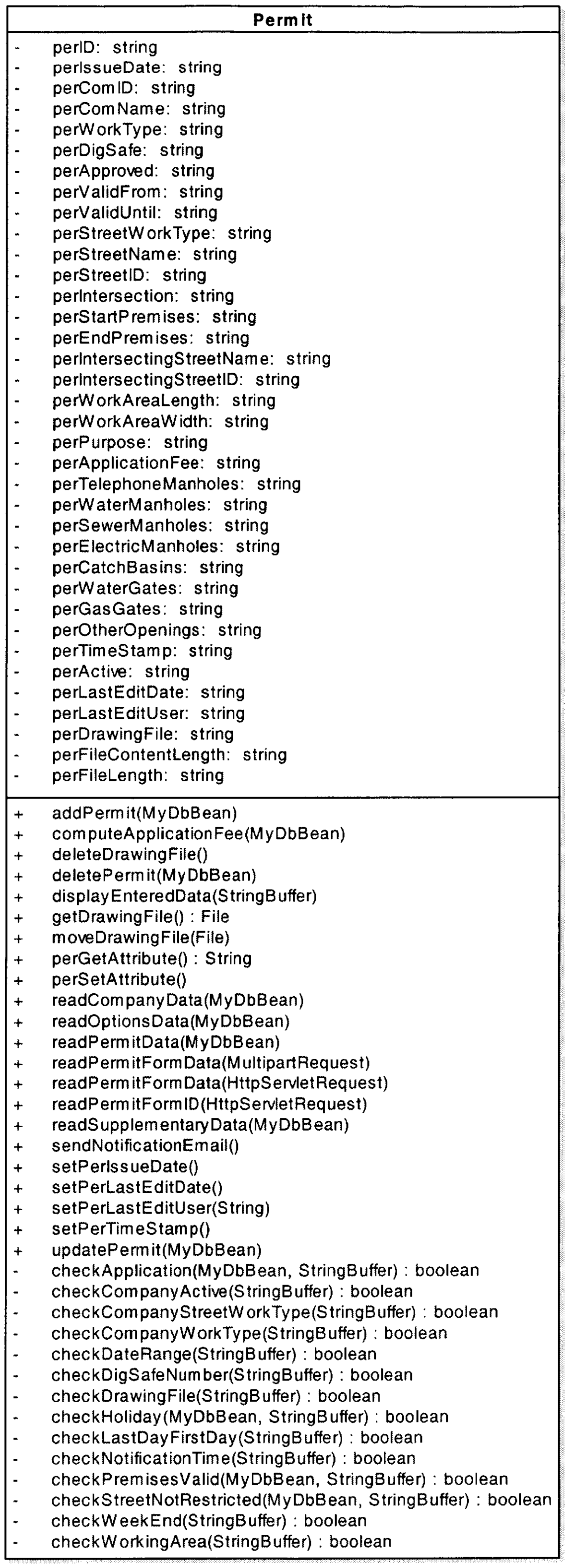
As stated earlier, the class diagram technique is central within object-oriented methods. A class diagram describes the types of the objects in the system and various kinds of static relationships that exist among them. Below we show the various classes that are central to the Pavement Permit system application. The classes are shown along with their attributes and operations.

6.2.1 Permit Class

The Permit class has attributes that pertain to the permits that are issued by the system, in response to applications made by the internal or external users. The operations within this class take care of the all the attributes related to the individual permits and also of the other operations that are performed on the permits data by the system. The diagram below shows the permit class along with its attributes and operations.

The symbols used have the following meanings:

- + Public class
- - Private class
- $ Static class.



```
Permit
-    perID: string
-    perIssueDate: string
-    perComID: string
-    perComName: string
-    perWorkType: string
-    perDigSafe: string
-    perApproved: string
-    perValidFrom: string
-    perValidUntil: string
-    perStreetWorkType: string
-    perStreetName: string
-    perStreetID: string
-    perIntersection: string
-    perStartPremises: string
-    perEndPremises: string
-    perIntersectingStreetName: string
-    perIntersectingStreetID: string
-    perWorkAreaLength: string
-    perWorkAreaWidth: string
-    perPurpose: string
-    perApplicationFee: string
-    perTelephoneManholes: string
-    perWaterManholes: string
-    perSewerManholes: string
-    perElectricManholes: string
-    perCatchBasins: string
-    perWaterGates: string
-    perGasGates: string
-    perOtherOpenings: string
-    perTimeStamp: string
-    perActive: string
-    perLastEditDate: string
-    perLastEditUser: string
-    perDrawingFile: string
-    perFileContentLength: string
-    perFileLength: string

+    addPermit(MyDbBean)
+    computeApplicationFee(MyDbBean)
+    deleteDrawingFile()
+    deletePermit(MyDbBean)
+    displayEnteredData(StringBuffer)
+    getDrawingFile() : File
+    moveDrawingFile(File)
+    perGetAttribute() : String
+    perSetAttribute()
+    readCompanyData(MyDbBean)
+    readOptionsData(MyDbBean)
+    readPermitData(MyDbBean)
+    readPermitFormData(MultipartRequest)
+    readPermitFormData(HttpServletRequest)
+    readPermitFormID(HttpServletRequest)
+    readSupplementaryData(MyDbBean)
+    sendNotificationEmail()
+    setPerIssueDate()
+    setPerLastEditDate()
+    setPerLastEditUser(String)
+    setPerTimeStamp()
+    updatePermit(MyDbBean)
-    checkApplication(MyDbBean, StringBuffer) : boolean
-    checkCompanyActive(StringBuffer) : boolean
-    checkCompanyStreetWorkType(StringBuffer) : boolean
-    checkCompanyWorkType(StringBuffer) : boolean
-    checkDateRange(StringBuffer) : boolean
-    checkDigSafeNumber(StringBuffer) : boolean
-    checkDrawingFile(StringBuffer) : boolean
-    checkHoliday(MyDbBean, StringBuffer) : boolean
-    checkLastDayFirstDay(StringBuffer) : boolean
-    checkNotificationTime(StringBuffer) : boolean
-    checkPremisesValid(MyDbBean, StringBuffer) : boolean
-    checkStreetNotRestricted(MyDbBean, StringBuffer) : boolean
-    checkWeekEnd(StringBuffer) : boolean
-    checkWorkingArea(StringBuffer) : boolean
```

Figure 6-2 Permit Class

6.2.2 Company Class

The Company class has attributes that pertain to the companies that are allowed by the company to apply for the permit. The operations within the company class take care of the all the attributes related to the companies and also related to operations on the companies data by the application. The diagram below shows the company class along with its attributes and methods.

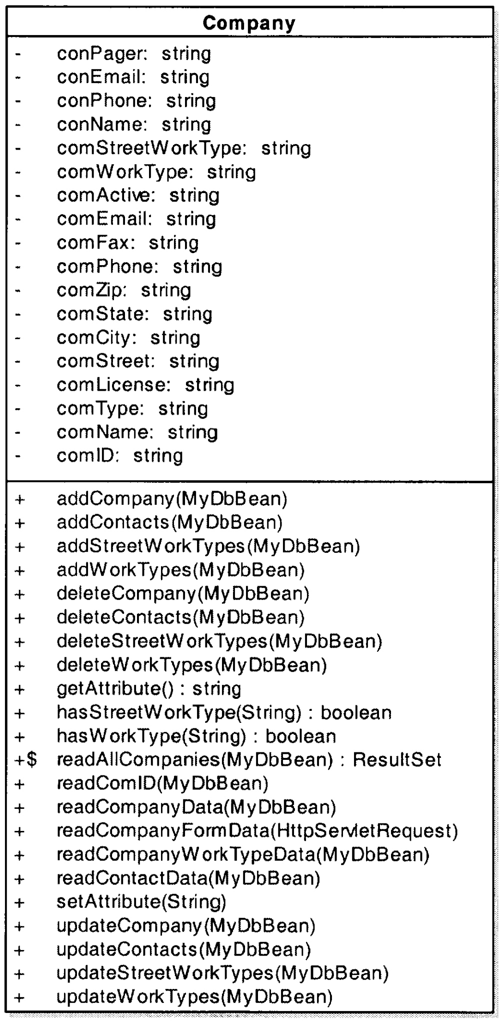

Company
- conPager: string
- conEmail: string
- conPhone: string
- conName: string
- comStreetWorkType: string
- comWorkType: string
- comActive: string
- comEmail: string
- comFax: string
- comPhone: string
- comZip: string
- comState: string
- comCity: string
- comStreet: string
- comLicense: string
- comType: string
- comName: string
- comID: string
+ addCompany(MyDbBean)
+ addContacts(MyDbBean)
+ addStreetWorkTypes(MyDbBean)
+ addWorkTypes(MyDbBean)
+ deleteCompany(MyDbBean)
+ deleteContacts(MyDbBean)
+ deleteStreetWorkTypes(MyDbBean)
+ deleteWorkTypes(MyDbBean)
+ getAttribute() : string
+ hasStreetWorkType(String) : boolean
+ hasWorkType(String) : boolean
+$ readAllCompanies(MyDbBean) : ResultSet
+ readComID(MyDbBean)
+ readCompanyData(MyDbBean)
+ readCompanyFormData(HttpServletRequest)
+ readCompanyWorkTypeData(MyDbBean)
+ readContactData(MyDbBean)
+ setAttribute(String)
+ updateCompany(MyDbBean)
+ updateContacts(MyDbBean)
+ updateStreetWorkTypes(MyDbBean)
+ updateWorkTypes(MyDbBean)

Figure 6-3 Company Class

6.2.3 Restriction Class

The Restrictions class has attributes that pertain to the street restrictions that are applied towards review of application for the permit. The operations within the restrictions class take care of the all the attributes related to the individual restrictions and also related to operations on the restrictions data by the application. The diagram below shows the Restriction class along with its attributes and methods.

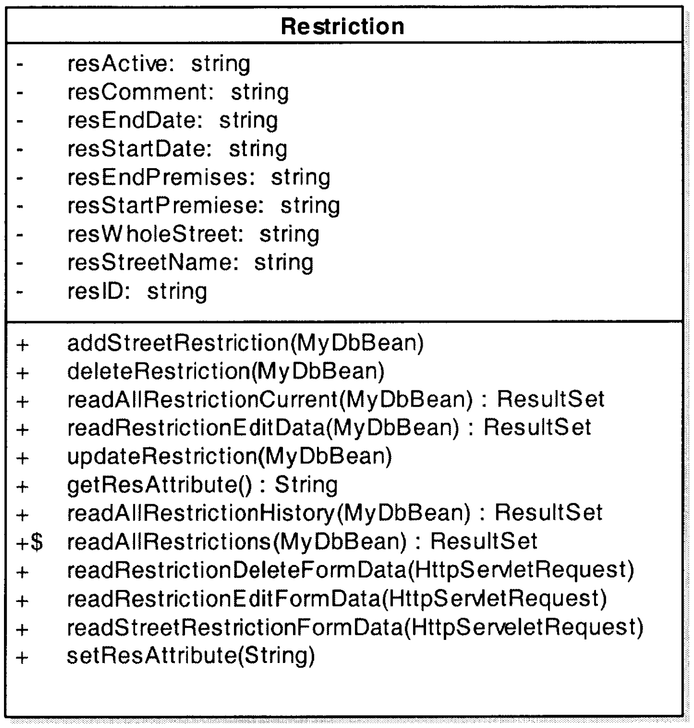

Restriction
- resActive: string
- resComment: string
- resEndDate: string
- resStartDate: string
- resEndPremises: string
- resStartPremiese: string
- resWholeStreet: string
- resStreetName: string
- resID: string
+ addStreetRestriction(MyDbBean)
+ deleteRestriction(MyDbBean)
+ readAllRestrictionCurrent(MyDbBean) : ResultSet
+ readRestrictionEditData(MyDbBean) : ResultSet
+ updateRestriction(MyDbBean)
+ getResAttribute() : String
+ readAllRestrictionHistory(MyDbBean) : ResultSet
+$ readAllRestrictions(MyDbBean) : ResultSet
+ readRestrictionDeleteFormData(HttpServletRequest)
+ readRestrictionEditFormData(HttpServletRequest)
+ readStreetRestrictionFormData(HttpServeletRequest)
+ setResAttribute(String)

Figure 6-4 Restrictions Class

6.2.4 Holiday Class

The Holiday class has attributes that pertain to the holidays that apply within the year. This works in conjunction with street restrictions towards review of application for the permit. The operations within the Holiday class takes care of the all the attributes related to the individual holidays and also related to operations on them by the application. The diagram below shows the Holiday class along with its attributes and methods.

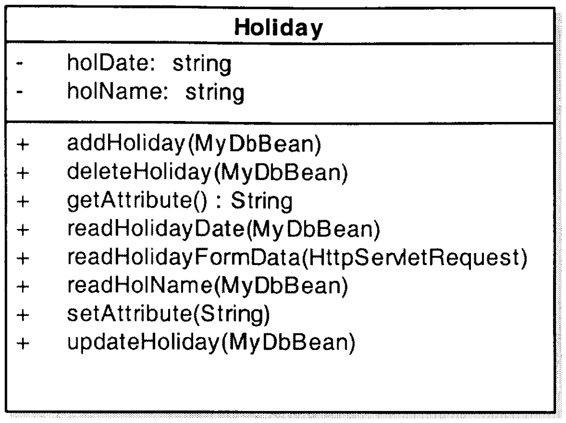

Holiday
- holDate: string
- holName: string
+ addHoliday(MyDbBean)
+ deleteHoliday(MyDbBean)
+ getAttribute() : String
+ readHolidayDate(MyDbBean)
+ readHolidayFormData(HttpServletRequest)
+ readHolName(MyDbBean)
+ setAttribute(String)
+ updateHoliday(MyDbBean)

Figure 6-5 Holiday Class

6.2.5 Option Class

The Options class has attributes that pertain to the miscellaneous options that apply to the other parameters needed for permit issuance. The operations within this class take care of the all the attributes related to the various options and also related to operations on the options data by the application. The diagram below shows the Option class along with its attributes and methods.

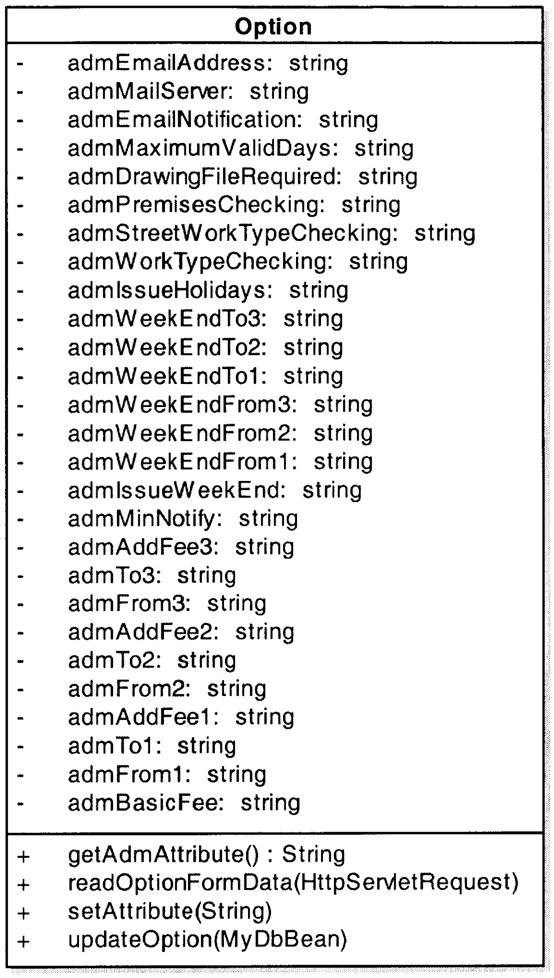
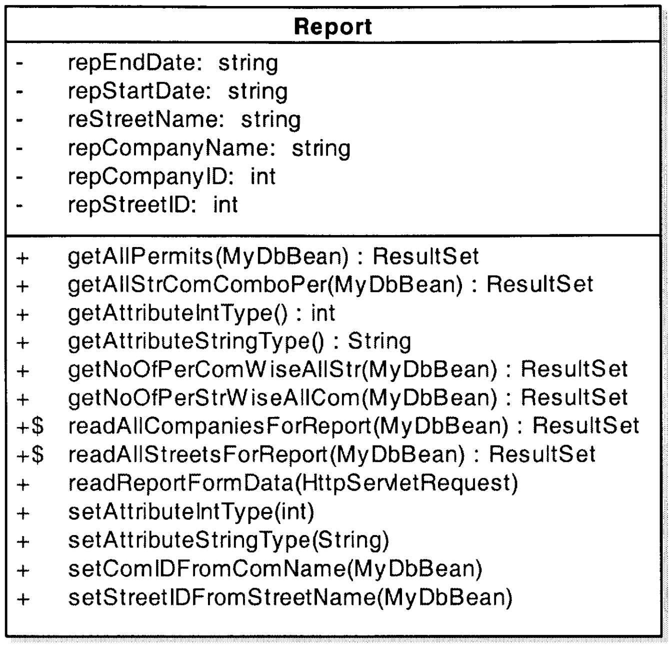

Option
- admEmailAddress: string
- admMailServer: string
- admEmailNotification: string
- admMaximumValidDays: string
- admDrawingFileRequired: string
- admPremisesChecking: string
- admStreetWorkTypeChecking: string
- admWorkTypeChecking: string
- admIssueHolidays: string
- admWeekEndTo3: string
- admWeekEndTo2: string
- admWeekEndTo1: string
- admWeekEndFrom3: string
- admWeekEndFrom2: string
- admWeekEndFrom1: string
- admIssueWeekEnd: string
- admMinNotify: string
- admAddFee3: string
- admTo3: string
- admFrom3: string
- admAddFee2: string
- admTo2: string
- admFrom2: string
- admAddFee1: string
- admTo1: string
- admFrom1: string
- admBasicFee: string
+ getAdmAttribute() : String
+ readOptionFormData(HttpServletRequest)
+ setAttribute(String)
+ updateOption(MyDbBean)

Figure 6-6 Option Class

6.2.6 Report Class

The Report class has attributes that pertain to the generation of reports by the system. In line with the other classes, this class also has attributes and operation related to reports. The operations within this class take care of the all the attributes related to the various reports that are generated by the system. The diagram below shows the Report class along with its attributes and methods.



```
Report
-    repEndDate: string
-    repStartDate: string
-    reStreetName: string
-    repCompanyName: string
-    repCompanyID: int
-    repStreetID: int

+    getAllPermits(MyDbBean) : ResultSet
+    getAllStrComComboPer(MyDbBean) : ResultSet
+    getAttributeIntType() : int
+    getAttributeStringType() : String
+    getNoOfPerComWiseAllStr(MyDbBean) : ResultSet
+    getNoOfPerStrWiseAllCom(MyDbBean) : ResultSet
+$   readAllCompaniesForReport(MyDbBean) : ResultSet
+$   readAllStreetsForReport(MyDbBean) : ResultSet
+    readReportFormData(HttpServletRequest)
+    setAttributeIntType(int)
+    setAttributeStringType(String)
+    setComIDFromComName(MyDbBean)
+    setStreetIDFromStreetName(MyDbBean)
```

Figure 6-7 Report Class

6.2.7 FAQ Class

The FAQ class has attributes that pertain to the generation of FAQs page by the internal user using the system. In line with the other classes, this class also has attributes and operation related to generation of FAQs. The operations within this class take care of the all the attributes related to the generation of FAQs. The diagram below shows the FAQ class along with its attributes and methods.

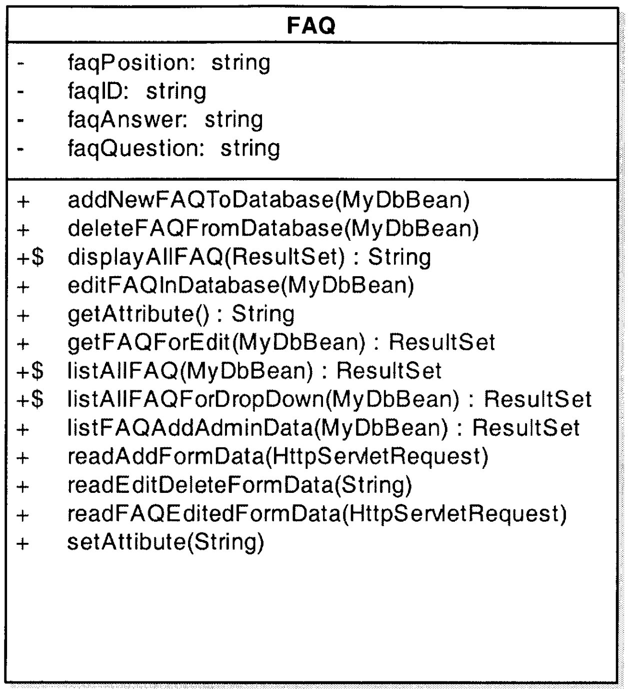

FAQ
- faqPosition: string
- faqID: string
- faqAnswer: string
- faqQuestion: string
+ addNewFAQToDatabase(MyDbBean)
+ deleteFAQFromDatabase(MyDbBean)
+$ displayAllFAQ(ResultSet) : String
+ editFAQInDatabase(MyDbBean)
+ getAttribute() : String
+ getFAQForEdit(MyDbBean) : ResultSet
+$ listAllFAQ(MyDbBean) : ResultSet
+$ listAllFAQForDropDown(MyDbBean) : ResultSet
+ listFAQAddAdminData(MyDbBean) : ResultSet
+ readAddFormData(HttpServletRequest)
+ readEditDeleteFormData(String)
+ readFAQEditedFormData(HttpServletRequest)
+ setAttibute(String)

Figure 6-8 FAQ Class

6.2.8 Class Diagram for Pavement Permit System

The diagram below shows the relationship within the various classes shown above. This diagram shows the interaction between the classes. It is also appropriate to mention that these sets of classes need to be supported by other supplementary classes that are not shown below. These supplementary classes are language related like MyHTML class, which has built in functionality to create HTML tables, dropdowns, and so on in the Pavement Permit System look and feel format. The MyCalendar class has functionality to deal with all the date and time related issues for the project. There are a host of other such classes, which may be looked up in the project documentation.

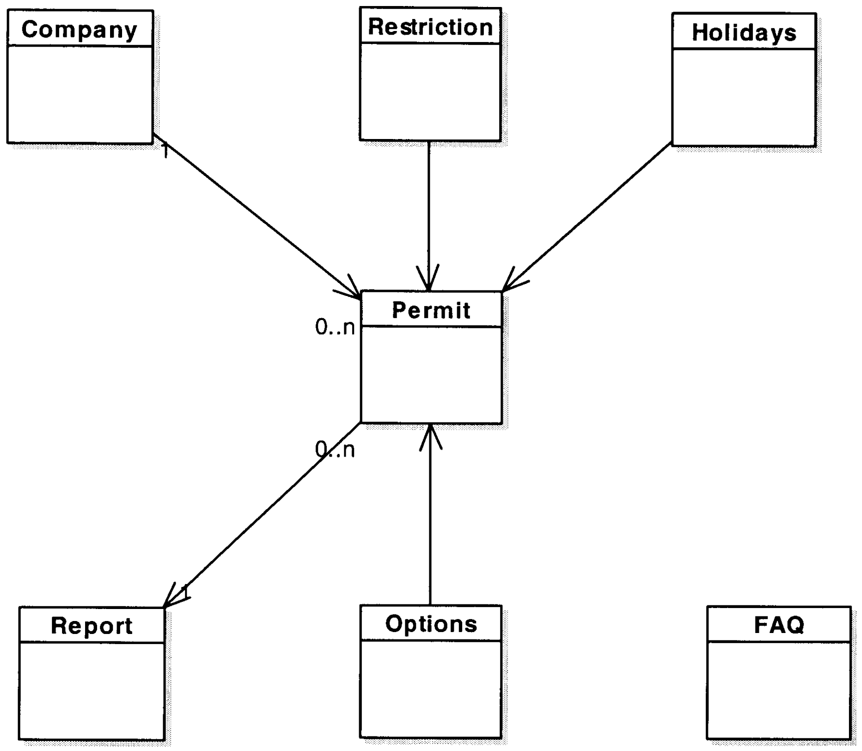


Figure 6-9 Class Diagram-Pavement Permit System

The class diagram for the pavement Permit system shown above is very similar to the actual data-model for the project. The similarities are by design. Below we show the data-model for the project. The diagram shows the data-model in detail. The only differences are that there are some additional tables in the data-model like “CompanyWorkType” etc. These are also present in the class diagram but are incorporated within the company class itself. The reason for this is that the data-model has been normalized and therefore items were placed in separate tables items to avoid duplication. Moreover, in the class diagrams there are also some helper classes like “MyHTML” and so on, which are not present in the data-model.

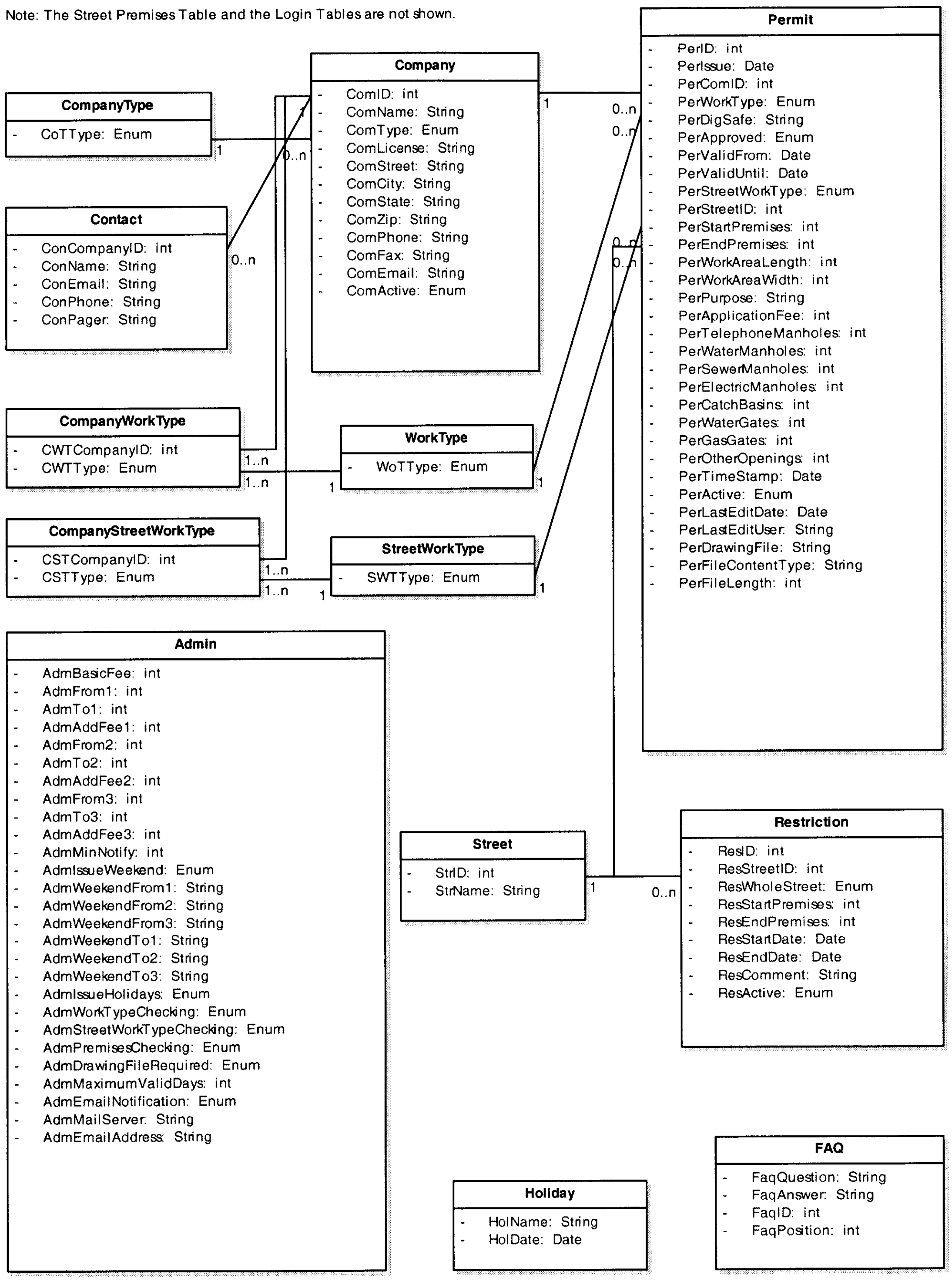


Figure 6-10 Pavement Permit System Data-Model (Source: Web Application Development Using Open Source Technologies and Java by Wolfgang Andreas Klimke)

6.3 Sequence Diagram

The interaction diagrams are models that describe how groups of objects collaborate in some behavior. The sequence diagram is one way of representing the interaction diagrams.

Within the Pavement permit system there are various instances when objects are added, edited, deleted or listed and so on. Below, we show a set of sequence diagrams that exhibit the sequence of activities involved in activities mentioned earlier.

6.3.1 Sequence Diagram For Adding A Street Restriction

The diagram below shows the sequence of activities and operations in adding a street restriction.

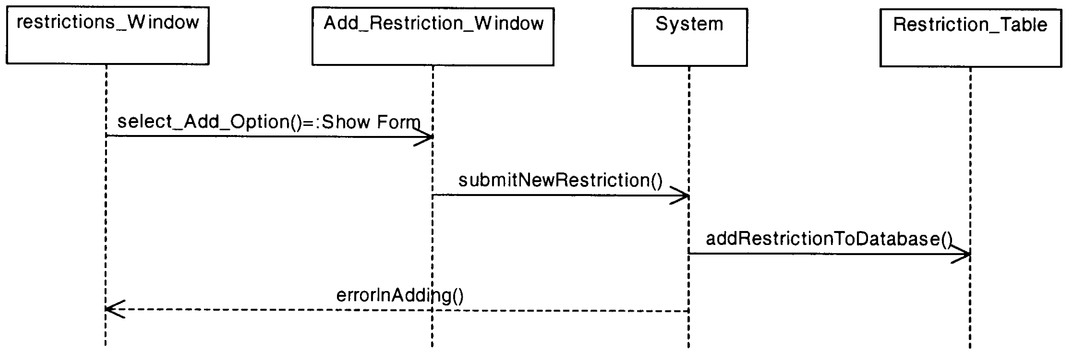


Figure 6-11 Sequence Diagram For Adding A Street Restriction

6.3.1.1 Sequence Diagram For Editing Company Data

The diagram below shows the sequence of activities and operations in editing data on a company.

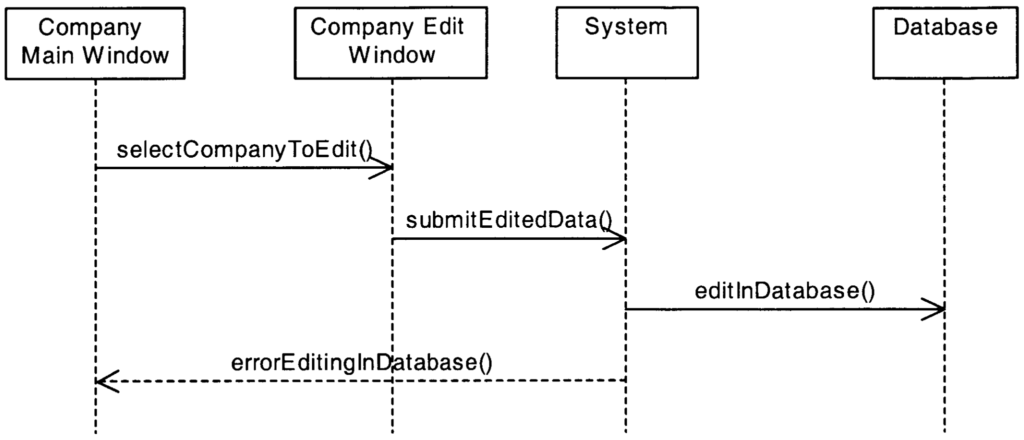


Figure 6-12 Sequence Diagram For Editing Company Data

6.3.2 Sequence Diagram For Deleting Data On A Holiday

The diagram below shows the sequence of activities and operations in deleting data on a holiday.

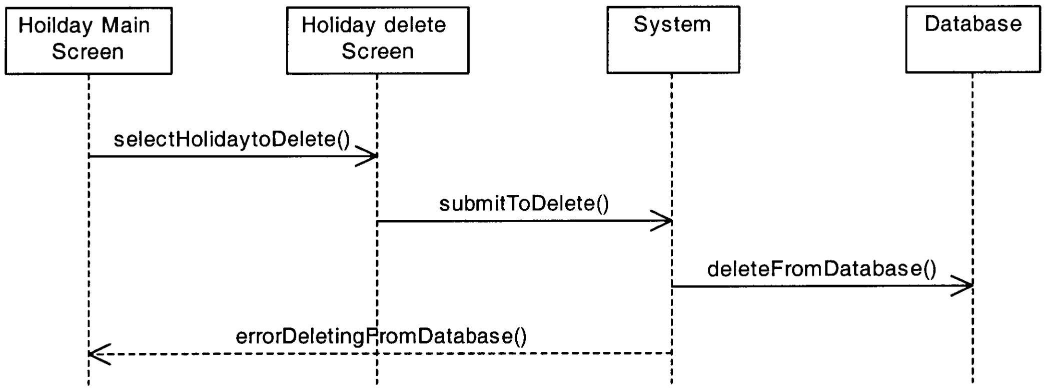


Figure 6-13 Sequence Diagram For Deleting Data On A Holiday

6.3.3 Sequence Diagram For Permit Issuance

The diagram below shows the sequence of activities and operations in applying for, review and issuance or rejection of permit.

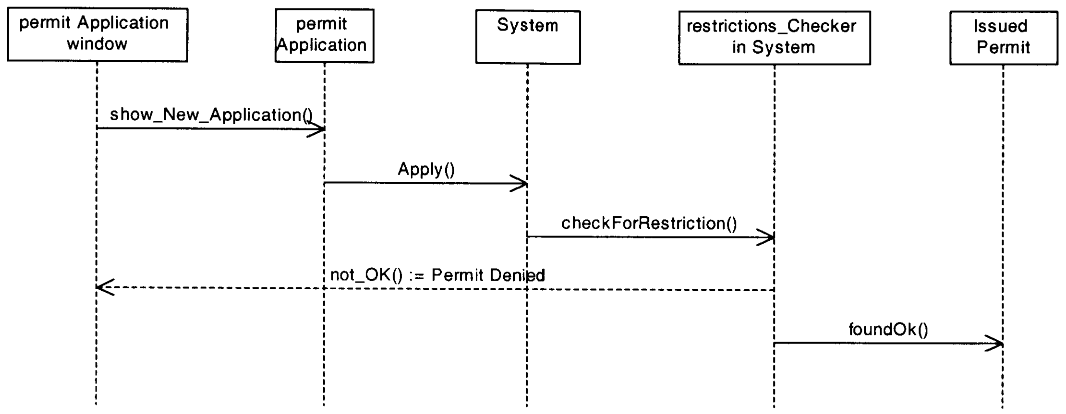


Figure 6-14 Sequence Diagram For Permit Issuance

6.4 State Diagrams

State Diagrams are familiar technique to describe the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object.

Within the pavement permit system we are showing below the state diagram for the permits. The diagram shows the various states that the permit is within the permit application process.

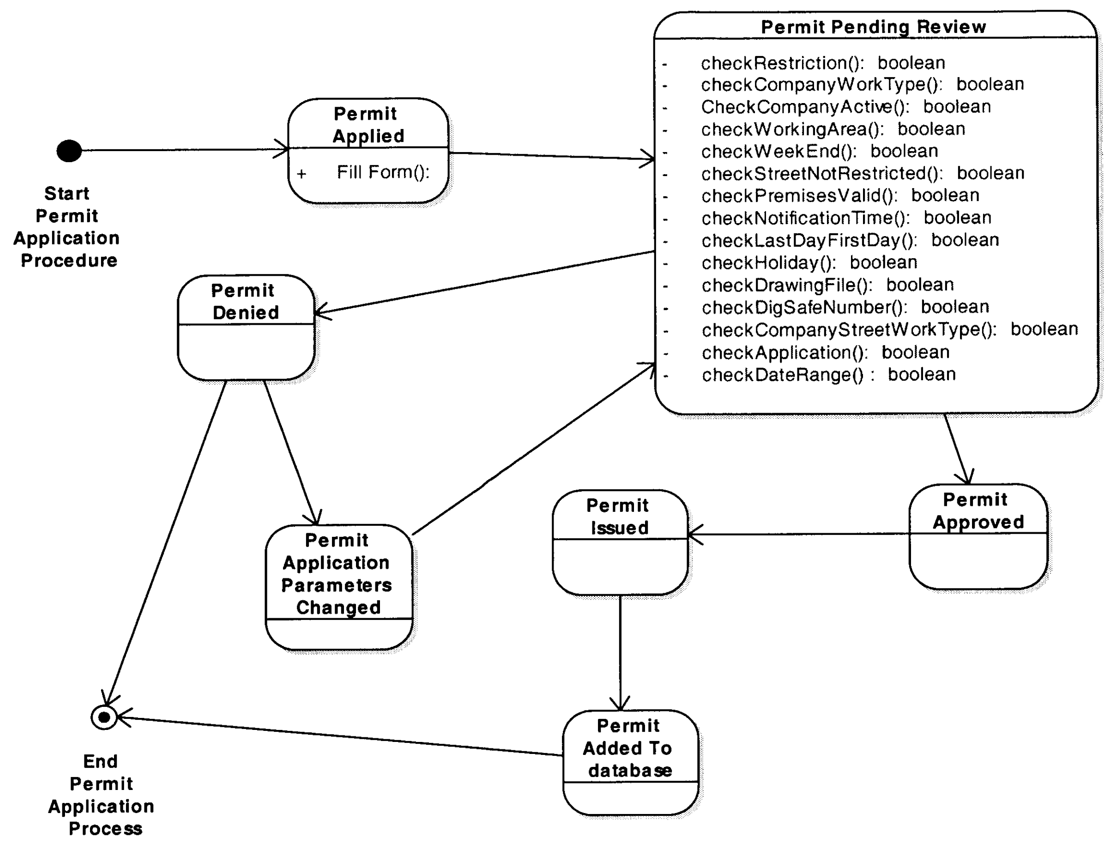

Start Permit Application Procedure
Permit Applied
+ Fill Form():
Permit Pending Review
- checkRestriction(): boolean
- checkCompanyWorkType(): boolean
- CheckCompanyActive(): boolean
- checkWorkingArea(): boolean
- checkWeekEnd(): boolean
- checkStreetNotRestricted(): boolean
- checkPremisesValid(): boolean
- checkNotificationTime(): boolean
- checkLastDayFirstDay(): boolean
- checkHoliday(): boolean
- checkDrawingFile(): boolean
- checkDigSafeNumber(): boolean
- checkCompanyStreetWorkType(): boolean
- checkApplication(): boolean
- checkDateRange() : boolean
Permit Denied
Permit Application Parameters Changed
Permit Issued
Permit Approved
Permit Added To database
End Permit Application Process

Figure 6-15 Permits: State Diagram

6.5 Component Diagrams

A component diagram shows the various components in a system and their dependencies. The diagram below shows the basic component diagram for the application. It shows the various components of the system and their interaction.

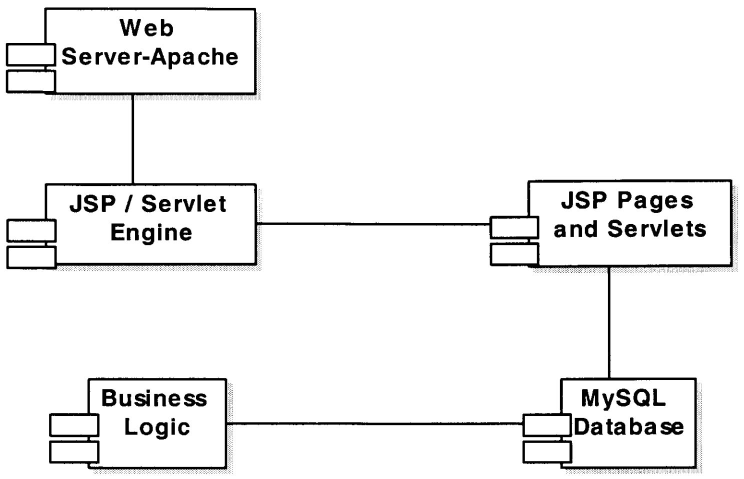


Figure 6-16 Component Diagram

In the figure above, the Business Logic refers to the business rules that the application aims to model. It consists of the rules that the customers' organization follows and which the application maps. The other components are self-explanatory.

7 Conclusions

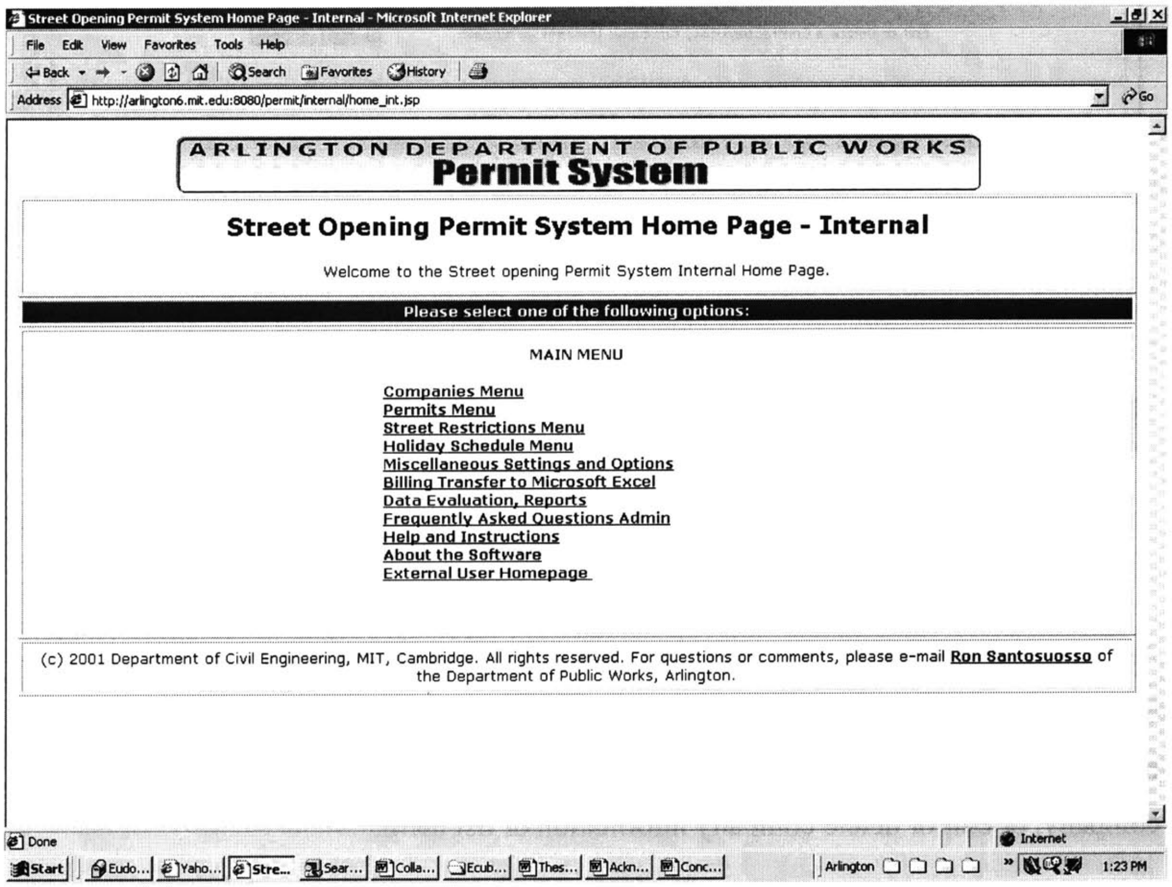
Today, when a substantial portion of software applications being developed is of intensely distributed nature and sophisticated configuration, there is a more pronounced need for more logical and adept technology for aiding the software process. The object oriented technology and UML have been a great help towards this end.

We have seen that during the development of the Pavement Permit System, the application of the object oriented paradigm and use of UML for the software process has been very successful. The requirements specification for the project was substantially simplified on account of use of the UML technology. Similarly, object oriented technology and UML also made the design of the project much easier and also more understandable across the team and other stakeholders. The project was completed in time and the application not only mirrored the requirements specification, but also the software design. In the end the software developed met the users expectation and therefore was a success. We are providing the screen shots of the actual application in the appendix for the use cases that we had discussed in chapter five.

Personally, I found the use of the object oriented technology and UML as a very helpful. It helped me immensely in understanding the bigger picture of the software process. I feel much more adapt at taking up large-scale projects now that I have learned these techniques.

8 Appendix

Herein we are providing a representative collection of Screenshots for the Pavement Permit System. The collection consists of screenshots of all the main screens, representative screenshots of some of the common operations, of Listing, Adding, Editing and Deleting and for the Billing Transfer and Reports.


Street Opening Permit System Home Page - Internal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/home_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System

Street Opening Permit System Home Page - Internal

Welcome to the Street opening Permit System Internal Home Page.

Please select one of the following options:

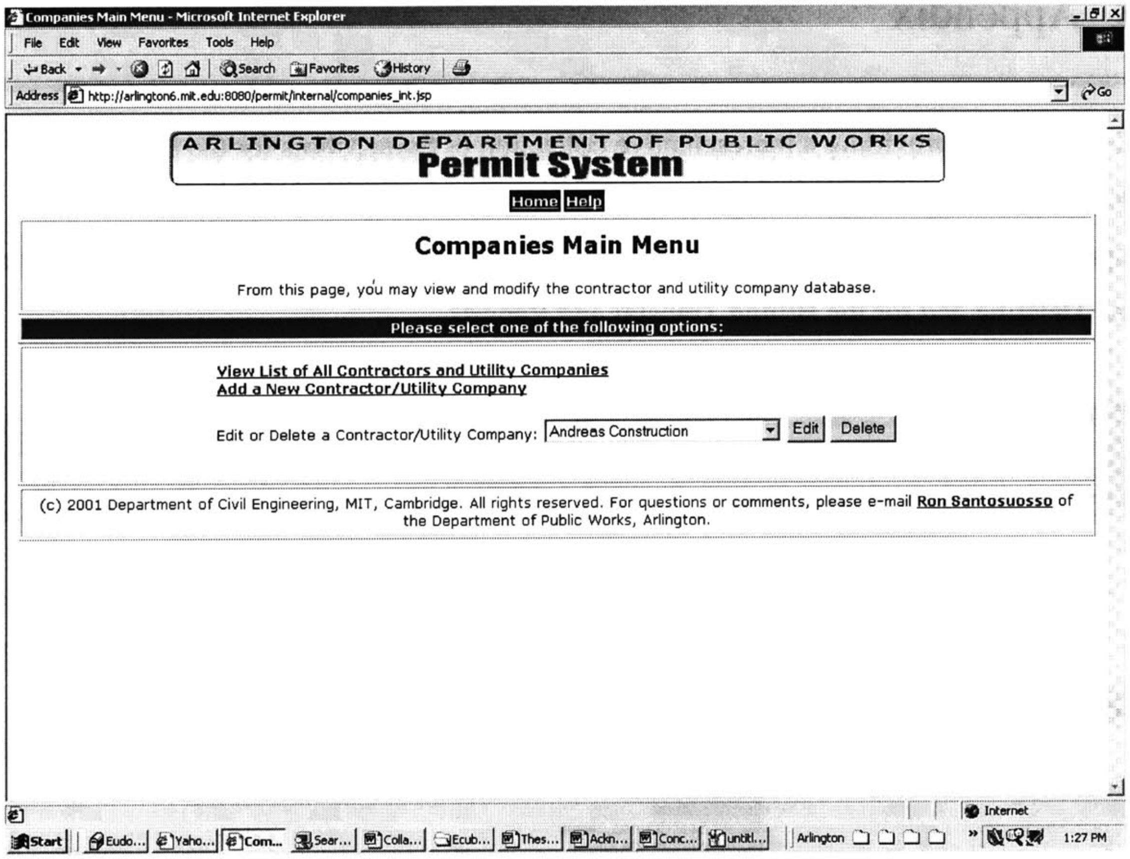
MAIN MENU

Companies Menu
Permits Menu
Street Restrictions Menu
Holiday Schedule Menu
Miscellaneous Settings and Options
Billing Transfer to Microsoft Excel
Data Evaluation, Reports
Frequently Asked Questions Admin
Help and Instructions
About the Software
External User Homepage

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-1 Pavement Permit System: Internal User Main

This page is the home page for the internal user. After successful login he comes to this page and then he can select any of the links for undertaking whatever operation he desires to perform.



ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System

Companies Main Menu

From this page, you may view and modify the contractor and utility company database.

Please select one of the following options:

View List of All Contractors and Utility Companies
Add a New Contractor/Utility Company

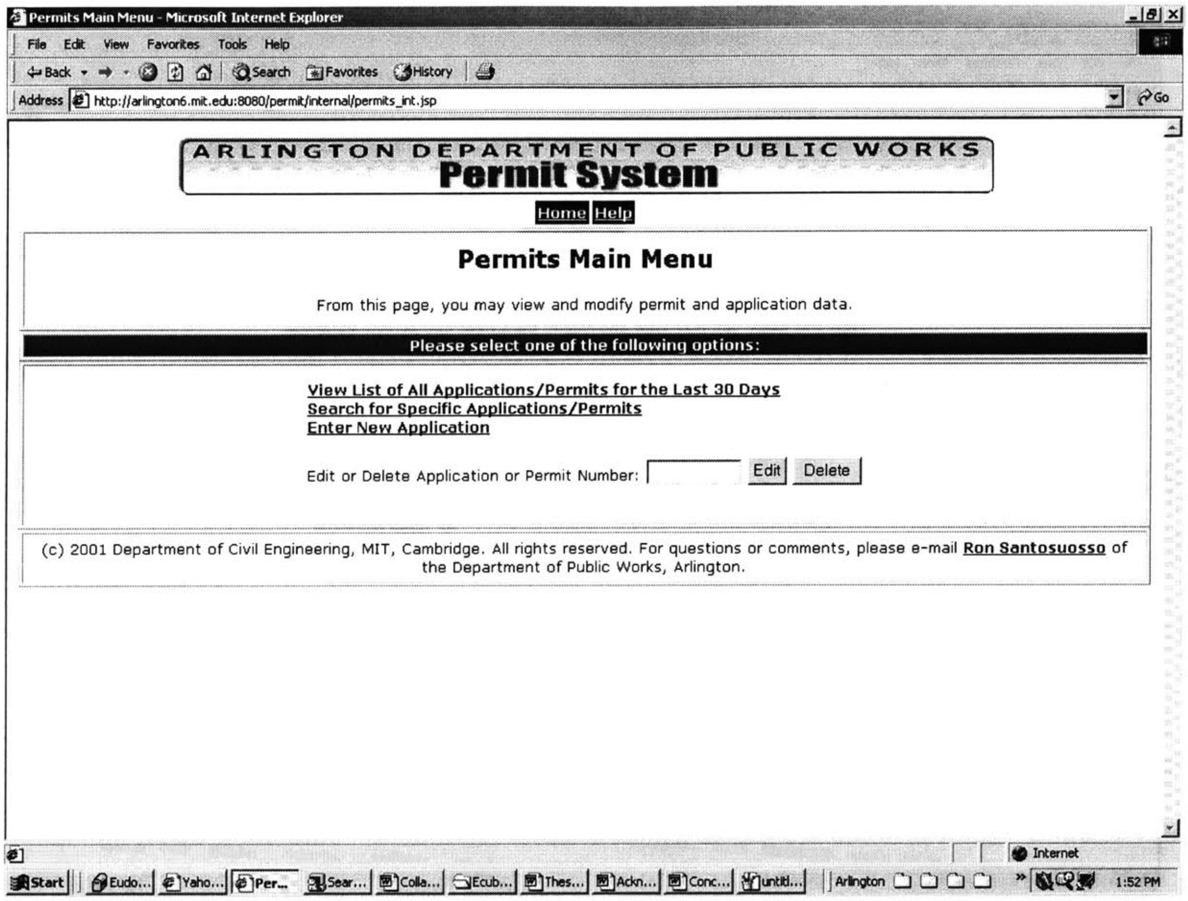
Edit or Delete a Contractor/Utility Company: Andreas Construction

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Companies Main Menu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/companies_int.jsp
Go
Home Help
Edit Delete
Internet
Start
1:27 PM

Figure 8-2 Companies Main Menu

An important element for the system administration is the manipulation of the companies' related data. The figure above is the companies' main menu, which the internal user can access from the main page. Here he can follow the link to add a new company, or edit or delete company information or list them.


Permits Main Menu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/permits_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Help

Permits Main Menu

From this page, you may view and modify permit and application data.

Please select one of the following options:

View List of All Applications/Permits for the Last 30 Days
Search for Specific Applications/Permits
Enter New Application

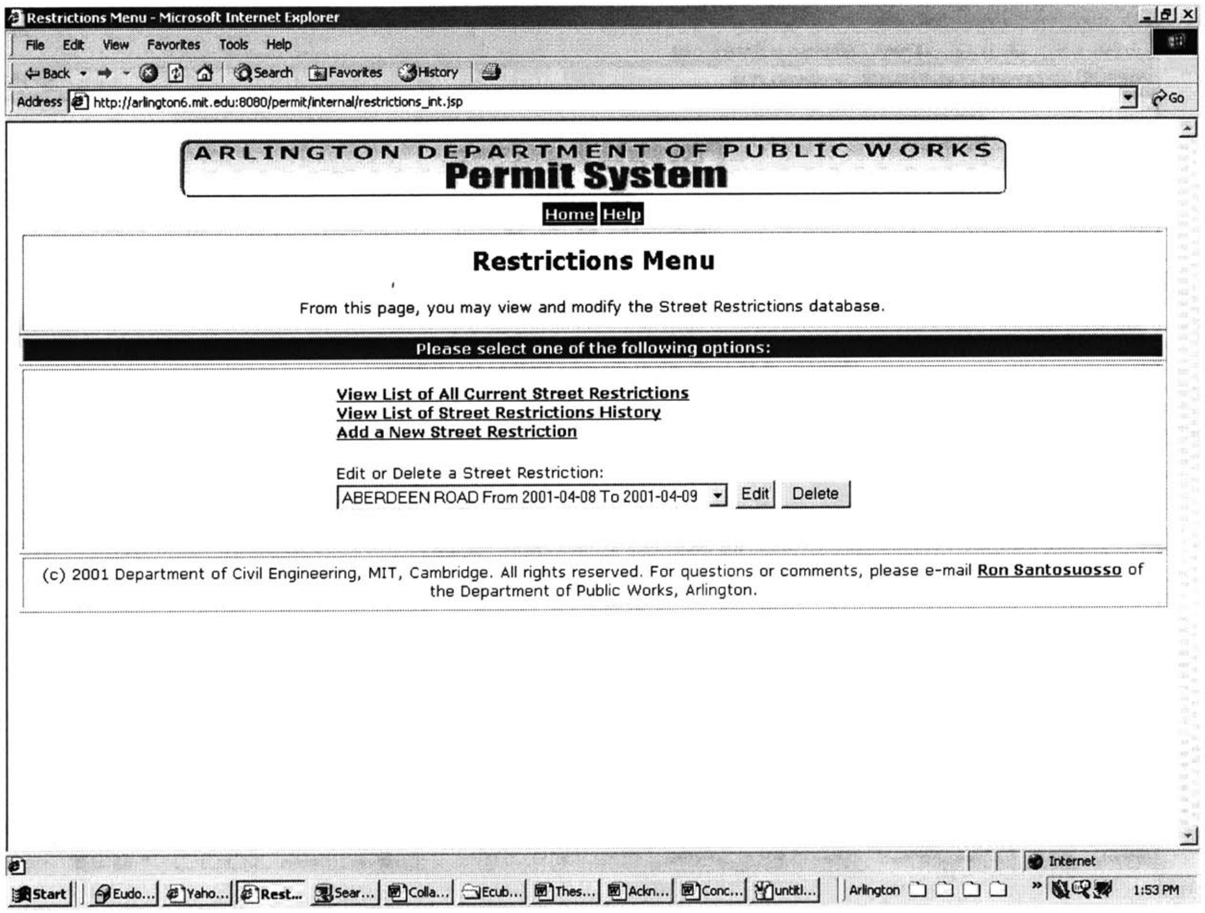
Edit or Delete Application or Permit Number: Edit Delete

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Internet
Start Eudo... Yaho... Per... Sear... Colla... Ecub... Thes... Ackn... Conc... untitl... Arlington
1:52 PM

Figure 8-3 Permits Main menu

The Permits page is the core page of the application. The figure above is the Permits' Main Menu, which the internal user can access from the main page. From this page the user can also follow the links and view permits list or search for permits and also do the other routine administration works for the permits.


Restrictions Menu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/restrictions_int.jsp
Go
Home Help
Edit Delete
Internet
Start
1:53 PM

ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System

Restrictions Menu

From this page, you may view and modify the Street Restrictions database.

Please select one of the following options:

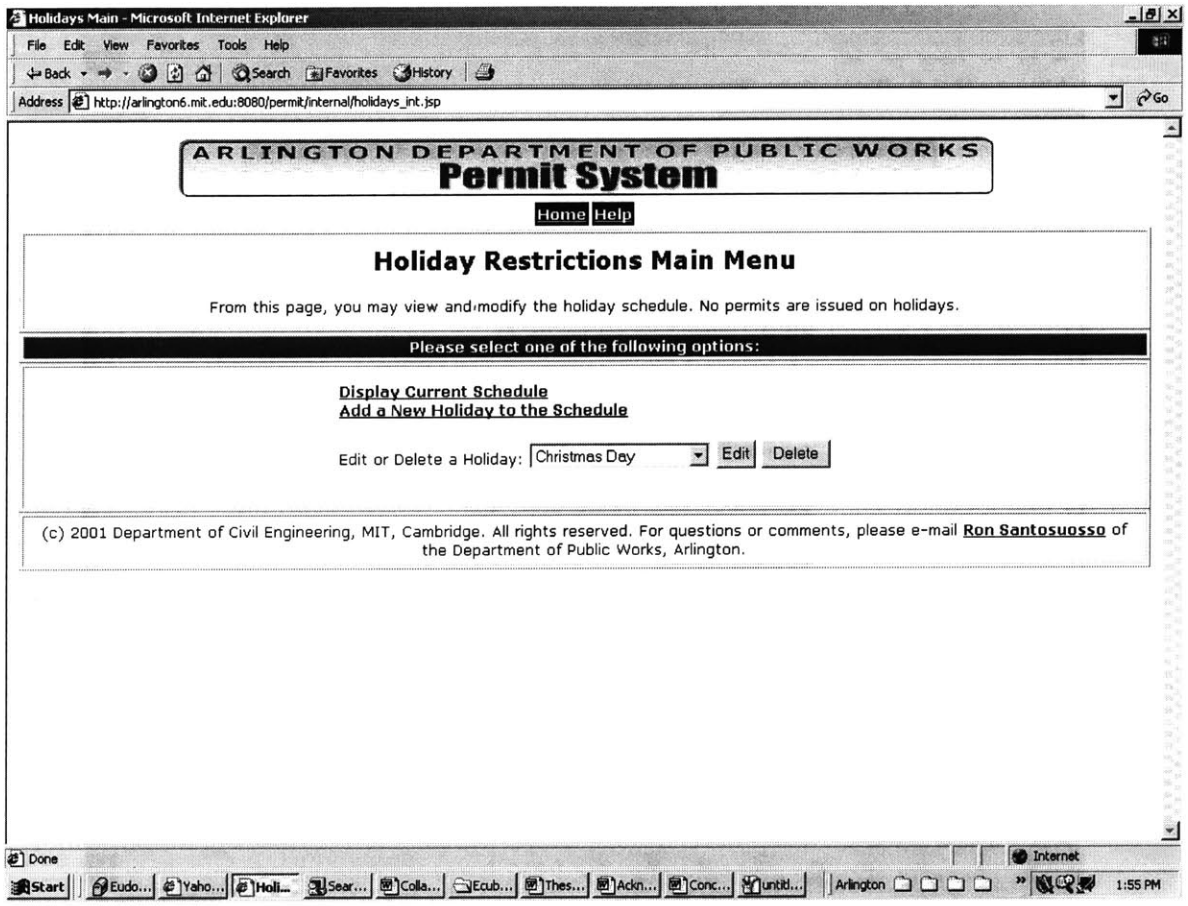
View List of All Current Street Restrictions
View List of Street Restrictions History
Add a New Street Restriction

Edit or Delete a Street Restriction:
ABERDEEN ROAD From 2001-04-08 To 2001-04-09

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-4 Restrictions Main Menu

The restrictions on the streets have to be administered for making the application up to date in checking the parameters of the permits application. The figure above is the Restrictions' main menu (Streets), which the internal user can access from the main page. The user can administer the street restrictions from here by following the links.


Holidays Main - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/holidays_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Help

Holiday Restrictions Main Menu

From this page, you may view and modify the holiday schedule. No permits are issued on holidays.

Please select one of the following options:

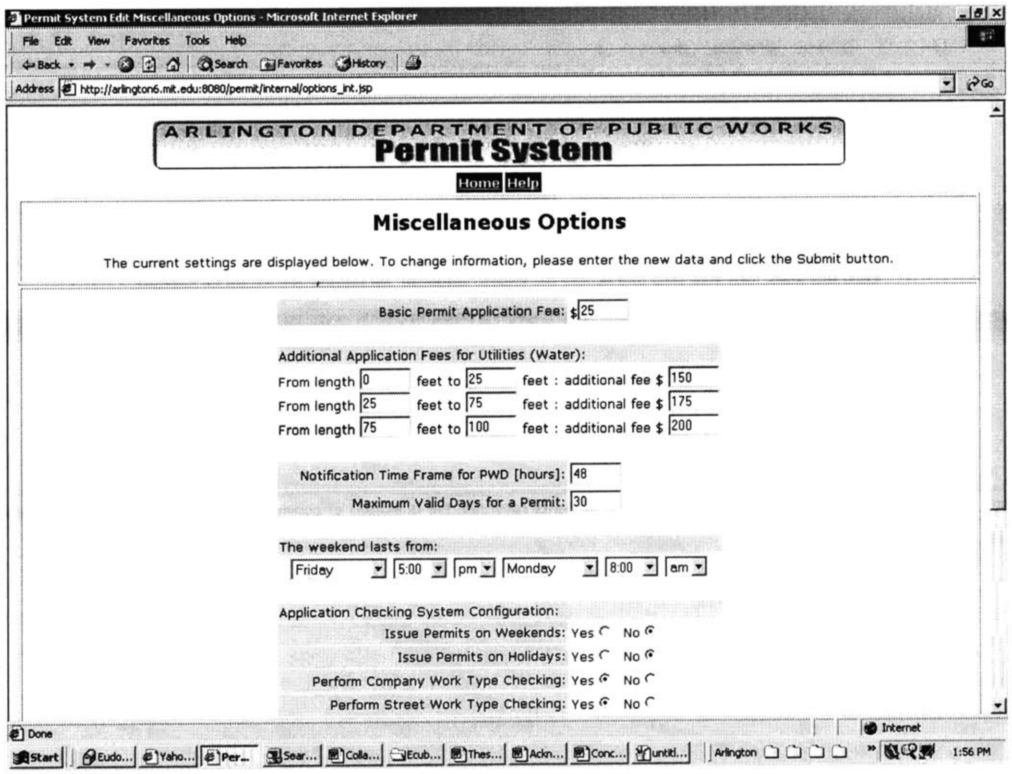
Display Current Schedule
Add a New Holiday to the Schedule

Edit or Delete a Holiday: Christmas Day [Edit] [Delete]

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-5 Holiday Restrictions Main Menu

The figure above is the Holidays restrictions' main menu, which the internal user can access from the main page. This page allows for administration of the holidays restrictions data.


Permit System Edit Miscellaneous Options - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://arlington6.mit.edu:8080/permit/internal/options_int.jsp
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Help

Miscellaneous Options

The current settings are displayed below. To change information, please enter the new data and click the Submit button.

Basic Permit Application Fee: $25

Additional Application Fees for Utilities (Water):

From length	0	feet to	25	feet : additional fee $	150
From length	25	feet to	75	feet : additional fee $	175
From length	75	feet to	100	feet : additional fee $	200

Notification Time Frame for PWD [hours]: 48

Maximum Valid Days for a Permit: 30

The weekend lasts from:

Friday 5:00 pm Monday 8:00 am

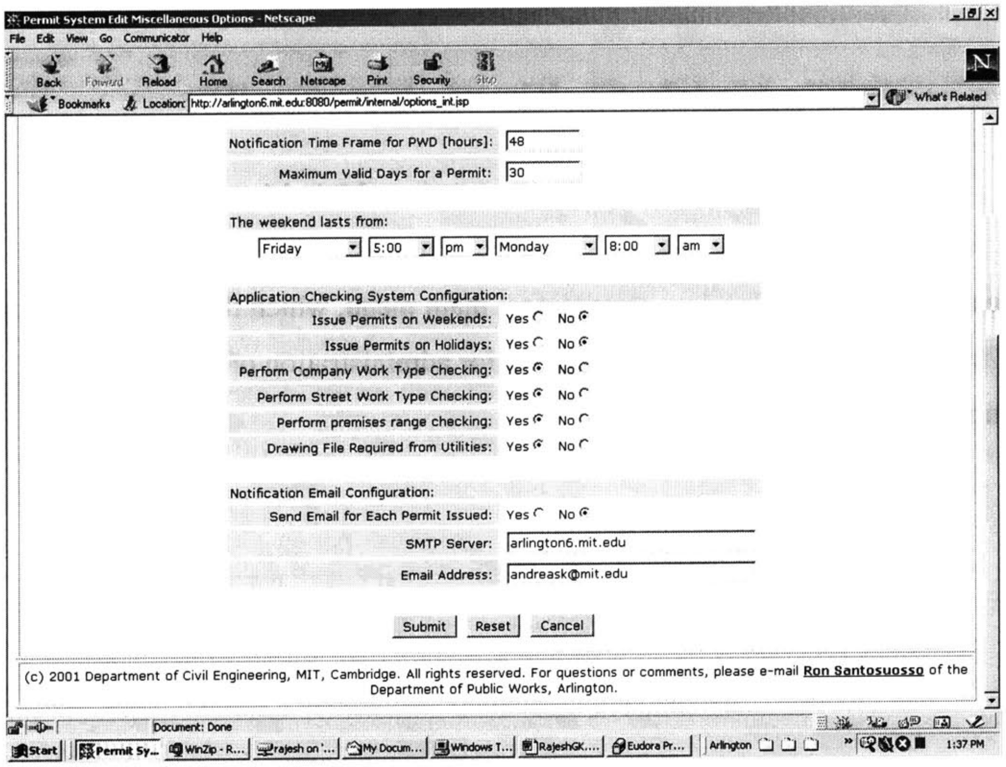
Application Checking System Configuration:

Issue Permits on Weekends: Yes [✓ No]

Issue Permits on Holidays: Yes [✓ No]

Perform Company Work Type Checking: [✓ Yes] No

Perform Street Work Type Checking: [✓ Yes] No


Permit System Edit Miscellaneous Options - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://arlington6.mit.edu:8080/permit/internal/options_int.jsp
What's Related

Notification Time Frame for PWD [hours]: 48

Maximum Valid Days for a Permit: 30

The weekend lasts from:

Friday 5:00 pm Monday 8:00 am

Application Checking System Configuration:

Issue Permits on Weekends: Yes [✓ No]

Issue Permits on Holidays: Yes [✓ No]

Perform Company Work Type Checking: [✓ Yes] No

Perform Street Work Type Checking: [✓ Yes] No

Perform premises range checking: [✓ Yes] No

Drawing File Required from Utilities: [✓ Yes] No

Notification Email Configuration:

Send Email for Each Permit Issued: Yes [✓ No]

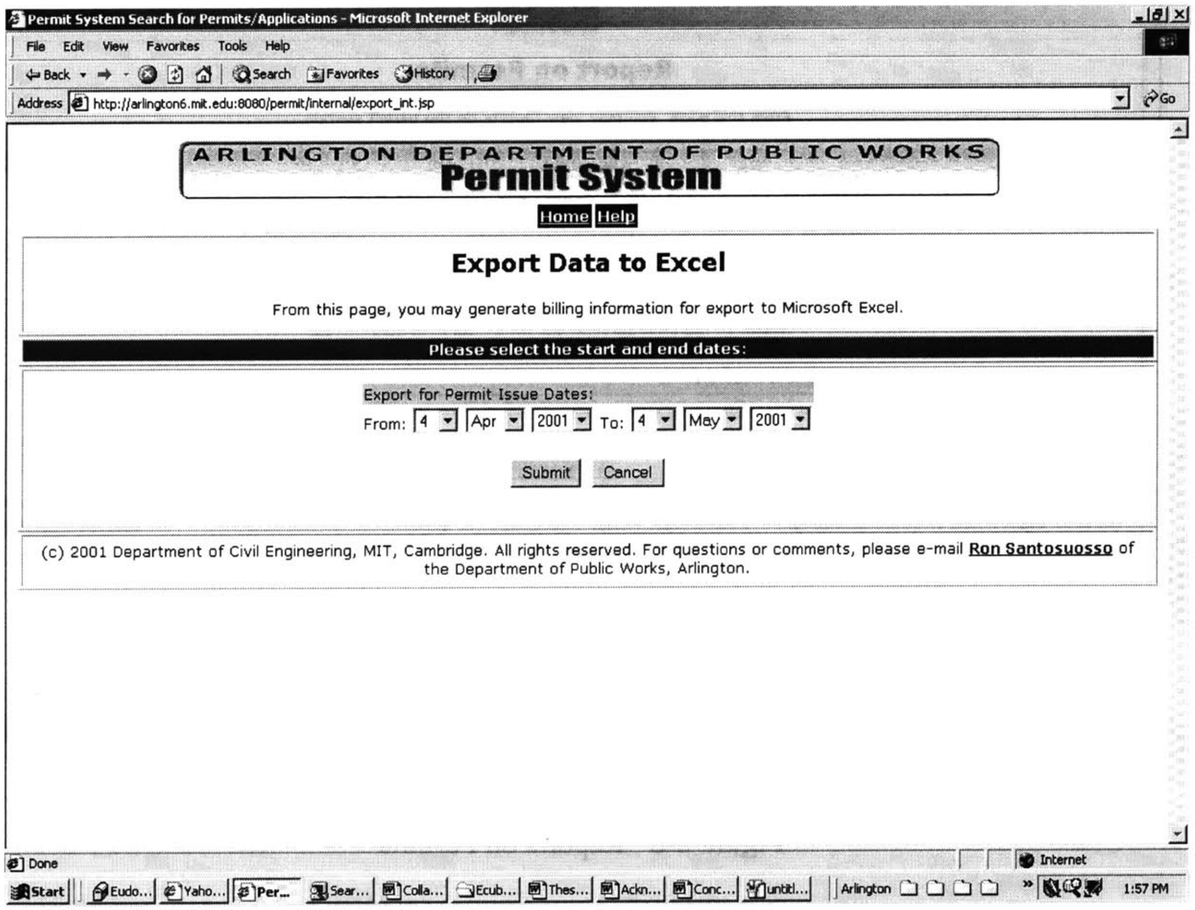
SMTP Server: arlington6.mit.edu

Email Address: andreask@mit.edu

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-6 Miscellaneous Options

There are various non-static miscellaneous parameters that are relevant for the application. The internal user will need to change these parameters as per the existent conditions. The figure above is the page for administering these miscellaneous options.


Permit System Search for Permits/Applications - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/export_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Help

Export Data to Excel

From this page, you may generate billing information for export to Microsoft Excel.

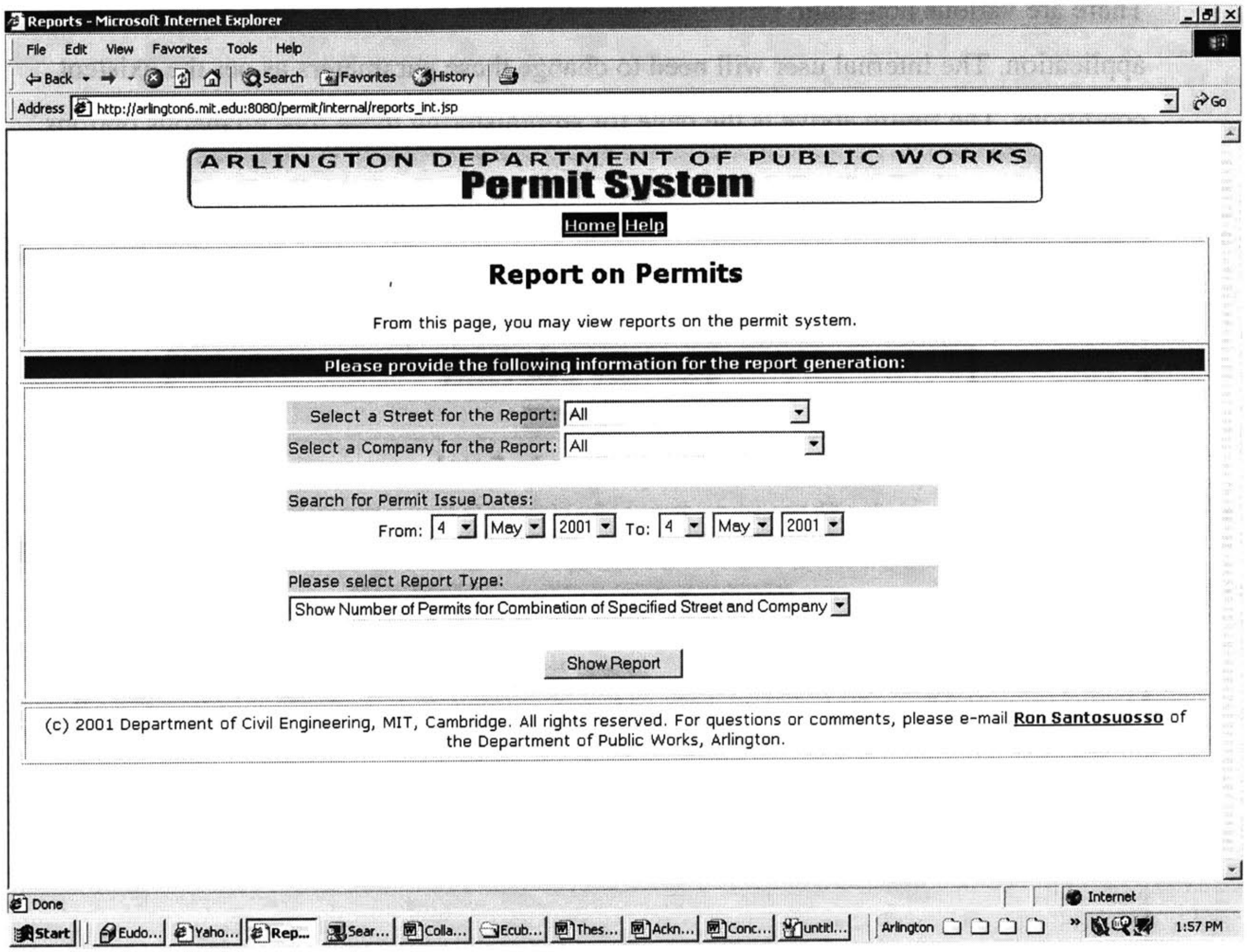
Please select the start and end dates:

Export for Permit Issue Dates:
From: 4 Apr 2001 To: 4 May 2001

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-7 Export Billing data to MS-Excel

The figure above is the page for Exporting billing data to MS Excel. Only the internal user can access from the main page. This page allows for transfer of the billing data for the billing purposes.


Reports - Microsoft Internet Explorer

Address http://arlington6.mit.edu:8080/permit/internal/reports_int.jsp

ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System

Home Help

Report on Permits

From this page, you may view reports on the permit system.

Please provide the following information for the report generation:

Select a Street for the Report: All

Select a Company for the Report: All

Search for Permit Issue Dates:

From: 4 May 2001 To: 4 May 2001

Please select Report Type:

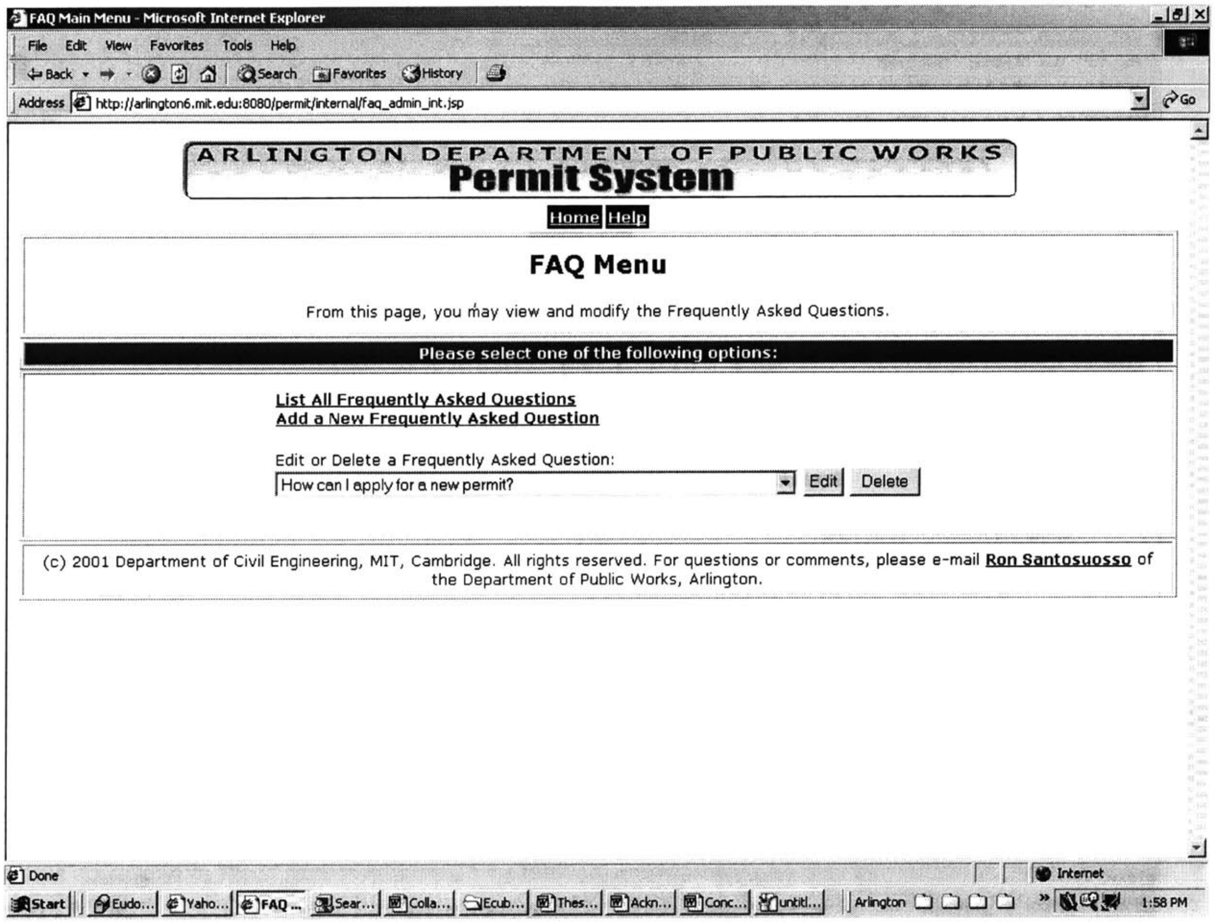
Show Number of Permits for Combination of Specified Street and Company

Show Report

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail **Ron Santosuosso** of the Department of Public Works, Arlington.

Figure 8-8 Reports on Permits

The application also allows the users to generate reports regarding the permits. The user is required to select a combination of factors and the system then generates the reports. The figure above is the Reports page, which the internal user can access from the main page. This page is the main page from which reports can be generated.


FAQ Main Menu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/faq_admin_int.jsp
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Help

FAQ Menu

From this page, you may view and modify the Frequently Asked Questions.

Please select one of the following options:

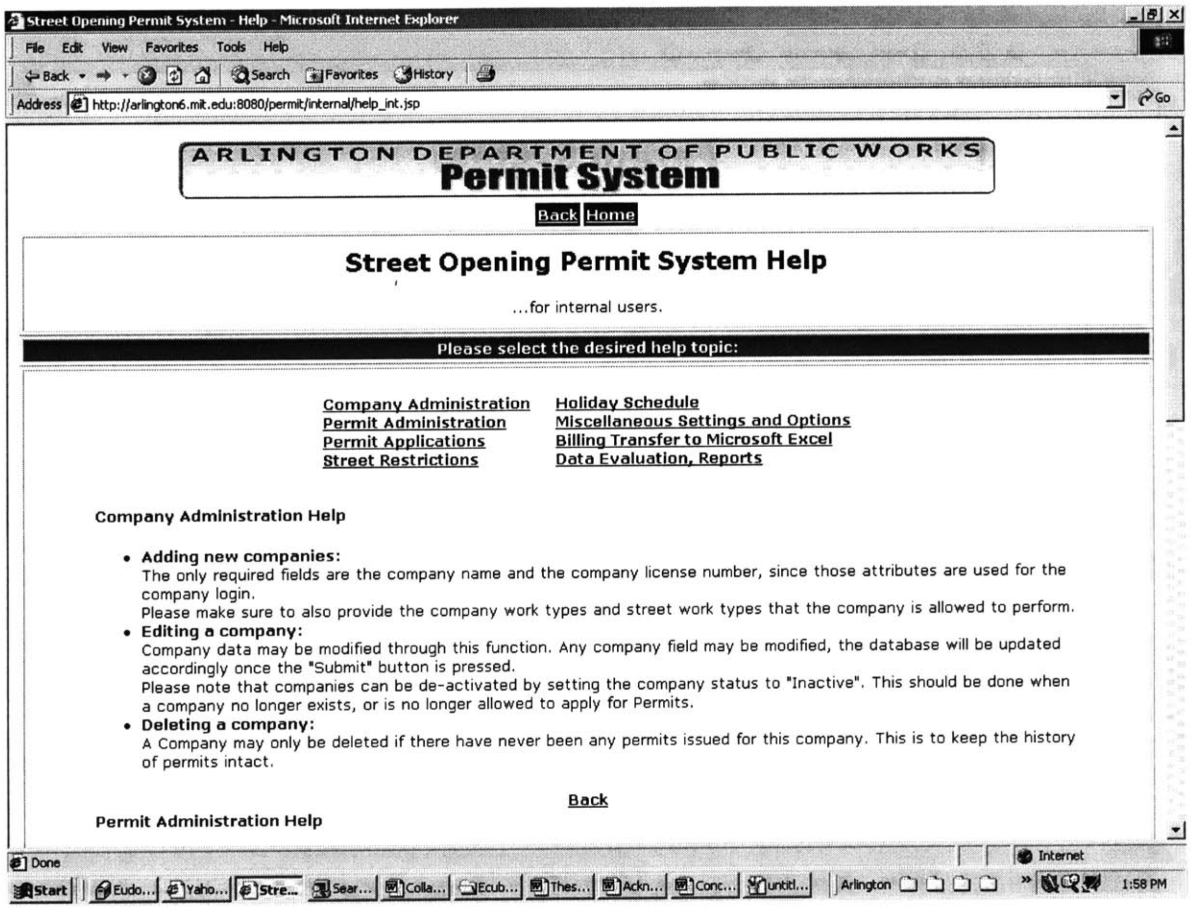
List All Frequently Asked Questions
Add a New Frequently Asked Question

Edit or Delete a Frequently Asked Question:
How can I apply for a new permit?

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-9 FAQ Menu

It was expected that the FAQs would not be static, but would have to be administered from time to time by the department officials based on the queries that they receive regarding the system. The figure above is the FAQs' menu, which also the internal user can access from the main page. This page allows the user to follow links to administer the FAQs.


Street Opening Permit System - Help - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://arlington6.mit.edu:8080/permit/internal/help_int.jsp
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Back Home

Street Opening Permit System Help

...for internal users.

Please select the desired help topic:

Company Administration	Holiday Schedule
Permit Administration	Miscellaneous Settings and Options
Permit Applications	Billing Transfer to Microsoft Excel
Street Restrictions	Data Evaluation, Reports

Company Administration Help

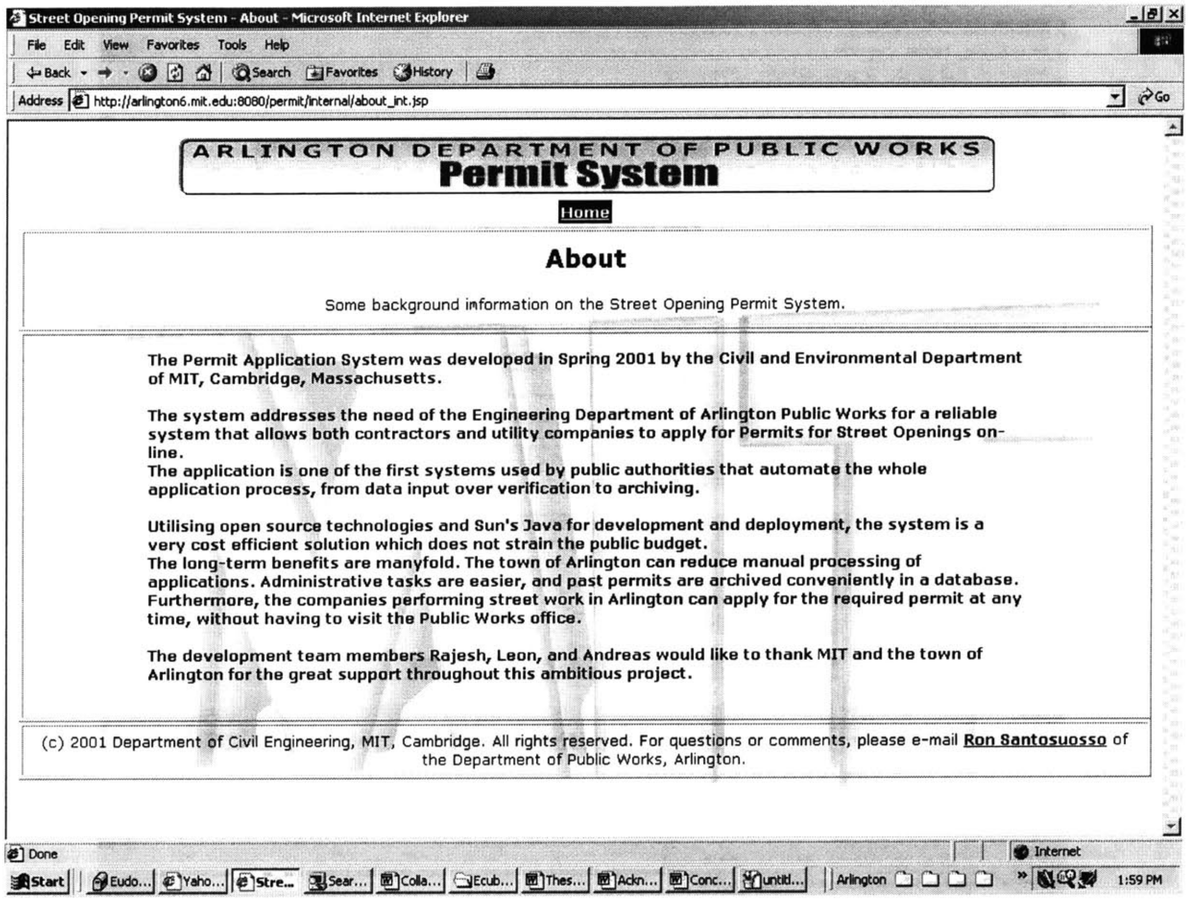
- **Adding new companies:**
  The only required fields are the company name and the company license number, since those attributes are used for the company login.
  Please make sure to also provide the company work types and street work types that the company is allowed to perform.
- **Editing a company:**
  Company data may be modified through this function. Any company field may be modified, the database will be updated accordingly once the "Submit" button is pressed.
  Please note that companies can be de-activated by setting the company status to "Inactive". This should be done when a company no longer exists, or is no longer allowed to apply for Permits.
- **Deleting a company:**
  A Company may only be deleted if there have never been any permits issued for this company. This is to keep the history of permits intact.

Back

Permit Administration Help

Figure 8-10 Street Opening Permit System Help

The page above shows the help page, as it is available to the internal users.


Street Opening Permit System - About - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/about_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home
Done
Internet
Start
1:59 PM

About

Some background information on the Street Opening Permit System.

The Permit Application System was developed in Spring 2001 by the Civil and Environmental Department of MIT, Cambridge, Massachusetts.

The system addresses the need of the Engineering Department of Arlington Public Works for a reliable system that allows both contractors and utility companies to apply for Permits for Street Openings on-line.
The application is one of the first systems used by public authorities that automate the whole application process, from data input over verification to archiving.

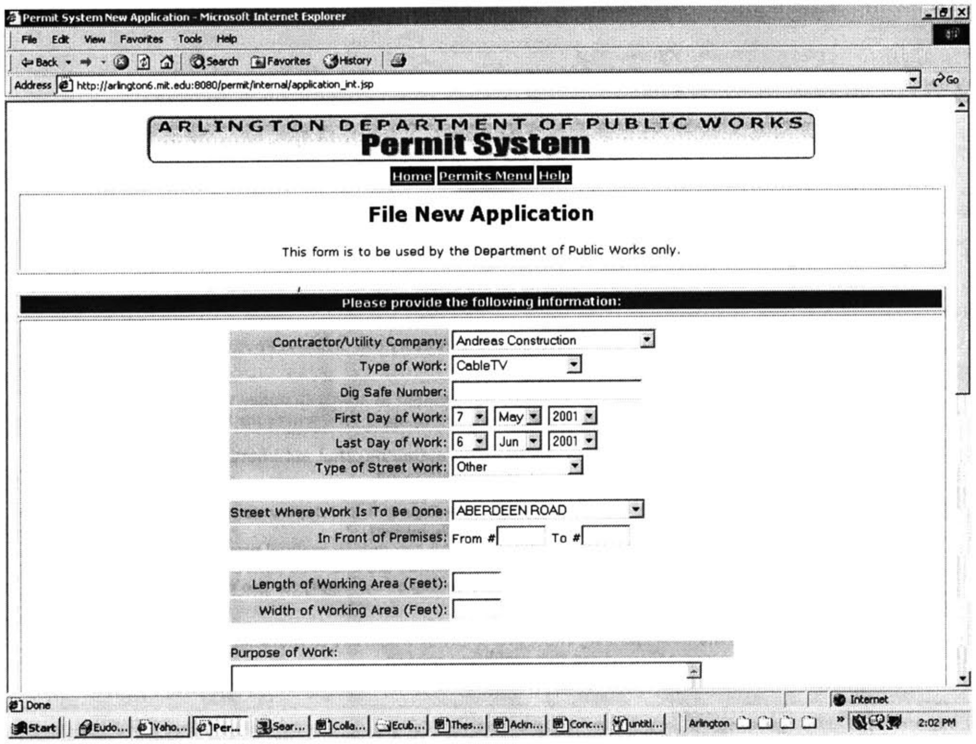
Utilising open source technologies and Sun's Java for development and deployment, the system is a very cost efficient solution which does not strain the public budget.
The long-term benefits are manyfold. The town of Arlington can reduce manual processing of applications. Administrative tasks are easier, and past permits are archived conveniently in a database. Furthermore, the companies performing street work in Arlington can apply for the required permit at any time, without having to visit the Public Works office.

The development team members Rajesh, Leon, and Andreas would like to thank MIT and the town of Arlington for the great support throughout this ambitious project.

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-11 About The Software

The figure above shows the page describing about the software.


Permit System New Application - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/application_int.jsp
Home Permits Menu Help
Done
Internet
2:02 PM

ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System

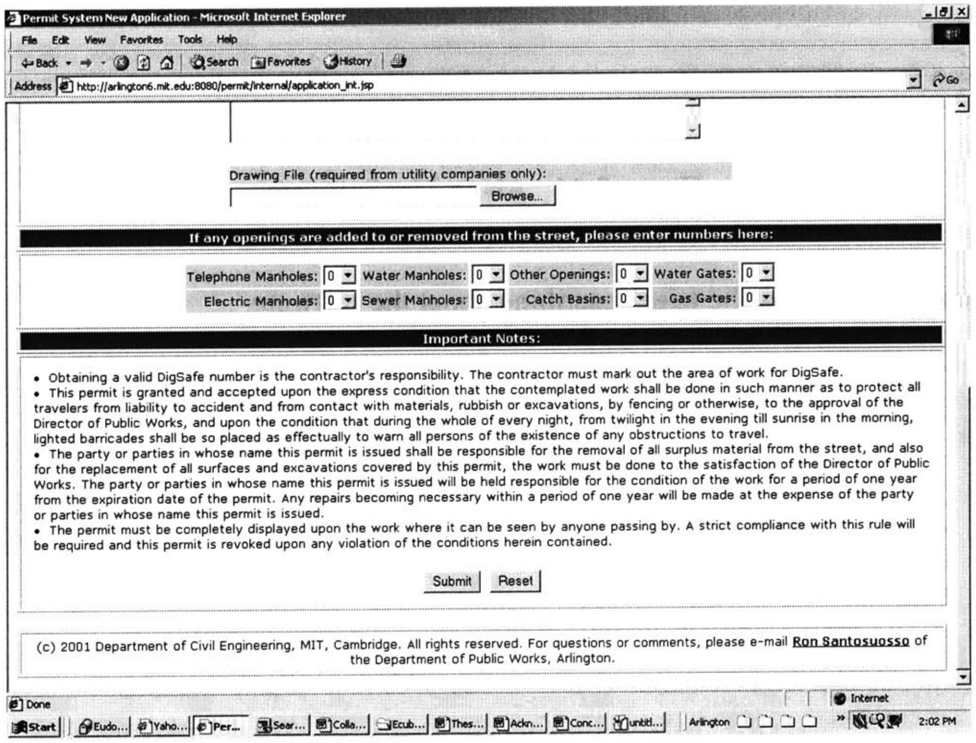
File New Application

This form is to be used by the Department of Public Works only.

Please provide the following information:

Contractor/Utility Company:	Andreas Construction
Type of Work:	CableTV
Dig Safe Number:	
First Day of Work:	7 May 2001
Last Day of Work:	6 Jun 2001
Type of Street Work:	Other
Street Where Work Is To Be Done:	ABERDEEN ROAD
In Front of Premises:	From # To #
Length of Working Area (Feet):	
Width of Working Area (Feet):	

Purpose of Work:


Permit System New Application - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://arlington6.mit.edu:8080/permit/internal/application_int.jsp
Browse...
Done
Internet
2:02 PM

Drawing File (required from utility companies only):

If any openings are added to or removed from the street, please enter numbers here:

Telephone Manholes:	0	Water Manholes:	0	Other Openings:	0	Water Gates:	0
Electric Manholes:	0	Sewer Manholes:	0	Catch Basins:	0	Gas Gates:	0

Important Notes:

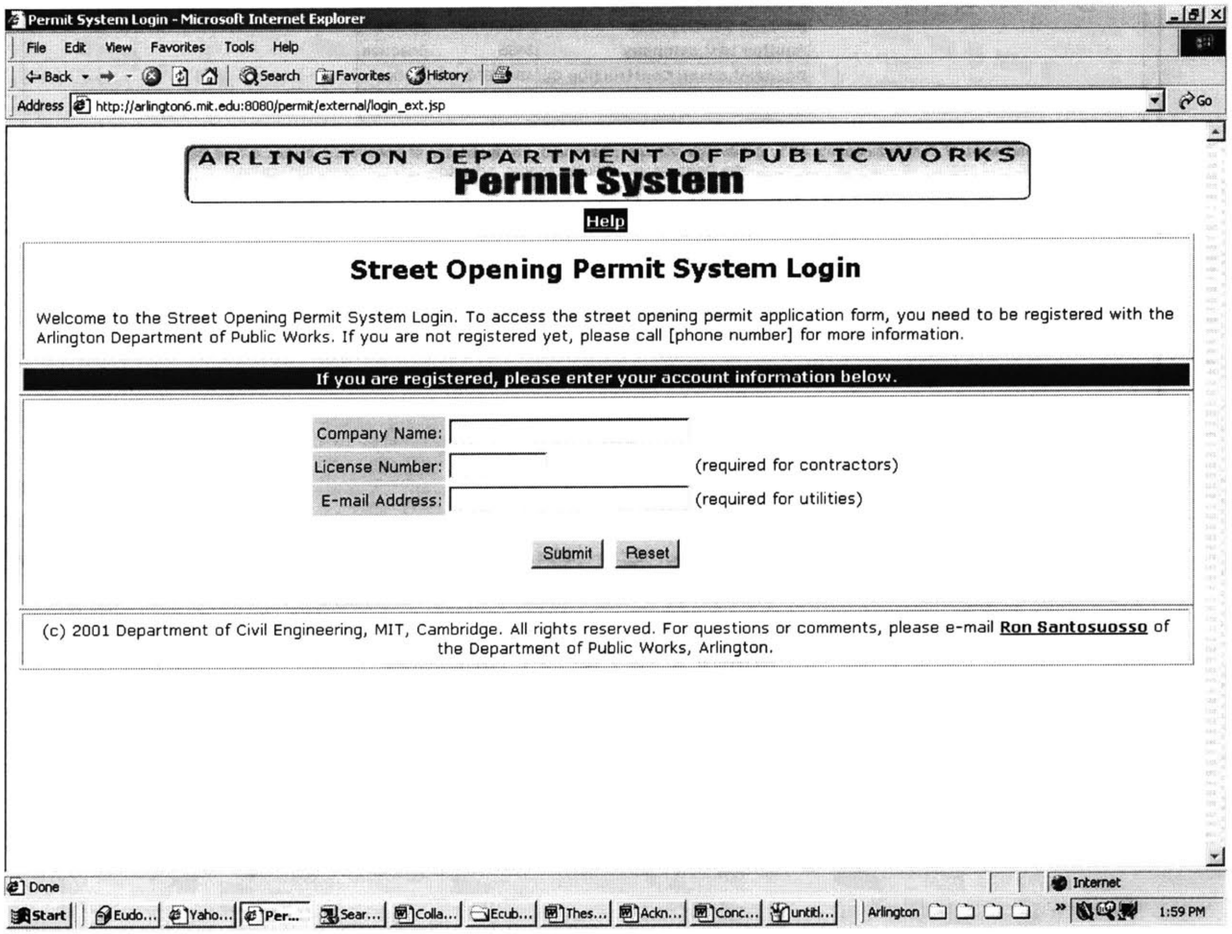
- Obtaining a valid DigSafe number is the contractor's responsibility. The contractor must mark out the area of work for DigSafe.
- This permit is granted and accepted upon the express condition that the contemplated work shall be done in such manner as to protect all travelers from liability to accident and from contact with materials, rubbish or excavations, by fencing or otherwise, to the approval of the Director of Public Works, and upon the condition that during the whole of every night, from twilight in the evening till sunrise in the morning, lighted barricades shall be so placed as effectually to warn all persons of the existence of any obstructions to travel.
- The party or parties in whose name this permit is issued shall be responsible for the removal of all surplus material from the street, and also for the replacement of all surfaces and excavations covered by this permit, the work must be done to the satisfaction of the Director of Public Works. The party or parties in whose name this permit is issued will be held responsible for the condition of the work for a period of one year from the expiration date of the permit. Any repairs becoming necessary within a period of one year will be made at the expense of the party or parties in whose name this permit is issued.
- The permit must be completely displayed upon the work where it can be seen by anyone passing by. A strict compliance with this rule will be required and this permit is revoked upon any violation of the conditions herein contained.

Submit Reset

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-12 Complete Permit Application

The two figures above show the complete application form that the internal user fills out for applying for a permit. A similar page is also available to the external users. The system checks the parameters of the application and then displays the results of the processing of the application.


Permit System Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/external/login_ext.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Help

Street Opening Permit System Login

Welcome to the Street Opening Permit System Login. To access the street opening permit application form, you need to be registered with the Arlington Department of Public Works. If you are not registered yet, please call [phone number] for more information.

If you are registered, please enter your account information below.

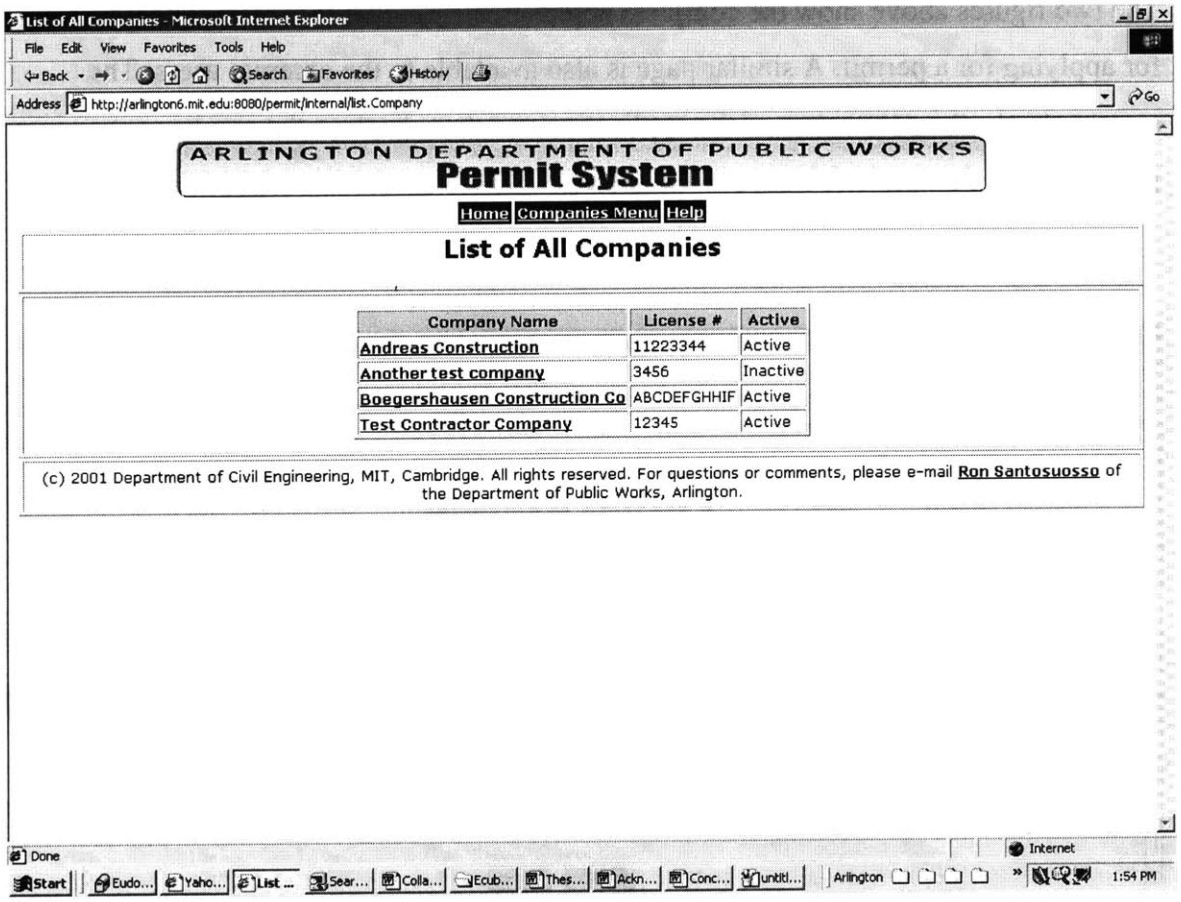
Company Name:		
License Number:		(required for contractors)
E-mail Address:		(required for utilities)

Submit Reset

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail **Ron Santosuosso** of the Department of Public Works, Arlington.

Figure 8-13 Login Screen for External Users

The figure above shows the Login Screen for the External users. The internal users also will be presented with a similar login screen.


List of All Companies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://arlington6.mit.edu:8080/permit/internal/list.Company
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home
Companies Menu
Help

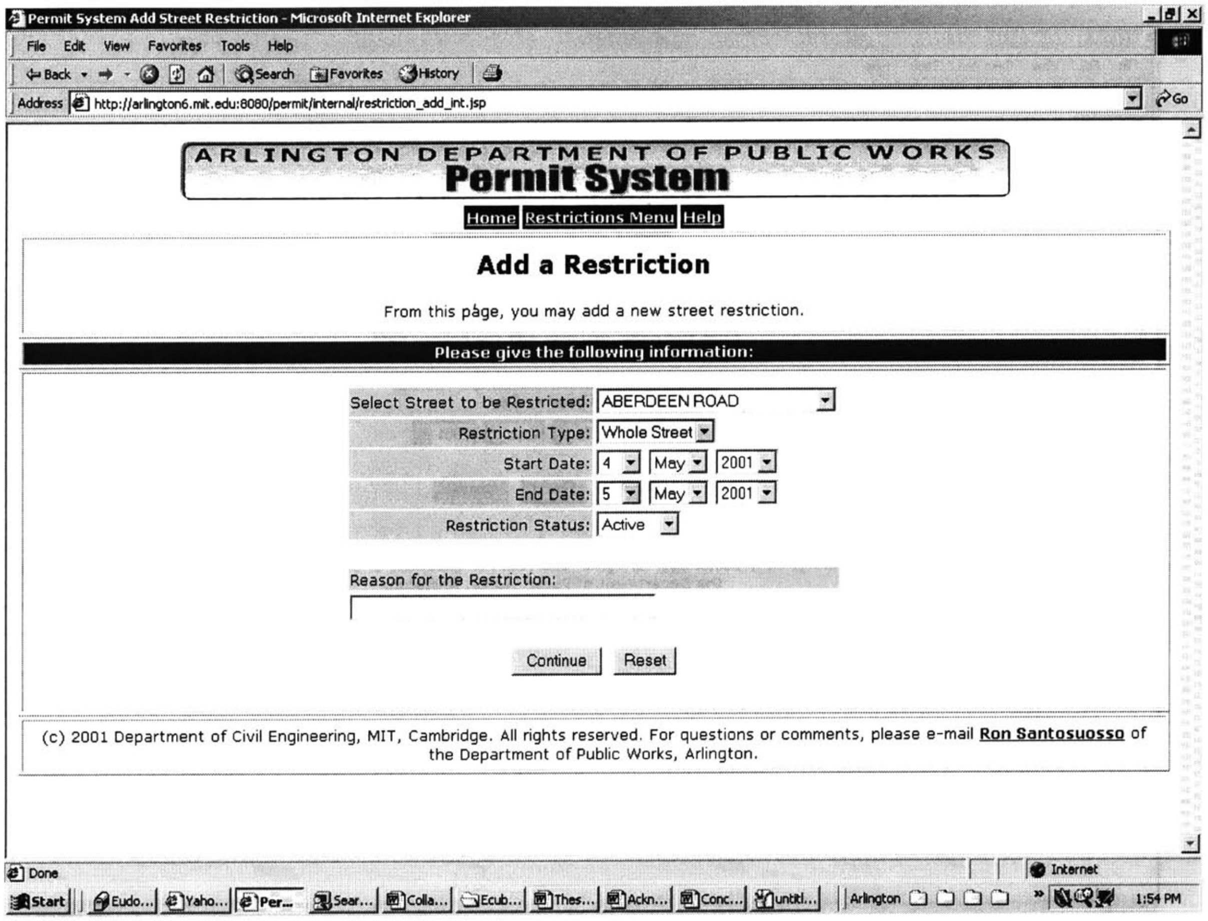
List of All Companies

Company Name	License #	Active
Andreas Construction	11223344	Active
Another test company	3456	Inactive
Boegershausen Construction Co	ABCDEFGHHIF	Active
Test Contractor Company	12345	Active

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-14 List of Companies

The figure above shows one of the common functionalities of the application, i.e. listing of items. Here we show the listing of the companies for this purpose.


Permit System Add Street Restriction - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/restriction_add_int.jsp
Go
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Restrictions Menu Help
Continue
Reset
Done
Internet
Start
1:54 PM

Add a Restriction

From this page, you may add a new street restriction.

Please give the following information:

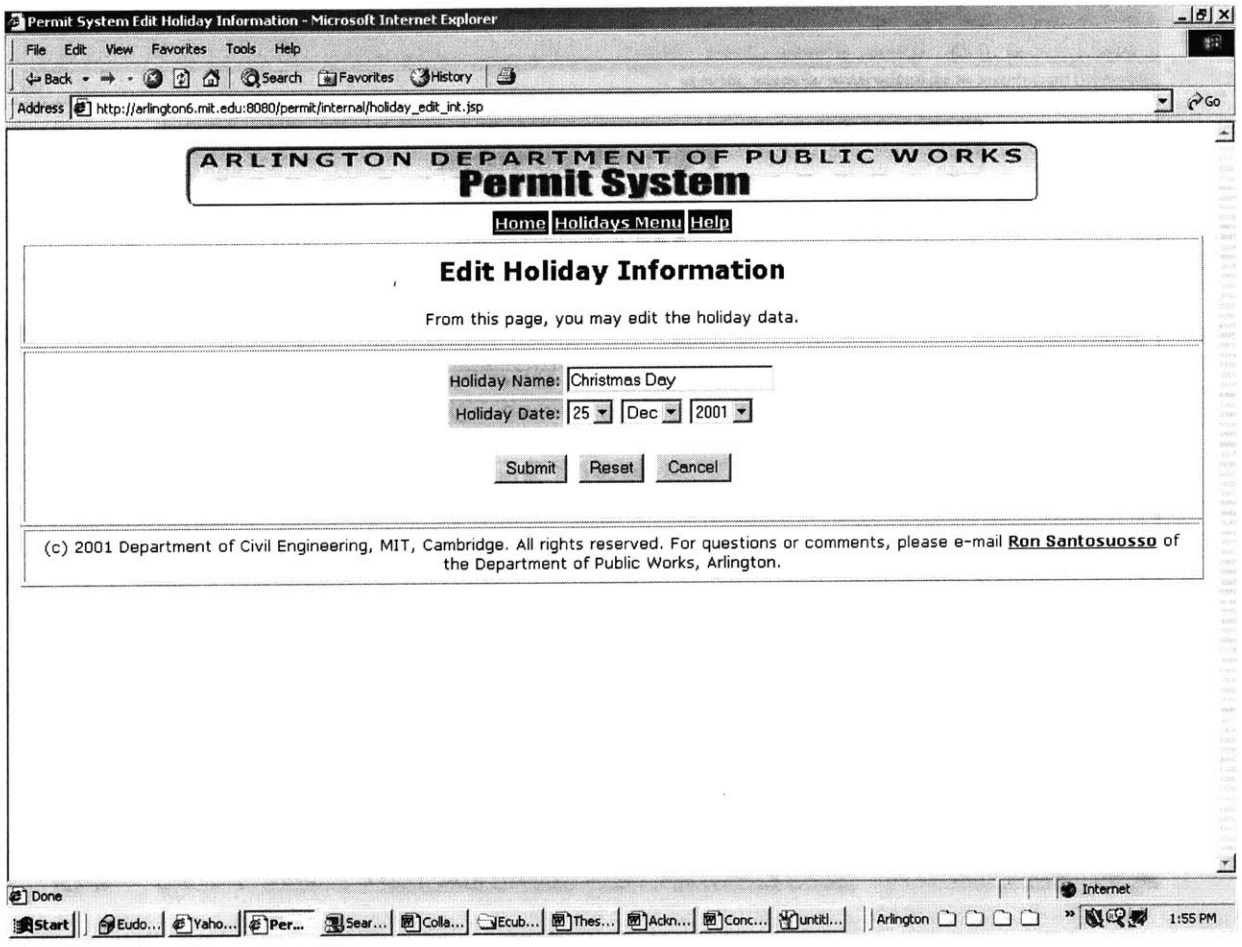
Select Street to be Restricted:	ABERDEEN ROAD
Restriction Type:	Whole Street
Start Date:	4 May 2001
End Date:	5 May 2001
Restriction Status:	Active

Reason for the Restriction:

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail **Ron Santosuosso** of the Department of Public Works, Arlington.

Figure 8-15 Add a Restriction

Similarly the above figure shows the other common activity of adding items to the database for the administration of the application. As an example we show the screen for the addition of a Restriction.


Permit System Edit Holiday Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://arlington6.mit.edu:8080/permit/internal/holiday_edit_int.jsp
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home
Holidays Menu
Help

Edit Holiday Information

From this page, you may edit the holiday data.

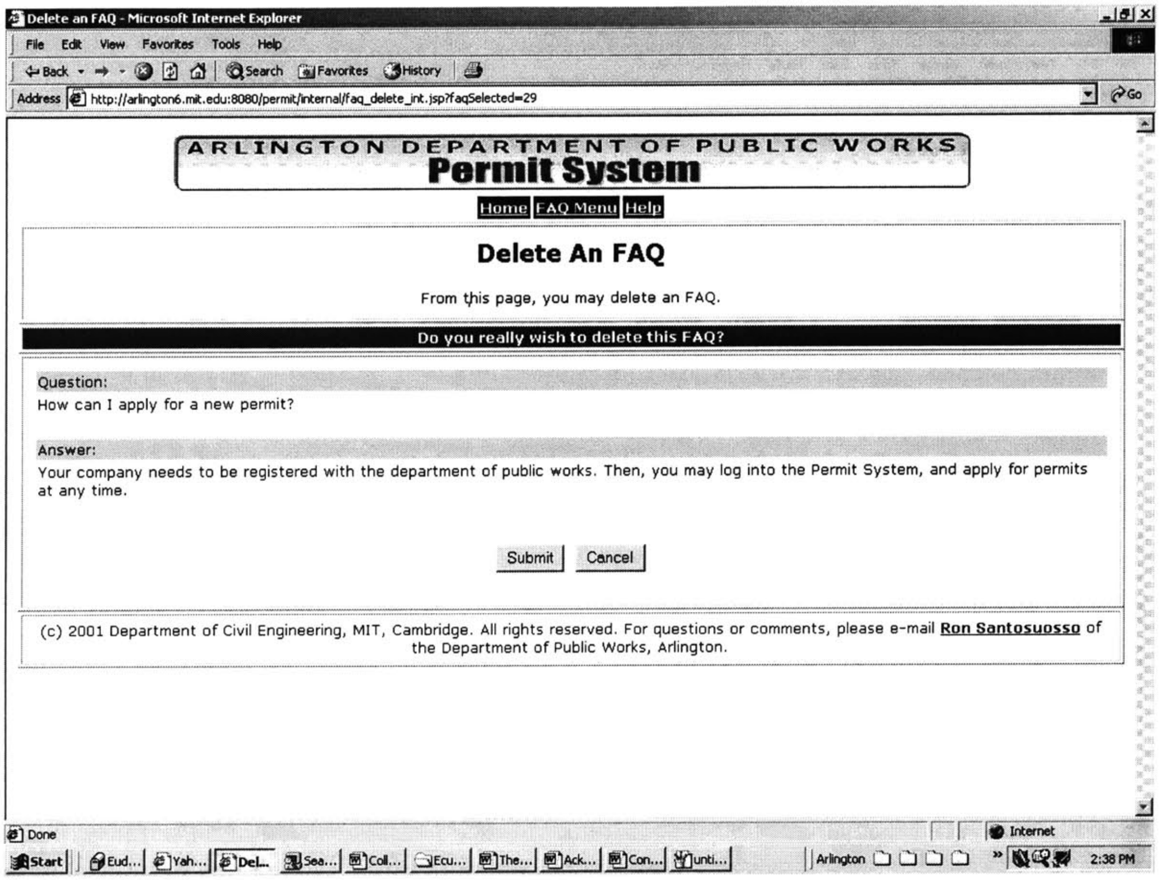
Holiday Name:	Christmas Day
Holiday Date:	25 Dec 2001

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Done
Internet
Start
1:55 PM

Figure 8-16 Edit a Holiday Information

The figure above shows a representative screen for the activity of editing with editing holiday information as an example.


Delete an FAQ - Microsoft Internet Explorer

http://arlington6.mit.edu:8080/permit/internal/faq_delete_int.jsp?faqSelected=29

ARLINGTON DEPARTMENT OF PUBLIC WORKS

Permit System

Home FAQ Menu Help

Delete An FAQ

From this page, you may delete an FAQ.

Do you really wish to delete this FAQ?

Question:
How can I apply for a new permit?

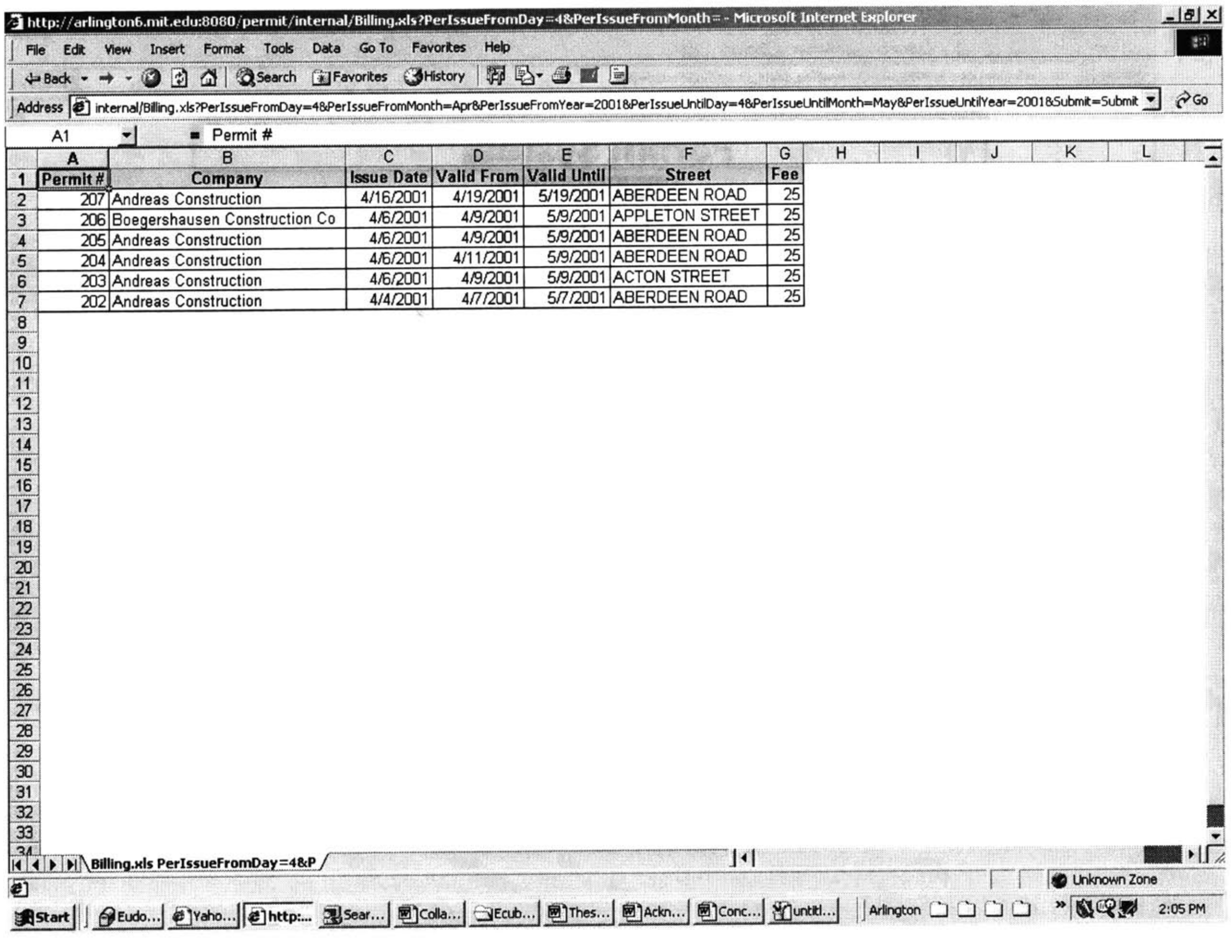
Answer:
Your company needs to be registered with the department of public works. Then, you may log into the Permit System, and apply for permits at any time.

Submit Cancel

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail **Ron Santosuosso** of the Department of Public Works, Arlington.

Figure 8-17 Delete the FAQ

Similarly the figure shows the other commonly used administration activity of deleting an item with the deleting an FAQ as an example here.



http://arlington6.mit.edu:8080/permit/internal/Billing.xls?PerIssueFromDay=4&PerIssueFromMonth= - Microsoft Internet Explorer

Address: internal/Billing.xls?PerIssueFromDay=4&PerIssueFromMonth=Apr&PerIssueFromYear=2001&PerIssueUntilDay=4&PerIssueUntilMonth=May&PerIssueUntilYear=2001&Submit=Submit

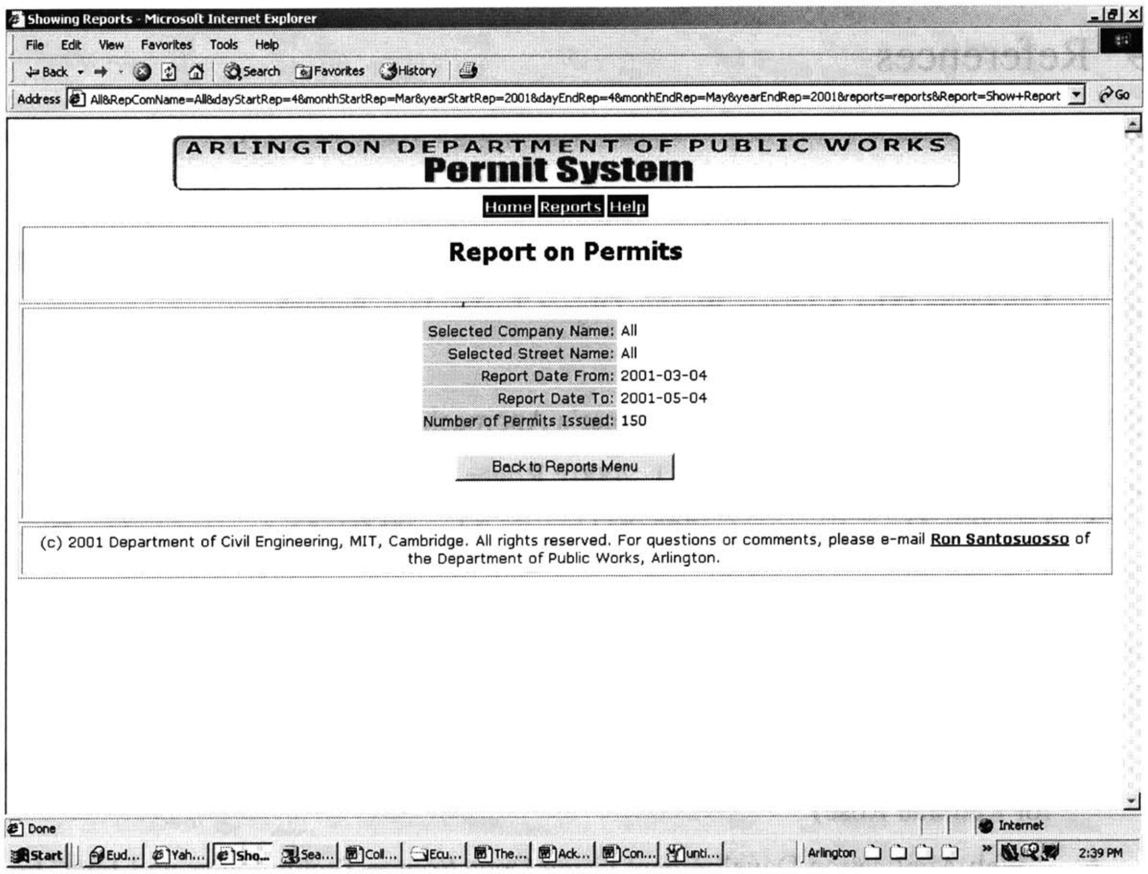
A1 = Permit #

Permit #	Company	Issue Date	Valid From	Valid Until	Street	Fee
207	Andreas Construction	4/16/2001	4/19/2001	5/19/2001	ABERDEEN ROAD	25
206	Boegershausen Construction Co	4/6/2001	4/9/2001	5/9/2001	APPLETON STREET	25
205	Andreas Construction	4/6/2001	4/9/2001	5/9/2001	ABERDEEN ROAD	25
204	Andreas Construction	4/6/2001	4/11/2001	5/9/2001	ABERDEEN ROAD	25
203	Andreas Construction	4/6/2001	4/9/2001	5/9/2001	ACTON STREET	25
202	Andreas Construction	4/4/2001	4/7/2001	5/7/2001	ABERDEEN ROAD	25

File Edit View Insert Format Tools Data Go To Favorites Help
Back Search Favorites History
Go
Billing.xls PerIssueFromDay=4&P
Unknown Zone
Start Eudo... Yaho... http:... Sear... Colla... Ecub... Thes... Ackn... Conc... untitl...
Arlington
2:05 PM

Figure 8-18 Billing Data

The figure above shows the billing data, as it would be available on an MS Excel sheet (on the browser window).


Showing Reports - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address All&RepComName=All&dayStartRep=4&monthStartRep=Mar&yearStartRep=2001&dayEndRep=4&monthEndRep=May&yearEndRep=2001&reports=reports&Report=Show+Report
ARLINGTON DEPARTMENT OF PUBLIC WORKS
Permit System
Home Reports Help
Back to Reports Menu
Done
Internet
2:39 PM

Report on Permits

Selected Company Name:	All
Selected Street Name:	All
Report Date From:	2001-03-04
Report Date To:	2001-05-04
Number of Permits Issued:	150

(c) 2001 Department of Civil Engineering, MIT, Cambridge. All rights reserved. For questions or comments, please e-mail Ron Santosuosso of the Department of Public Works, Arlington.

Figure 8-19 Reports on Permits

The screenshot above shows the report being displayed to the internal user.

9 References

1. UML Distilled, Second Edition: A Brief Guide to the Standard Object Modeling Language (The Addison-Wesley Object Technology Series)
   by Martin Fowler, Kendall Scott
2. Software Engineering A Practitioner's Approach (McGraw-Hill Higher Education), 5th edition (June 1, 2000): Roger S Pressman
3. http://stud4.tuwien.ac.at/~e8726711/ummw3.html#MU15A (aggregated by C. Demmer using material from G. Booch, J. Rumbaugh and I. Jacobson)
4. http://www.rational.com/products/whitepapers/285.jsp (Quality Software and the Unified Modeling Language *by Grady Booch)*
5. http://www.rational.com/uml/resources/quick/index.jsp (UML™ Quick Reference for Rational Rose)
6. "Web Application Development Using Open Source Technologies and Java" by Wolfgang Andreas Klimke

[illegible]