Evolution of Platform and User Interface in Infrastructure Management System with Case Study of Arlington Pavement Management System

BY
YATLUN CHOI

BACHELOR OF SCIENCE IN CIVIL ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2001

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:
DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
MAY 11, 2001

CERTIFIED BY:
GEORGE KOCUR
SENIOR LECTURER, CIVIL AND ENVIRONMENTAL ENGINEERING
THESIS SUPERVISOR

ACCEPTED BY:
ORAL BUYUKOZTURK
CHAIRMAN, DEPARTMENTAL COMMITTEE ON GRADUATE STUDIES

Evolution of Platform and User Interface in Infrastructure Management System with Case Study of Arlington Pavement Management System

BY

YATLUN CHOI

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING ON
MAY 11, 2001
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING

ABSTRACT

Infrastructure management systems have been used in public agencies for many years to manage a wide variety of public facilities, of which pavement management systems are one of the most commonly deployed. A well-designed pavement management system facilitates the efficient strategic planning of pavement maintenance and lowers the cost required in the long term.

There are many approaches to the formulation of pavement management processes and design of pavement management systems. Advancement in computing technologies has presented the Internet as a new choice of platform for pavement management systems. This thesis aims to (1) provide the reader with an introduction to the pavement management process and the algorithms involved in strategy selection, (2) examine the components behind the evolution and development of Web technologies and application user interfaces, and (3) study the implementation of Web technology on pavement management systems, using the Arlington PMIS as a case study.

THESIS SUPERVISOR: GEORGE KOCUR
TITLE: SENIOR LECTURER, CIVIL AND ENVIRONMENTAL ENGINEERING

Acknowledgements

I would like to express my gratitude to the following people for this thesis:

- Dr. George Kocur for his guidance and assistance on the thesis as well as on the M.Eng. Project throughout the year;

- Mr. Ron Santosuosso of Department of Public Works, Arlington, MA for his help on the M.Eng. Project, which is used as a case study in the thesis;

- my M.Eng. Project partners William Cheung, Warit Durongdej, and Wai Kei Yim for turning what could have been the worst nightmare into an experience filled with fun, and the class of 2001 M.Eng. students for making this an enjoyable year;

- my parents and my sister, for their love and encouragement, especially through these five years when I am away from home;

- all my friends in MIT, for helping me survive the three years of tough life here, especially to Daniel Kwok, now an Assistant Professor at Univesity of Alberta, who has been very supportive and truthful to me at all times and turned me into a much more cheerful person, and Kenneth Yu, who as my roommate in Junior year had given me tremendous support in adjusting my life from UCLA to MIT.

Table of Content

List of Figures

List of Tables

1 Introduction

Millions or even billions of people commute by cars every day, and although often overlooked, pavements in good conditions are essential in providing safe and enjoyable riding experience as well as efficient traffic flow. Pavements need to be constantly inspected and maintained so that any deteriorated pavements can be restored to acceptable standard; this is especially true for pavements in areas with extreme weather conditions, where deterioration tends to take place at a higher rate. Pavement Management Systems provide a means for public agencies to efficiently monitor pavement conditions and formulate maintenance strategies.

This thesis investigates the development of Pavement Management Systems as a Web application and reasons for the trend to develop toward a Web interface.

Chapter 2 discusses the pavement management process and algorithms in pavement management strategy selection. Chapter 3 investigates what Web applications are, its architecture, the various Web application user interface models, and the advantages and disadvantages of developing application on the Web interface. Chapter 4 covers the common technologies deployed in development of Web applications. Chapter 5 studies different implementations of platform and technology in pavement management systems. Chapter 6 gives a summary of the study and also looks into possible future development of pavement management systems.

2 Pavement Management Systems

2.1 Background

Maintenance of pavement systems can provide several benefits to both the public authorities and commuters over long term. In the past, pavements were maintained but not well managed. Experience of engineers used to dictate the maintenance decisions of pavement, and there lacked a systematic approach to managing pavements. The process lacked life cycle costing or prioritization given to streets that reap a higher cost-benefit ratio. To maximize benefits and minimize costs, pavement networks need to be managed in addition to being maintained.

Developments in computing technology as well as pavement management technology have allowed the pavement management process to be carried out in a more economical manner. Maintenance and rehabilitation (M & R) of pavements can be planned using pavement management systems (PMSs) that use systematic methods for determining needs and priorities. As illustrated in Figure 2-1, if M & R is performed during early stages of deterioration before any sharp decline in pavement condition, more than 70% of rehabilitation cost can be saved.

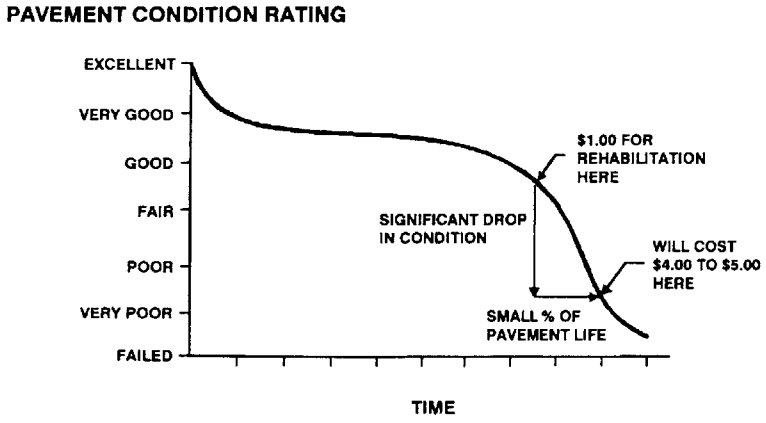


Figure 2-1 Conceptual illustration of a pavement condition life cycle [1]

[1] M.Y. Shahin. *Pavement Management for Airports, Roads, and Parking Lots*. Figure 1-1 (p.2).

Apart from cost, PMSs can also help reduce the time needed for maintenance, as closure of pavement for maintenance can be shortened with better street traffic management.

2.2 Pavement Management Algorithms

There are many different algorithms in selecting M & R alternatives. Some of the most common ones are described below:

Ad hoc method: A number of sections that have reached a significant level of deterioration are grouped into a pool of prospective maintenance candidate. Pavement sections are then selected randomly from within the pool until each yearly budget is exhausted. Projects that are not funded in one year would be postponed to the subsequent years when funding becomes available.

Ranking method: Pavement sections within the pavement network are ranked using some index of pavement condition and selections are made based on a worst-first case each year until the yearly budget is exhausted. Similar to the ad hoc method, projects not funded in one year would be postponed.

Incremental Benefit Cost (IBC) method: The yearly benefit is optimized by selecting the project based on a ranking of pavement sections from high IBC to low IBC. Pavement sections with higher IBC are given priority in maintenance plans.

Linear Programming Optimization method: Linear programming optimization is used to determine the priorities of different management plans. It also aims at optimizing the yearly benefit within a yearly budget limit.

From research done by Mohseni, Darter, and Hall examining the effects of the four different algorithms using ILLINET pavement management program developed for the Illinois Department of Transportation, the following results were obtained:

Network parameter	**Ad hoc**	**Ranking**	**Incremental benefit-cost**	**Linear programming optimization**
Annual budget (million $)	7.5	7.5	7.5	7.5
Percent VMT on deteriorated pavements	15.0	3.5	6.1	6.2
VMT on good pavements (billion miles)	2.98	3.82	5.64	5.63
VMT Benefit/ Total Cost	40	52	77	**79**

(VMT: vehicle miles traveled)

Table 2-1 Summary of 10 year costs and benefits for case study highway network using ILLINET [2]

Results show that the VMT on deteriorated roads dropped significantly from 15% to 3.5% - 6.2% when more systematic approaches such as ranking, IBC, or linear programming, are adopted in place of ad hoc methods. An ideal pavement management system should therefore not only provide the database and interface for maintaining

[2] Mohseni, Darter, Hall. *Benefits from Improved Management of Pavement Facilities*; Infrastructure Planning and Management, p.24.

pavements, but also implement a suitable algorithm for prioritizing pavement maintenance decisions.

2.3 Pavement Management Process

Figure 2-2 shows the pavement management process:

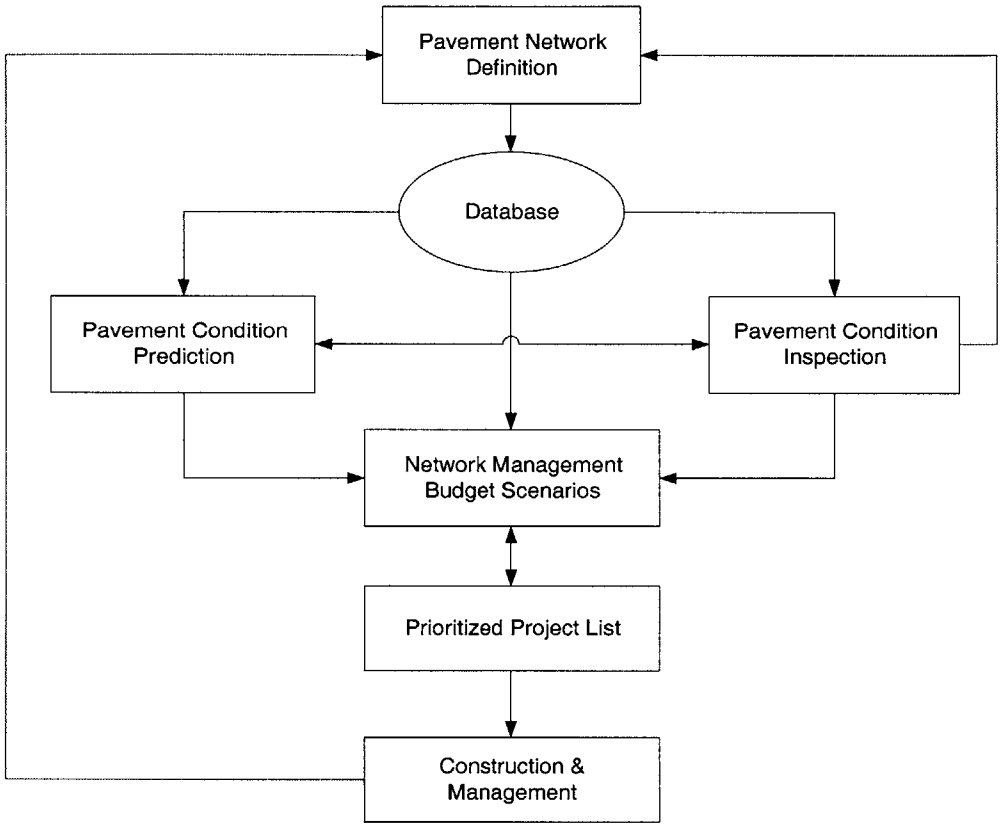


Figure 2-2 The Pavement Management Process

The first task of the pavement management process is the network definition, which involves defining the set of pavement network(s) to be managed, such as roads, parking lots, airfields and other types of facilities. Network definition would define the scope of facilities to be considered as a separate network. Each separate network would be entered into a single database in a PMS.

After building each network a PMS database, pavement condition measurement has to be carried out to obtain data for pavement condition prediction. Depending on the

criteria for the determination of the pavement serviceability index (PSI), where higher PSI indicates better pavement condition, different conditions of the streets are inspected. The degree of pavement deterioration is a function of defect type, defect severity, and amount or density of the defect. Inspection data are entered into the relevant database and PSIs are calculated to quantify the pavement conditions.

PSI alone would not be enough for managers to determine maintenance strategies, because some pavements with high PSIs may deteriorate faster than others in the future, and so priority should not be given simply based on the PSI level. Therefore a pavement management process should also include pavement condition prediction. Different PMS would incorporate different techniques for developing prediction models to determine future PSI, such as straight-line extrapolation, regression, or other algorithms.

From the quantified models of pavement conditions for the network, scenarios for different maintenance plans are created and compared, based on criteria discussed in the previous section (such as ad hoc, or ranking). A list of pavements that carry the highest priorities for maintenance would then be compiled, and managers would make decisions on construction and rehabilitation based on the budget limits. Upgrading of pavement conditions resulting from M & R would then change the data in the database.

2.4 Computerized Pavement Management Systems

Pavement management systems have evolved from paper systems to information systems that make use of the power of computers. With the advancement of technology, pavement management systems are now getting increasingly network-based, as opposed to packaged software that runs on a single computer. The following section investigates this trend.

3 Web Applications

3.1 What Web Applications Are

Web applications have evolved from the exponential growth in adoption of the Internet. Instead of executing code within desktop applications, Web applications are executed in browsers. Web pages were first used to simply display static content and let users navigate through that content. The ability to display static content for users to navigate and exchange information remains one of the most important functions of Web pages, but Web pages with the ability to display dynamic content, which in fact are Web applications, have become a major propellant in the rapid penetration of the Internet and a fuel for development of a new Internet-dependent economy.

Web applications can be broadly defined as Web pages that serve dynamic content, which can range from document pages with simple keyword search forms, to complex systems involving database transactions and interactions between users and servers. Web applications vary in size, and can be deployed on the Internet, as well as on corporate intranets and extranets. Web applications leverage Web servers, Web clients, standard Internet protocols, and also existing applications and data from external non-Web services.

Web clients involved in Web applications span a diverse range of products from information appliances such as cellular phones and personal digital assistants (PDAs), to network computers and personal computers. While the capabilities of the Web clients vary significantly, they are unified with the Web application server by their reliance upon a set of Web-based technologies and protocols, such as Java, TCP/IP, HTTP, HTTPS, HTML, etc.

A typical Web interaction begins when a user fills out an HTML based form and hits the submit button. This produces an HTTP request that is sent to the Web server and the

HTTP request names the Web application element to be used to process the request with a Universal Resource Identifier (URI). The HTTP request also carries information about the client and the data that the user has entered into the form. The Web server sends the request to the appropriate application element, which performs operations based on the information in the request, often leveraging a database or external applications. The application element then constructs a response, most often in the form of an HTML document, which the Web server sends back to the Web browser where it is displayed to the user. The interaction is completed.

Web applications are generally useful only if they can serve dynamic content, and typically this is achieved through interactions with databases. The Web server obtains input from the users and generates feedback through executing codes and queries and using the inputs as parameters. There are different ways how code can be generated. For example, a Web server can pass requests to an external program, which generates output to be sent back to client as a static file. Alternatively, a Web server can handle requests by separate threads within the Web server process, instead of having multiple processes to handle separate requests; the server then serves the output content through the network infrastructure, and users would be able to view the output and interact with the Web applications through browsers sitting on their computers.

3.2 Web Architecture

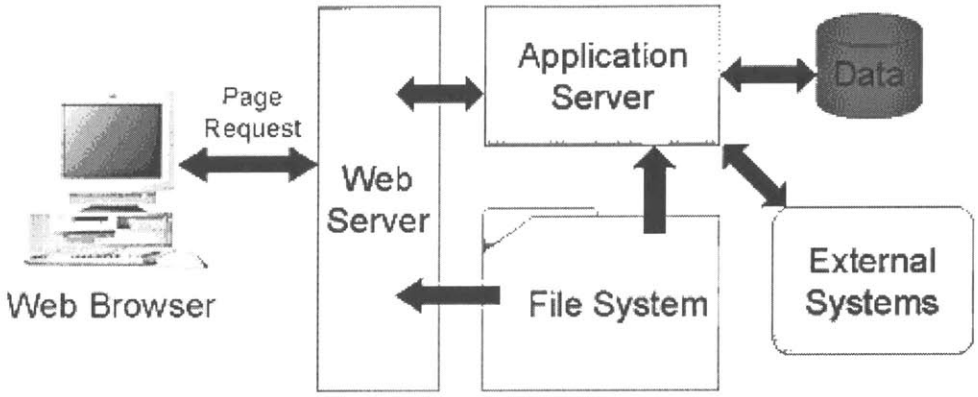


Figure 3-1 Representation of a canonical Web architecture

Figure 3-1 shows the representation of a canonical Web architecture. The elements on the right — the file system, the application server, data, and external systems — are essentially the same as found in traditional client and server systems. The elements on the left — the browser, the Web server, and again the file system (in this case, a distributed one) — are elements unique to the Web space.

From the perspective of the user experience, this otherwise physically distributed back-end looks like traditional mainframe computing. However, there are significant architectural differences between the two kinds of systems owing to the differences in mechanisms that tie these elements together. For example, between the browser and the Web server, communication is generally stateless, involving the request for, and then the delivery of, a Web page. To retain information of the user in subsequent transactions through the Web, a Web application needs to preserve the user's session state. There are a number of alternatives to achieve this, of which cookies and URL rewriting are the most common.

The placement of the application's business logic represents another architectural challenge: it could live in the server; it could live in the client; or it could be spread out overall. There are three layers of processing that involve a typical Web application: the

presentation layer (the user interface), the application layer (handles application-specific processing), and the data management layer (which deals with the actual storage of data.) Thin Client systems are systems in which the client only implements the presentation layer. Fat client systems, on the other hand, have the client implement the presentation and application layers so that there is also local processing on the client side. In the spectrum of thin to fat client, each alternative has its own advantages and disadvantages.

Certain Web applications belong to the Fat Client systems where some of the processing is at the client side. It introduces better server scalability and also less network traffic. However, in such systems, the client is more complex, and it is difficult to port to different platforms. Also, changes in server architecture are more likely to require changes in client.

In thin client systems, it is easy to port the client to different architectures. The client is decoupled from changes in the application so that an application upgrade is transparent to the client. However, the server is responsible for all the operations, and it may easily get saturated. The reliance on the server for all data processing also potentially lengthens the network delays, adversely affecting the performance of the application. In Web application development, changes are rapid and applications need to be constantly updated; therefore most systems today tend to place business logic to the server.

Connection to the application's persistent data, which may be bound in legacy systems, also involves many architectural challenges. First, the system tries to give the illusion of objects to the user while data continues to live in relational tables. Second, the developer has to consider how the connection from the system's business logic to its data should be manifested. For example, a coupling via JDBC (Java Database Connectivity) is more direct but requires that the application developer have intimate knowledge of the form of the

data. Alternatively, a messaging architecture is less direct but is used when direct interfaces are not possible or volumes are low and performance requirements are not high.

3.3 User Interfaces

There are two options for choosing a model for the user interface (UI) of a Web application. From the client's point of view, the user interface can be driven from either the client or the Web application server. This leads to the two programming models: Server Driven UI and Client Driven UI, and the choice of model to use depends on a number of factors including Web technology available or expected to be available on the client, client and server capacity, and network bandwidth.

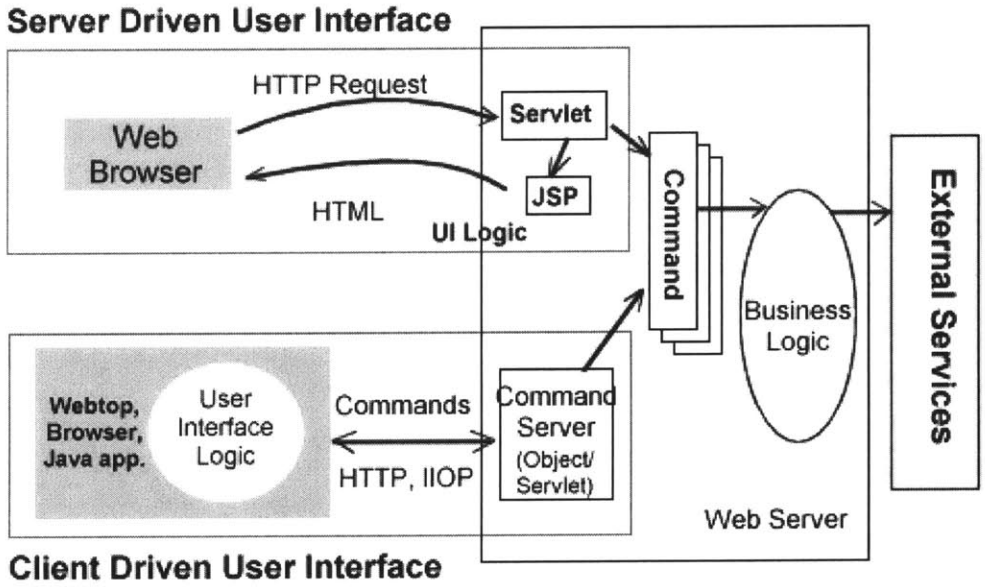


Figure 3-2 User Interface Models

3.3.1 Server Driven UI

The Server Driven UI model is more frequently used in Web applications. In this model, the UI to be displayed on the client is generated by the Web application that executes on the Web server. The Web server responds to every request from the client with an entirely new HTML page. The new page can be a static or dynamically generated HTML page, or it could contain Java applets, JavaScript, or Dynamic HTML (DHTML).

The Web server specifies the format that the data is to be displayed on the client. When new information is to be displayed on the client, the Web server generates a new HTML page and sends it to the client.

The Server Driven UI model has several advantages that make it the most common model used:

- Web applications are easier to install and maintain because the Web server determines what should be displayed.
- Web applications based upon HTTP and HTML can be displayed on any browser.
- The client part of the Web application is small and can be downloaded quickly.
- The server can tailor the content returned to the client based upon user attributes.

The Server Drive UI model, however, has one main disadvantage. Because the server defines the UI, generation of UI for each client consumes more processor resources on the application server such as extra clock cycles, or file and database accesses required to obtain and perform formatting of the data.

3.3.2 Client Driven UI

In the Client Driven UI model, the UI to be displayed on the client is generated by the part of the Web application that executes on the client. In response to a request from the client, a Web server returns data, which is interpreted by the client and displayed on the browser. The server does not return a new HTML page.

The most significant difference from the Server Driven UI model is the use of a command server that acts as a message dispatcher between the client and the business logic running on the Web server in the Client Driven UI model. The command server receives an interaction request from the client and interfaces with the same business logic components as the Server Driven UI model. Instead of returning HTML to the client, the

command server returns raw data. The UI part of the Web application running on the client then interprets this data and formats it so that it can be displayed in a desired format to the user. A unique command server can be implemented for each interaction or a general command server can be implemented to handle many different interactions.

The Client Driven UI model has the following advantages:

- Web application developers with a traditional client/server background would find this programming model familiar.
- Applications can be made to look more like standard windows applications.
- The computational load on the Web server is reduced because instead of the server, the client now is responsible for generating the UI.

The model also has some disadvantages:

- It is tricky to correctly partition the client application events from the raw data generation events, so that display updates can occur with the least number of remote method calls.
- The client code (for example, Java applets or JavaScripts) tends to be large and requires more time to download and initialize. Over a slow Internet connection, this could introduce significant delay before the start of an application.

3.4 Advantages of Web Interface for Applications

3.4.1 Ease of Version Control of Data

With desktop-based applications, data input and output are archived on local machines. For example, salespeople for a company may need to enter the amount of sales made in each month, and the manager has to make sure the sales team has been updating the sales log sequentially in order to make sure no data is overwritten or corrupted. This

can be done in many different ways. For example, data files are contained on removable storage media (such as floppy disk or Zip disk) that are passed along by the team members, which leads to significant delay for completion of data updates. Another example would be having a data administrator to take care of all the data manipulations, but the workload of the administrator would become unbearable if there were a large amount of data. There can also be a shared directory where team members can all access the same files and carry out their updates; this practice is typically seen in corporate intranets.

Web applications typically involve one central database or several distributed databases. Users all carry out their data updates and manipulations through the Web interface and access the same database. This can assure that each person's actions on the data are accounted for. This is similar to a shared directory on corporate intranets except that Web applications do not necessarily need to be restricted to corporate intranets and thus provide greater flexibility. Desktop applications that communicate to a database server – that is quite similar to the Web model and is the most common system architecture today – solve the problem of version control but demands more computational power on the desktops.

3.4.2 Accessibility from Different Locations

Web applications can be deployed on a single computer, an intranet within an organization, virtual private networking (VPN), or on the Internet. This greatly reduces the geographical restrictions of access to the applications. A user can have access to the functionalities and data of the application as long as it is within the scope of deployment of the Web application, so that it is not necessary to work on a specific computer or workstation. This flexibility gives users the appealing options of telecommuting and

working from home, and allows them to have access to useful and urgent data from different geographic locations.

3.4.3 Cost Saving

There is high cost associated with the purchase of desktop-based applications. Such software often has a high price, and companies that need to purchase software for use among several users would need to pay for the cost of not just one copy of the software but for the number of users using the software, and upgrading of existing software to a newer version also requires capital investment. Some desktop applications have "floating licenses" that can be purchased to have just enough licenses for the active users at one time but this does not eliminate the cost of installation on terminals as well as the need for more computational powers on the desktops.

Web applications can help solve the cost problem because companies can purchase subscriptions based on the number of logins needed. Expensive systems such as accounting and project management software are good candidates for deployment as Web applications. Subscriptions fees tend to be lower than purchase of software, and there are incentives for the application service providers to improve the quality of the software to raise their competitiveness in the market, and subscribers can enjoy the benefits of the regular updates without having to worry about incurring additional costs.

In addition, the application elements, including the content that is sent to the client to drive the application, all reside on the server. This allows all management of the application to be done on the server. The application leverages the Internet infrastructure for client/server communication and is dependent only on standard software (the Web browser) in the client. Therefore, connectivity and client management costs are virtually eliminated on a per application basis.

3.4.4 Platform Independence

Desktop-based software is platform dependent, and it will not run properly on platforms it is not intended for. This introduces difficulties when companies consider upgrades or migration of their operating systems. Software developers would also need to develop the same software for different platforms. Many applications, such as the popular Office suite, have a Windows version as well as Macintosh version. Exchange of data across platforms using the same software would also have problems with regard to the file format; for example, documents generated on Macintosh platform may not be fully compatible with the Windows operating system. Companies that use more than one operating system would need to find ways to convert their files from one format to another.

Web applications are run on browsers, which are often included in operating system packages or can be downloaded from the Internet at virtually no cost. Application service providers only have to think about what kind of server to be used and do not need to worry about the operating system on the machines of end users. Companies with multiple operating systems can also benefit because they do not need to obtain different copies of the software for each platform or worry about file formats across platforms.

3.4.5 Abundance of Tools and Technologies

There are many different tools and technologies for development of Web applications and developers have great flexibility in combining the use of several tools and technologies. Developers can also choose among them the technology that suits their development needs the best.

3.4.6 Ease of Distribution

The use of Web applications saves users the time and trouble of installation of software. The Internet being a very broad and effective channel of distribution makes the introduction and adoption of a Web application much faster and easier than the traditional software. The need for bug fixes would also be greatly reduced because the application resides on the server and the application service providers can remove most bugs internally without notice of the end users.

3.4.7 Use Across Different Client Products

Clients support industry standard communication protocols such as TCP/IP and HTTP. Thus, the simplest information appliance potentially has access to the same Web application that a Web browser running on a computer does. This enables both users of wireless PDAs and PCs to access their e-mails, spreadsheets, etc. in the same way.

3.5 Disadvantages of Web Interface for Applications

3.5.1 Security

A Web server can be placed on an intranet or on the Internet, where it is accessible to anyone with a computer and a browser. Security measures need to be taken to protect server's data from being seen, changed, or damaged, and protecting the server from malicious attacks.

Web applications often involve access to sensitive data or company information. Servers for electronic commerce, for example, would hold credit card information and data related to customer privacy such as address and shopping patterns. The server has to provide different access levels to the data, protecting information from being accessible to the general public.

Web servers are open to the possibility of attacks by crackers who want to change the contents of the server, interfere with the normal operation of the server (denial of service), or even crash the server. Servers with Web applications have programs and scripts that would be executed on request, and they can be a security hole that opens the server to attack.

Web applications often identify users by a username and password. More robust forms of authentication (such as certificates and encryptions) can make remote access to the Web applications more secure. After the user has connected and logged in, the amount of access should be limited to the minimum needed to get the job done.

Administrators that log in remotely would have the same access to the server as if he were physically present in front of the computer, and it is another security concern if the administrator's account is compromised.

3.5.2 Single Point of Failure

Centralized network suffers from the risk of single point of failure. If the Web server or database server is down, users cannot use the Web applications. In traditional desktop-based software, if one workstation is down, users can go to another workstation in the company if the software is also installed on that computer, given that the data needed is available. The availability of the Web applications is dependent upon the availability of the servers as well as the network traffic condition. Multiple servers running as backup Web servers and database servers would help solve this problem.

3.5.3 Performance Problems

A key difference between Web applications and desktop applications is that Web applications deliver slower performance as compared with desktop-based software because

performance depends heavily on network traffic condition, as well as the time needed for data to flow back and forth between the server and the client.

3.5.4 Network Costs

A company utilizing Web applications would also incur a cost in maintaining the server as well as paying for the network usage. This can accumulate to a significant amount of cost and offset the cost saving from purchase of only one copy of the software.

4 Technologies

4.1 History of Web Applications

4.1.1 CGI

One of the first practical techniques for extension of static content to dynamic content is the development of Common Gateway Interface (CGI). CGI was first developed to define a standard method for an information server to talk with external applications. The method enables a Web browser to send a request to the server for execution of a program. The output of the program is converted and formatted to a form that is readable by the browser and displayed in the client as if it is a static HTML page. With this capability, it is possible to implement a large variety of functionalities in Web pages by developing corresponding programs on the server side, and the CGI quickly became a popular method to include dynamic content on Web pages. It also transformed the Internet from a place to share information to a platform for information processing. Although CGI programs can be written in many different languages, Perl has become the most popular choice because of its advanced text-processing capabilities and its semblance of platform independence, but each request would need to start a separate Perl interpreter and that would take more processor time and resources.

When a request is sent from the client to the server to execute a CGI program, the server creates a new process to run the CGI program. The server then passes to the process the variables, user inputs, and other information that would be necessary to execute any computations and generate response. Creation of process for every request requires server resources, and therefore limits the number of requests the server can handle simultaneously and also affects the performance speed. To help improve

performance, there had been the creation of client-side scripting solutions that enable the client's browser to process some of the tasks before sending request.

Another problem with CGI is that CGI programs cannot interact with Web server or take advantage of the server's running processes because CGI programs run in separate processes.

4.1.2 Client-side Scripting

HTML does not allow for interactivity for Web applications, and so Netscape Communications developed along with Sun Microsystems a solution called LiveScript that allowed limited programming instructions to appear in Web pages and had the instructions being processed when the pages were viewed using Netscape Navigator. LiveScript was later renamed to JavaScript because of the popularity and attention around Java. However, it is not a subset of the Java language; only its syntax is similar to that of Java. JavaScript was made to interact with limited capacity with the client machine and over the time it slowly evolves to be more secure and safe.

Microsoft also created a scripting language called Visual Basic, Scripting Edition, or VBScript. VBScript is a subset of Visual Basic for Applications (VBA) language and its syntax is exactly the same as VBA. Microsoft also created Jscript, which is similar but not identical to JavaScript. Its browser would support all these different scripting languages.

4.1.3 FastCGI

FastCGI was developed by Open Market and it works just like CGI. The big difference between CGI and FastCGI is that instead of having a new process for each request, FastCGI creates a single persistent process for each FastCGI program. FastCGI, however, still has problems with process proliferation. It needs a pool of processes to handle concurrent requests, which can be slow.

4.1.4 Other Solutions

CGI has the advantage of being largely platform independent. Companies have developed some other more efficient solutions on specific platforms.

Several companies developed proprietary server extension Application Programming Interfaces (APIs) for the Web servers to more efficiently carry out Web requests. The APIs utilize server extensions to handle requests. After the Web server gets the first request for a particular application, that server extension is loaded in the same memory space on the Web server on subsequent requests for the application. The server extension stays in memory and answer requests until it is explicitly released from memory. Therefore there is no need to instantiate a new application every time a request is made. This arrangement has the advantages of being more memory efficient and faster.

The server extensions commonly use linked C or C++ code. These plug-ins, as well as CGI programs, operate with relatively unregulated access to the server on which they are executed, introducing big security issues.

4.2 Netscape Communications Server/Netscape Commerce Server

Netscape came up with Netscape Communications Server and Netscape Commerce Server. The Commerce Server is essentially identical to the Communications Server except that it adds Secure Sockets Layer (SSL) security to the mix, for performing secure transactions over the Internet, as well as other advanced security features such as server authentication, data encryption, data integrity, and user authorization.

Netscape delivers HTML documents and dynamically generated output over the Internet and other TCP/IP networks using HTTP with its Web server and CGI extensions.

Netscape Commerce Server's API (NSAPI) allows dynamic extension of server functionality and easy integration of add-on applications and systems. Process manager allows the creation of a configurable number of processes that reside in memory, waiting to fulfill HTTP requests. This eliminates unnecessary overhead of creating and deleting processes to fulfill every HTTP request. The dynamic process management algorithm increases the number of server processes within configuration limits to efficiently handle periods of peak demand. Netscape claims that it would be able to deliver several times greater throughput by this algorithm. It also dramatically reduces system load and increases system reliability. This efficiently leaves additional CPU resources available for running other applications. The NSAPI also provides significantly higher performance than CGI because a new process needs not be created to run the external function or application.

4.3 ASP

Microsoft came up with an alternative to CGI, called the Internet Server Application Programming Interface, or ISAPI. ISAPI applications are normally faster than CGI programs that perform equivalent tasks because it utilizes the server extensions concept. The ISAPI also allows for development of customized dynamic link libraries (DLL) that sits in the same memory space as the Web server and would be called by Web server in response to every HTTP request. Such DLLs are called ISAPI filters. There can be many examples of possible ISAPI filters, such as a security layer between the Web server and the client, an interpretation layer that presents a stream of information from server in a different format than would the original Web server, and a mapping function that can redirect a client's request to a different physical location on the server (such as a mirror site) when the Web site experiences high traffic.

In Microsoft's development of Internet Information Server (IIS) 2.0, a technology known as Denali was beta tested. Denali later evolved to the Active Server Pages (ASP) and has become an important factor in IIS strategy. ASP technology is encapsulated in a small ASP.DLL file that is an ISAPI filter in itself. When a browser sends a request for a file with extension of .ASP, IIS passes the requested document to ASP.DLL. ASP then loads the required scripting language interpreter DLLs into the server's memory, executes the code, and passed the interpreted result back to IIS. IIS then inserts the results into the HTML text stream and sends it to the client.

4.4 Java Servlets/JSP

Servlets are programs running on a Web server and serve as an intermediate layer between the clients' requests and the databases or applications on the server. They can be loaded dynamically into the Java Virtual Machine (JVM) on the server to expand the server's functionalities. Servlets can not only serve Web applications and handle HTTP requests, they can also extend any sort of server such as an FTP server or a mail server.

Unlike the traditional CGI that utilizes multiple processes to handle programs and requests, servlets are handled by separate threads within the Web server's main process. Because servlets run within the Web server, they can interact closely with the server, which was impossible with CGI scripts, at least not without using a server-specific API. Communication with Web server eases the translation of relative URLs into concrete path names. Multiple servlets can also share data among each other, making it easy to implement database connection pooling and optimize resource sharing.

Servlets work by first reading data submitted through requests by the user, which can be in the form of form data, Java applet, or a custom HTTP client program. The requests also contain other information that may be needed for processing the requests, such as

information stored on cookies, the IP address of the client machine, etc. Using the user input and parameters, the servlets would generate the results, which can include exchange of information with databases or direct computation by the Java servlets. In most cases, the output from the process would be embedded using HTML code, and the servlets would set the appropriate HTTP parameters, such as document type, and then send back the document to the client. The document can be in a variety of formats, such as text format as HTML file, binary format as GIF images, an Excel spreadsheet, or some other formats.

Servlets are run inside the JVM, which stays running and handles each request using a lightweight Java thread., so multiple requests would not require programs to be loaded into memory multiple times. CGI programs also have the disadvantage of being difficult to cache computations, keep database connections open, and perform other optimizations that rely on persistent data. On the other hand, the servlet classes remain in memory even after they complete a response and make it straightforward to store complex data between requests.

Servlets are written in the Java language and the standard APIs, and so they are portable across different platforms that run the JVM. Servlets are supported either directly or by a plug-in on virtually every major Web server.

The cost of Web servers for serving servlets is also relatively inexpensive. There are a number of Web servers available good for personal or small business use that are free or inexpensive. Adding servlet support to Web server without servlet capability costs little. This represents a much higher cost saving when compared with CGI alternatives or IIS.

Many dynamic pages that utilize servlets are largely static in content, with a few locations where dynamic contents need to be fed. Servlets generate the entire page via

programs even though most of it is always the same. A technology similar to ASP, the JavaServer Pages (JSP) technology was therefore developed to enable mixing of regular, static HTML with dynamically generated content from servlets. JSP can in principle accomplish any capability as servlets. But it has a major advantage over servlet that it allows separate development of the two parts, so that the tasks of Web design and programming can be distributed and better make use of various developers' abilities.

4.5 Others (such as PHP, Server-Side Includes)

PHP is a free, open-source HTML-embedded scripting language similar to ASP and JSP. It is a particular choice for use in combination with Linux platform, Apache server with PHP module, and MySQL database because of the apparently zero costs of these packages. PHP also has a large number of advocates in the open-source community and developers in PHP are able to seek answers and helps within the community's many message boards.

4.6 Structuring Web Applications: Case Study on Java Technologies

4.6.1 The Structured Interactions Model

This section illustrates structured interactions as a structure for Web applications. It helps produce maintainable applications that can be efficiently developed.

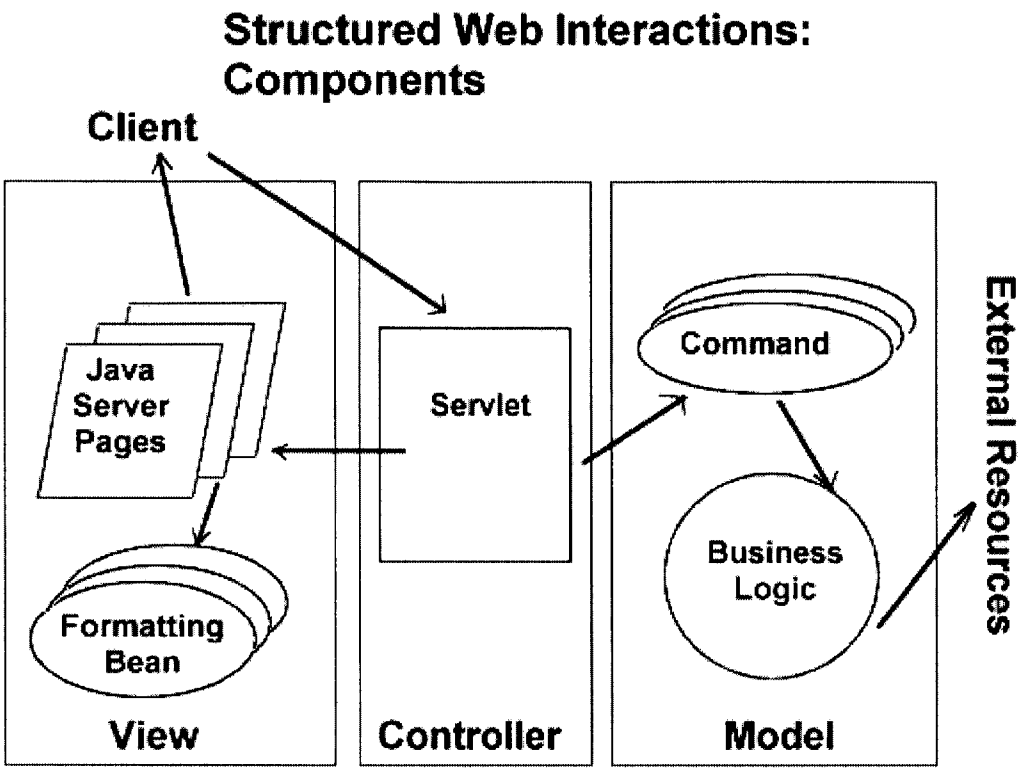


Figure 4-1 Structured Interactions

Figure 4-1 shows the major elements of a structured interaction and their relationships. In an ideal environment, the components indicated allow a perfect separation of skills and roles; however, most projects will find that some blurring of the lines will be acceptable and even desirable. In addition, each component is capable of doing more than its assigned role within this structure.

In structured interactions model, the interaction is divided into a number of components that closely follow the classical model-view-controller structure for applications. This structure allows different skills and tools to be used for different elements of the application and for the development sequence of the various parts to be largely decoupled.

4.6.2 Flow of Control

The normal flow of control that occurs in the processing of an HTTP request with a structured interaction is:

- The HTTP request arrives and is dispatched to the appropriate servlet. This is done based on configuration rules provided when the application is installed.
- The servlet makes calls on one or more command beans to perform logic of the request.
- The beans implement the business logic and may leverage relational database access, connectors, or the Web application infrastructure to perform the computations.
- Based on the results of the computation, the servlet either selects and invokes a JSP or does an HTTP redirection to another structured interaction. In the first case, the servlet will need to post some of the results of the computations to the execution context for use by the JSP before calling it. In the second case, the servlet will usually need to place some of the results of the computation into the session context so that it can be picked up by the next interaction.
- The dynamic content on the selected page is filled in from the request and session context and the page is sent back to the client. This may leverage formatting beans, other JSP pages, and static content components.

4.6.3 Components of the Structured Interaction

4.6.3.1 Servlets

Servlets provide the coordination between the programming elements and the content elements of an application. Servlets are generally implemented as beans with a set of customizable properties that are adjustable for different application configurations.

Examples of such properties include JSP names to invoke for specific interaction outcomes, or database names and connections to be passed to the command beans.

4.6.3.2 JSP

The role of JSP page is to make the final decision about the format and content of the response to a request. Each JSP page is associated with one or more outcomes of one or more servlets and this association defines the primary role of the particular page.

JSP pages are intended to focus on presentation rather than business logic. Therefore, it is important that JSP development does not require advanced programming skills in normal cases. However, sometimes presentation can be complex, and so the model also includes the use of formatting beans.

4.6.3.3 Formatting Beans

Formatting beans are Java beans that format various types of dynamic content for use in JSPs. These help to simplify JSPs so that they can generally be constructed without requiring programming skills. They also provide a single point of change for formatting decisions that span the entire application.

4.6.3.4 Command Beans

Command beans are beans that each implements one basic business logic step. It is common for a servlet to invoke several command beans to accomplish one application step as seen by the end user. Command beans are used in the following style:

- Create an instance of the bean
- Set some or all of its input properties (which may have some preconfigured values)
- Call its perform method

- Extract values from some or all of its output properties (which may also have a set of preconfigured values)

The command beans should be given any information they need from the HTTP request explicitly (by setting appropriate input properties) so that they are isolated from the form and details of the HTTP request.

4.6.3.5 Business Logic

The business logic refers to Java codes that implement the computational functions. There are two categories of programming logic within this model:

- General programming logic: It refers to anything that can be written in Java. General programming logic may leverage transactions in external resource managers such as relational databases, but it may not directly participate (with recoverable resources) in the transactions or initiate transactions that span multiple resource managers.
- Enterprise Java Bean (EJB) based programming logic: It is constructed according to the conventions and requirements of the EJB specification and leveraging EJB containers, EJB enabled relational database access, and/or EJB enabled connectors. EJB-based logic can initiate transactions, participate in transactions, and leverage transactions that span multiple resource managers.

4.6.4 Summary of Case Study

The structured interaction model allows the servlet to act as a contract between the computational logic and the dynamic content that constitutes the application. The servlet bean's interface can be specified by content designer and then inspected to determine what computation must be provided. This supports the HTML first model of development. The servlet can also be inspected to tell content developers which pages need to be

produced and what dynamic information is available to include in those pages. This supports the logic first model of development. Once constructed, this decomposition provides the basis for rapid customization of dynamic pages thus supporting the customization model of development.

5 Case Studies of Different Implementations of Platform

5.1 General Features of Pavement Management Systems

5.1.1 Pavement Network Databases

Each pavement management system contains a defined set of pavements for maintenance, and the set of pavements forms a pavement network within the scope of the PMS. Data from different pavement networks would enter into separate databases for easier management.

The Network Definition database is mandatory for any addition of street sections. The Network Definition database contains the pavement and section ID numbers, which are the keys that connect different databases together. The Network definition data also contains data of each section such as the facility type, district, surface type, functional classification, etc.

The Pavement Condition Survey database contains detailed condition information about pavements within the Network. After inspection of street conditions, the inspector inputs the ratings for conditions such as raveling, corrugations, flushing, transverse and longitudinal cracking, rideability, etc. The Pavement management system would then use the data inside this database to calculate pavement serviceability indices (PSIs) and maintenance strategies.

A Construction and Maintenance History Database holds information of street sections after repair work is performed. The system would need inputs from both this database and the pavement condition survey database to determine pavement serviceability indices for pavements that have undergone repair work.

Pavement management systems may also have Traffic Survey Data, such as the estimated average daily traffic, number of single and multiple axle commercial vehicles,

guardrail and sidewalk types and conditions and median type. Managers can take the data into account when planning for maintenance and repair strategies, so that the strategies would create the least amount of disturbance to the normal traffic.

An optional Drainage Survey Database contains information about drainage conditions of sections within the network, such as culvert functions and conditions, headwall, ditch, side slope drainage and cross slopes, and utility and curb information. This helps manager take into account of the drainage system underneath the road surface when planning on pavement maintenance strategies.

5.1.2 Inquiry Functions

The pavement management system should provide user with the ability to locate and view information about different pavement sections in the Network. The user should be able to carry out inquiries based on criteria including network definition, pavement defect, pavement serviceability index, pavement type, and others.

5.1.3 Pavement Analysis

The pavement management system should be able to carry out basic analysis of pavement conditions, such as calculating the pavement serviceability indices after each inspection, and also make predictions of future deterioration as well as improvement on pavement condition after different kinds of maintenance actions. The user can create custom plan and then carries out the budgeting for the customized work plans. The system should be able to roughly estimate the cost of maintenance strategies, and calculate the estimated benefit ratio for different M & R plans, so that the manager can optimize the benefit achieved with the funding available. Inflation and change in costs should also be taken into account for budget analysis.

5.1.4 Reporting

Reports of the pavement conditions as well as for maintenance plans would be useful for record and planning purposes. The pavement management system should be able to produce reports and graphs based on different criteria set by the users.

5.2 InfraStructure Management System at Arlington, MA

5.2.1 Introduction

Arlington's Department of Public Works (DPW) uses the Infrastructure Management System (IMS) II version 1.2 software developed by Public Works Software Inc. in 1987. IMS implements most of the features that meets the needs of Arlington DPW. IMS II also has the ability to determine the optimum strategy. There are four methods IMS uses to calculate the optimum strategy:

- Public Works Software algorithm which attempts to keep the PSI of the road at a user selected PSI level over the length of the plan (5 – 20 years), while at the same time analyzing the tradeoff between the first year cost and the total project cost.
- The highest benefit cost ratio of a strategy.
- The lowest overall cost.
- The lowest initial first year cost.

5.2.2 Features

IMS features the following functions:

- *Database for record of maintenance data*: The database holds the Network Definition Data, Flexible Condition Survey Data, Construction and Maintenance History Data, Drainage Survey Data, and Traffic Survey Data. The database at the back end is dBase III, developed by Ashton-Tate, Inc.

- *Calculations*: IMS calculates the maintenance strategies based on the survey data. The user inputs the number of year he wants IMS to calculate the strategy, as well as the inflation rate to determine the distribution of maintenance plan costs over the span of the inputted years.
- *Graphs*: IMS has the ability to present the results from calculations in the form of graphs, including the PSI frequency bar graphs by different criteria such as facility type or district, distributed PSI graph of current and future condition, and projected pavement life cycle graph.
- *Inquiry*: The users can carry out inquiry on the collection of streets using the inquiry function in IMS.
- *Maintenance and Repair Plan/Budget*: The users can look at the various maintenance plans and budgets and the details. IMS also would recommend an optimum maintenance and repair plan for each pavement section based on a criterion entered by the user, such ash benefit-to-cost ratio.
- *Reports*: IMS creates reports on the databases, PSI calculations, as well as maintenance strategies. The user can print out reports for record.

5.2.3 Interface

IMS is a DOS-based program and navigation is done by keyboard inputs. The user carries out inspection using the paper inspection forms that are included with the software package, and inputs the inspection data by hand to the IMS.

IMS has an interface where the user navigates through the function menus by pressing function keys.

F2 - Primary Databases

F3 – Calcuations

F4 – Graphs

IMS can present the PSI values using three different graph types:

- PSI Frequency Bar Graphs: These are graphs of current and estimated condition based on facility type, district, surface type, functional class, zone, and responsibility codes for each of the pavement sections.
- Distributed PSI Graphs: IMS shows the percentages of occurrences for each PSI within the network of pavement sections.
- Project Life Cycle Graphs: IMS calculates the possible future condition of a section if a particular action is selected and displays the respective graph.

F5 – Inquiry

F6 – Maintenance and Repair Plan and Budget

F7 – Reports

IMS can create roughly 100 combinations of standard printed reports for various groups of pavement sections, including reports of the network, survey data, PSI values, and maintenance plans.

F8 – Database Support

5.2.4 Problems with the DOS-based Text Interface

There are several disadvantages of this interface:

- *Expense and Complexity for Inspection:* The DOS-based interface makes it very time consuming to collect and update inspection data.
- *Security*: The program does not have a log in session and unauthorized personnel can have access to the data and make modifications.
- *Difficulty in Navigation*: The functionalities of the program are separated into 8 categories, each provoked by pressing a Function key. Although each individual

functional category can be accessed easily, a user cannot link different functionalities together. For example, when a user enters the inspection into the Flexible Condition Survey Database or performs a maintenance plan comparison on a specific pavement section, he cannot view the details of that pavement without having to quit the function he is working on.

- *Difficulty in Distribution of Software*: The software is a DOS-based program and can be accessed only by the local terminal. If different users want to work on IMS, they have to work on the same workstation so that the data would not be corrupted.
- *Excessive Functions*: IMS is a commercial product that is developed to meet the requirements of many different municipalities including Arlington DPW. The IMS contains more functionality than needed by Arlington DPW, therefore making the use of the software unnecessarily harder.
- *Slow Calculations*: IMS recalculates maintenance plan and budget for either one section or the entire network. If the user wants to recalculates the budget for only a group of pavement sections based on a certain criteria, he would have to wait for IMS to complete calculations of all sections, which can take an extended period of time. The fact that different data are placed in separate databases rather than different tables in one database also slows down the calculation process.

5.3 Pavement Management and Inspection System (PMIS) at Arlington, MA

5.3.1 Introduction

PMIS for Arlington was developed to replace the existing pavement management system and to provide software that meets the needs of Arlington DPW. PMIS is built to address the general pavement management needs of municipalities, and we hope that it will be used by other communities. PMIS implemented a pavement inspection module, a series of pavement condition, deterioration, improvement and cost models, and an overall management framework within which specific maintenance projects can be selected and programmed.

5.3.2 Features

PMIS implements the common features present in essentially all Infrastructure Management Systems.

- *Record of inspection data*: Surveys of pavement conditions are taken, and the data is downloaded into the database system.
- *Assessment of current pavement conditions*: Data obtained from inspection of the street defects is used to calculate PSI, which reflects the conditions of the street sections and the need for remedial actions.
- *Estimation of deterioration*: Street conditions deteriorate without maintenance actions. PMIS estimates the rate of deterioration over time based on traffic, weather, and other street activities.
- *Estimation of pavement maintenance cost*: Different pavement actions have different unit costs; total costs also depend on the dimensions of the street sections where the

actions are applied. PMIS estimates the cost of application of various pavement actions, as well as maintenance costs of sidewalks and curbs.

- *Estimation of pavement maintenance benefit*: The benefit of a pavement maintenance action depends not only on the increase in PSI after application of action but also on the sustainability of the increase in PSI. PMIS estimates the total benefit from different actions using the deterioration models.

In addition, PMIS has several features that are not always present in infrastructure maintenance models, which allow the creation and use of scenarios to help more effectively allocate resources across different maintenance plans.

- *Creation of scenarios*: Scenarios hold collections of streets specified by the user to have met some criteria, such as being within a particular PSI range, district (precinct), a specific (usually major) street. Users can apply different actions to different street sections and PMIS returns the estimated benefits as well as costs in carrying out the actions. Different scenarios can be created for different goals in mind, and users can compare the final benefits and costs of various scenarios to determine the action plan.
- *Comparison of scenarios*: Users can make decisions by looking at individual scenario, and can also make use of the "Compare Scenario" feature to have a quick overall view of scenarios by comparing the overall average benefit as well as overall costs.
- *Recommendation of pavement actions*: In creating scenarios, PMIS calculates the benefit-to-cost ratios of different actions and recommends the optimal action. It uses a simple optimization method called the "greedy knapsack" method to compute the optimal actions.

- *Added security*: PMIS requires the user to log in before accessing the system. Different levels of access rights can be assigned to different users so that important data are not overwritten or deleted by unauthorized personnel.
- *Reporting*: PMIS generates reports of scenarios as well as of street sections, in the form of Microsoft Excel spreadsheets that would allow easy storage as well as formatting.
- *Improved inspection process*: The original Arlington system required the inspector to use a complex paper form for recording inspection data; then the data was manually entered into IMS. This made the inspection process slow and error-prone. The new PMIS system integrated the abilities of the Global Positioning System and Personal Digital Assistants and made the inspection process much faster and reliable. The inspector clicks on the PDA when he comes across different defects and the PDA program automatically generates reports, using data from the connected GPS (Global Positioning System) to determine the locations of data being taken. The data is then be uploaded to a computer and through a Web interface to PMIS. Under this system, the inspection process is electronic and data is more easily processed and stored.

The key benefits of the pavement management system are that, when properly designed and implemented, PMIS is highly effective in improving pavement maintenance, as well as sharply reducing the cost of frequent data collection.

5.3.3 Interfaces

The key elements of the user interface are:

- *Administration.* There are key tables that contain most street and pavement information. Each of these tables has a set of administration Web pages that allow Arlington staff to display, search, add, delete and modify data and parameters for the

system in the tables, including Street Data, Pavement Action, Curb Action, Sidewalk Action, Pavement Type, Facility Type, Functional Classification, Precinct, User Information, and Parameters.

- *Pavement analysis.* This allows the Arlington staff to create pavement management plans (scenarios) for a set of streets, usually for a single construction season. The system allows a user to select the set of streets to examine, to estimate the benefits and costs of alternative pavement maintenance actions on these streets, to select actions to be performed, and to summarize the effectiveness of the scenario.
- *Inspection.* This allows Arlington staff to download inspection results from the field, which were collected using Palm and GPS units. Field data may be displayed, searched and modified, and data may be added or deleted if necessary.
- *Reports.* The system generates a set of on-screen reports, and also allows the export of data to Excel for further analysis.

5.3.3.1 Web Interface

The main user interface in the system was implemented as a Web interface. The use of a Web interface in PMIS allows multiple users to have access to the same database and be able to work on data, such as uploading inspection data on one terminal and creating scenarios on another, at different physical locations. The Town of Arlington uses machines running on Windows 98, but all PMIS needs is a Web browser; it is not restricted to Windows 98. It makes future transitions much easier if the Town decides to shift to another operating system on some other platforms. The use of a Web application also reduced the need of an IT support staff that would help supervise the installation of the application on all machines.

The dynamic contents are served by the Tomcat server with Java Server Pages (JSP) technology. JSP were used to process user inputs and pass the parameters to Java Beans as well as output results returned from the Beans. The database being used was MySQL and JDBC provides the connection for database transactions invoked by the Beans.

The choice of combining Linux platform, Apache/Tomcat server, JSP and other Java technologies, and MySQL database was made because of cost and performance. This set of products was all freely available on the Web and could be downloaded without cost, with performance comparable to most commercial products. They might not have the full range of functions as some commercial products, but the capabilities were sufficient to serve the purpose of pavement management for Arlington.

5.3.3.2 Inspection System: Palm and GPS

The Palm and GPS units are used to inspect streets. The Palm unit has an application program that allows the inspector to click on defects with a stylus as a vehicle is driven along the street. Each click registers the date and time of the observation. In parallel, the GPS receiver is automatically registering (if possible) the location of the vehicle approximately every 50 feet and also registering the date and time. The Palm and GPS files are then downloaded and synchronized on a PC, and this becomes the inspection data used by the pavement management system.

The Palm has two major user interface screens:

- An inspection screen with icons of each defect type to be inspected. When these icons were selected, a default amount of defect is recorded. The icons might represent, for example, cracks of length 10, 20 and 40 feet. The inspector selects the closest icon, as many times as necessary, to record the pavement conditions.

- A summary screen. When pausing the vehicle after inspecting a street segment, the inspector switches to the summary screen, which will:
    - Total the defects for the segment,
    - Show the results of the previous inspection,
    - Compute and display the PSI for the segment.
    - Allow editing of the inspection data in the Palm, while still in the field, rather than waiting until the data was processed at the office.

Additionally, it has a startup screen to choose the pavement application from the main menu.

5.3.4 Building the Pavement Management System

5.3.4.1 Pavement Network Definition

The first task of the pavement management system is the network definition. In developing Arlington's PMIS, the network definition is the set of pavement sections within the maintenance scope of Arlington DPW. Since the scope of the pavement network is small, a single database is sufficient to hold the data. The list of pavements included in the pavement network definition includes public roads, parking lot, private roads, and other pavement types. The Arlington DPW is responsible for maintaining streets with pavement type of public roads only.

5.3.4.2 Pavement Database

Pavement database contains the data required for maintaining the street sections. The database contains the following tables:

Table Name	Description
Action	Description of various pavement actions and their unit costs
ActionEffect	Effect of different actions on different defects
ConstructionHistory	Construction and maintenance history on the street sections
CurbAction	Available curb actions and their costs
CurbDetails	Details of a curb action on a street section in a scenario
DefectData	Defects of the street sections based on the data from inspection and construction
Deterioration	Deterioration model defined by maintenance actions and functional types
DefectValidation	Number of manholes and gates, and the other drainage data
DrainageData	Number of manholes and gates, and the other drainage data
FacilityType	Types and ownerships of the street sections or facilities
FuncClass	Functional type of street sections
InspectData	Defect data from road inspection
Login	Details of sign-up user
PavementType	Construction material of the street sections
Precinct	Precinct information of Arlington
ScenarioDetails	The actions and details on street sections associated with a scenario
ScenarioHeader	The details of scenarios created
SidewalkDetails	Details of sidewalk actions on a street section in a scenario
SidewalkAction	Available sidewalk actions and their costs
StreetData	Primary details of all street sections
StreetName	All streets in Arlington
Tiger	Details of geographic information of the streets in Arlington from Tiger data

Table 5-1 Database structure of Arlington PMIS [3]

5.3.4.3 Pavement System Class Diagrams [4]

Figure 5-1 shows the class diagram of the database connection and user verification. *DbBean* provides the connection string and JDBC driver, and *ArchiDbBean* extends the *DbBean* class to provide additional customized methods for the Pavement system.

[3] Refer to Data Modeling in a Pavement Management System by Yim for details on the database structure.
[4] Refer to Evaluation of Infrastructure Monitoring System Using PDA and GPS Technologies by Cheung for more details on class structure of the inspection module on Palm PDA.

Authenticate bean verifies the user information and also helps maintain session information after the user logs in. *User* bean has access to the user information in the database and it is used for verification of username and password as well as for creating new users.

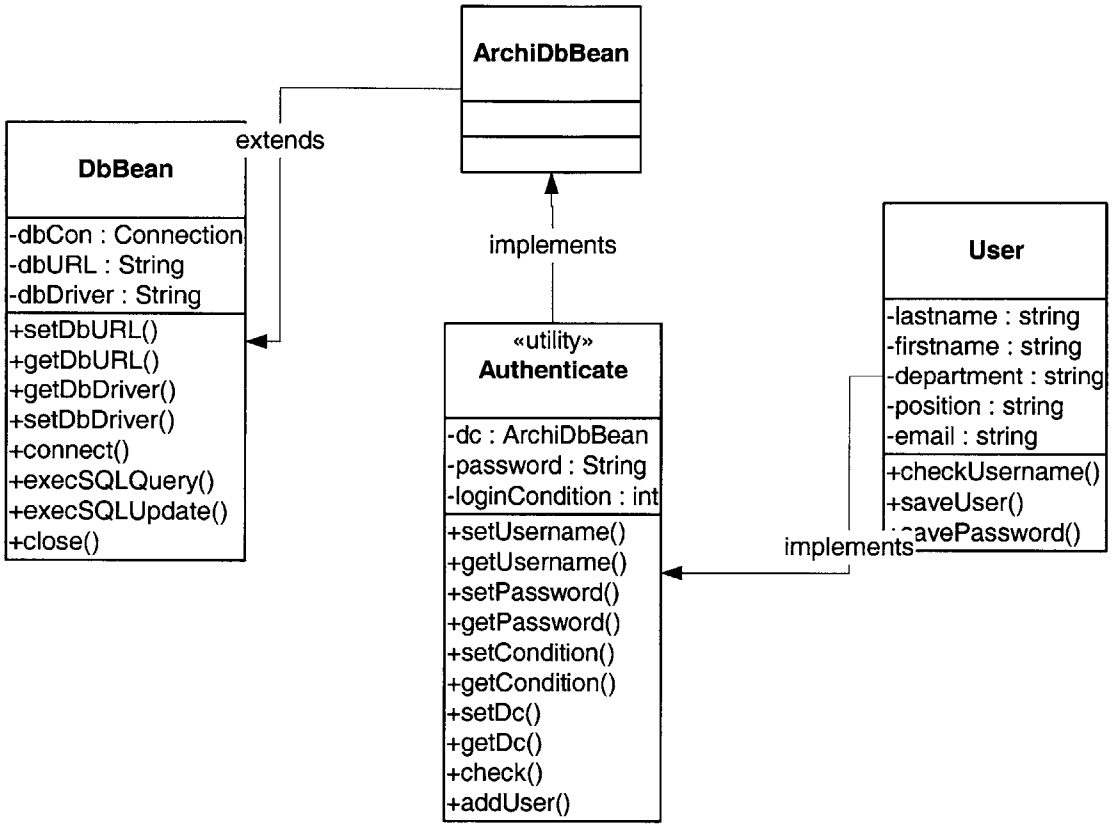


Figure 5-1 Class Diagram: User Authentication

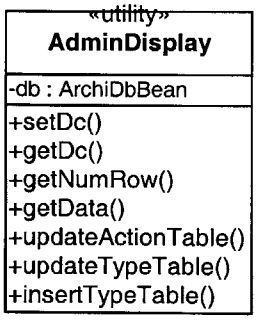


Figure 5-2 Administration

Figure 5-2 shows the class diagram for the administration display function. Tables such as street inventory, construction and maintenance, pavement type, etc. are viewed and modified using this function. *AdminDisplay* bean provides a way to view and edit different kinds of table using one business logic.

Figure 5-3 shows the classes used for Pavement Condition Inspection. *InspectionDataUploadServlet* is used to upload the inspection file (obtained from the Palm

inspection system) from the PC to the server by instantiating the *MultipartRequest* object. *InspectionFile* handles the data after the upload such as importing into the database. *InspectionFileBackup* controls and organizes the uploaded files in the Web server, in specific directories according to the upload time.

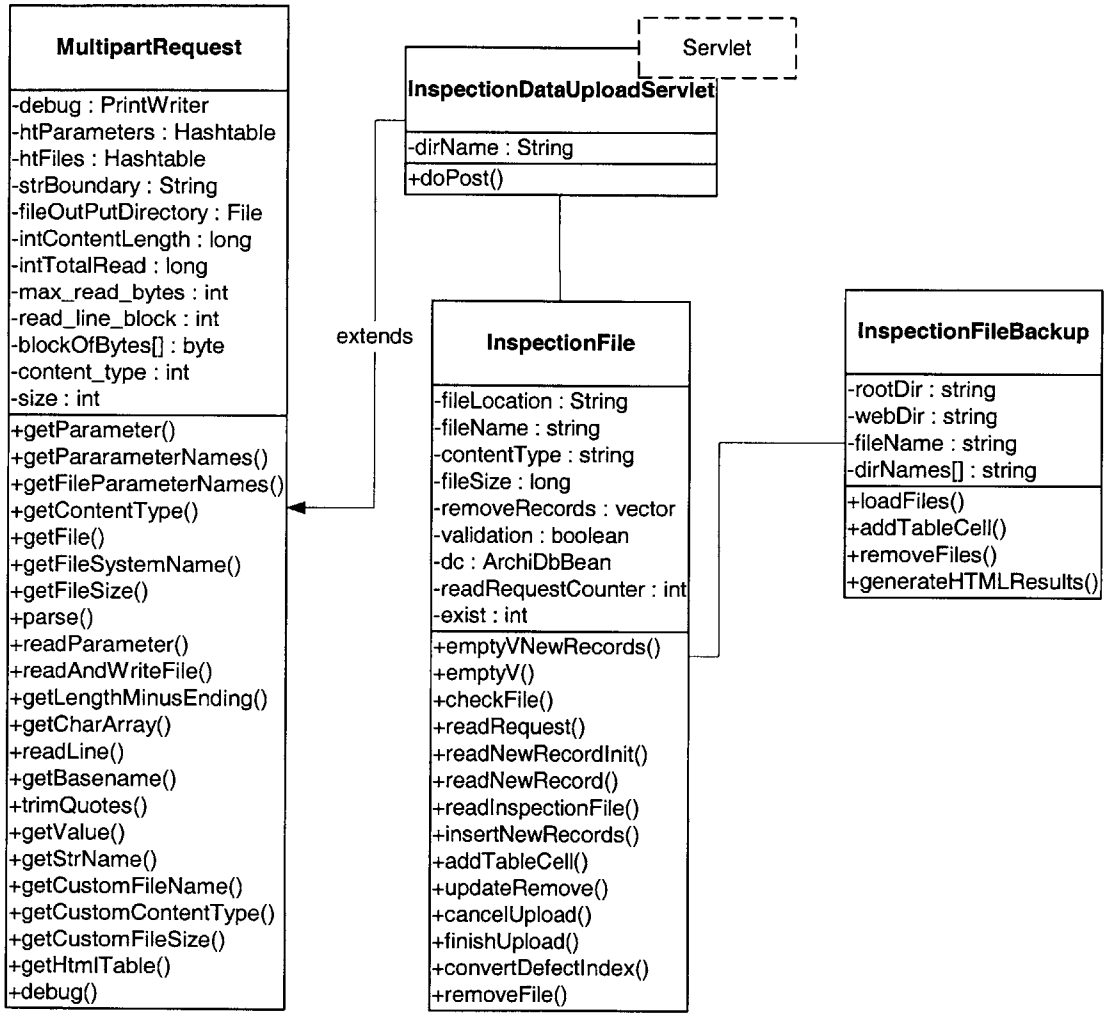


Figure 5-3 Class Diagram: Pavement Condition Inspection

Figure 5-4 shows the class diagram for the Network Management Budget Scenarios and Pavement Condition Prediction modules. The modules include 8 classes:

- *ScenName*: The user chooses a scenario to be loaded or creates a new scenario, and this class handles the initial switching of user options. It also stores the scenario name, scenario ID and year to pass to the subsequent pages.
- *CreateScenarioSummary*: This class creates a summary table of pavement sections that meet the query criteria entered by the user. It also handles addition and removal of street sections in the scenario as well as deletion of test scenario.
- *PavementAction*: After a user selects a set of streets that are potential candidates for maintenance, he can apply different actions to the street sections within the system and look at the predicted benefit of the actions. The user can also looks at the predicted PSI deterioration after the action is applied.
- *CurbSidewalk*: Apart from pavement action, curb and sidewalk maintenance actions can be applied to the street sections. Although they do not directly affect the PSI, they are essential for a pavement to be usable. This Java bean lets the user specifies different curb and sidewalk actions and also the costs for the actions.
- *CompleteScenario*: After a scenario has been put to action and the maintenance actions are carried out to the pavement, the user can enter the actual maintenance costs and other information about the construction and maintenance of the pavement, and the operations are handled by this Java bean.
- *CompareScen*: This class handles the comparison of all scenarios planned for a specific year. It creates summary that displays a weighted average PSI for each scenario, as well as details of each scenario. The user can alternatively choose to export the scenario details to spreadsheets for further data analysis.

- *StreetCondition*: The *StreetCondition* class is responsible for getting the various defect values and coefficients that are related to a specific street section for the calculation of PSI values.
- *PavementModel*: This is the class that contains the model for calculation of PSI from defect data, calculation of PSI after various pavement actions, and prediction of future PSI with deterioration model.

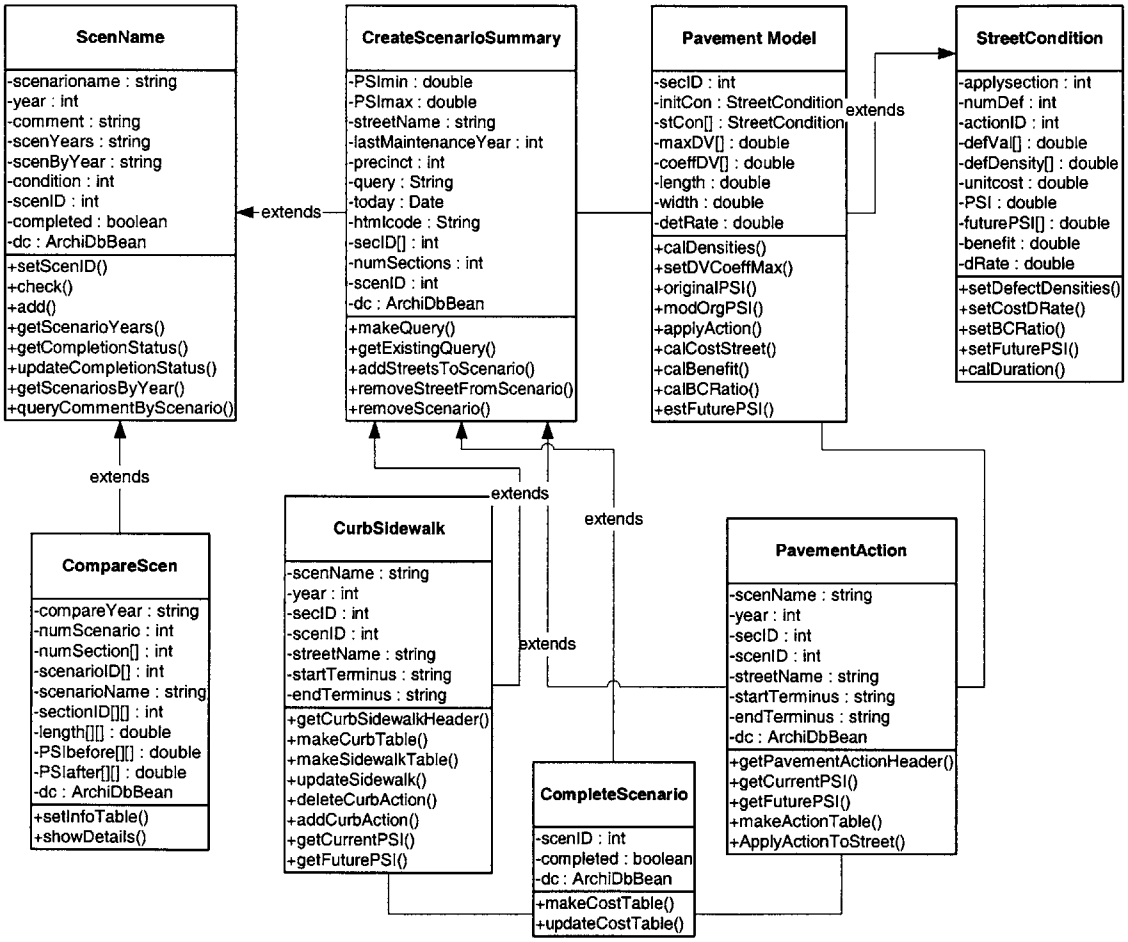


Figure 5-4 Class Diagram: Pavement Analysis and Budget Scenarios

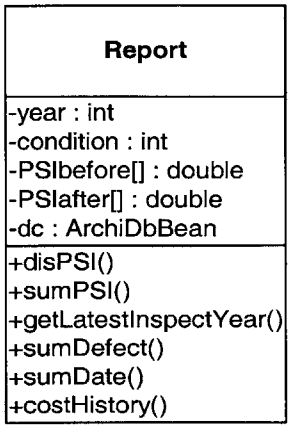


Figure 5-5 Report

The *Report* class contains methods to display reports based on different criteria selected by the user. For example, there are reports containing distribution of PSI values based on different criteria as well as defect data grouped by different functionalities of the pavement. The user can alternatively export the reports as spreadsheet files for further manipulation of data.

5.3.4.4 System Demonstration [5]

5.3.4.4.1 Pavement Network Database

Authorized users can log into the Arlington PMIS using their username and password pair. People who wish to obtain access to the system can apply for an account with the administrator and wait for approval. After logging into the system, a user can have access to the data in the pavement network database. Figure 5-6 shows the administration page, where the user can list the data within the database and also modify any data that they think are inaccurate or not up-to-date. The user can view and modify data in the following pavement data tables: Street Data, Pavement Action, Curb Action, Sidewalk Action, Pavement Type, Facility Type, Functional Classification, and Precinct. There is an additional table for User Information where the administrator can change login user information. The user can click on the links to view details of the corresponding tables.

[5] Refer to Software Development Process: Web-based Pavement Management System as Case Study by Durongdej for details on the Arlington PMIS application development.

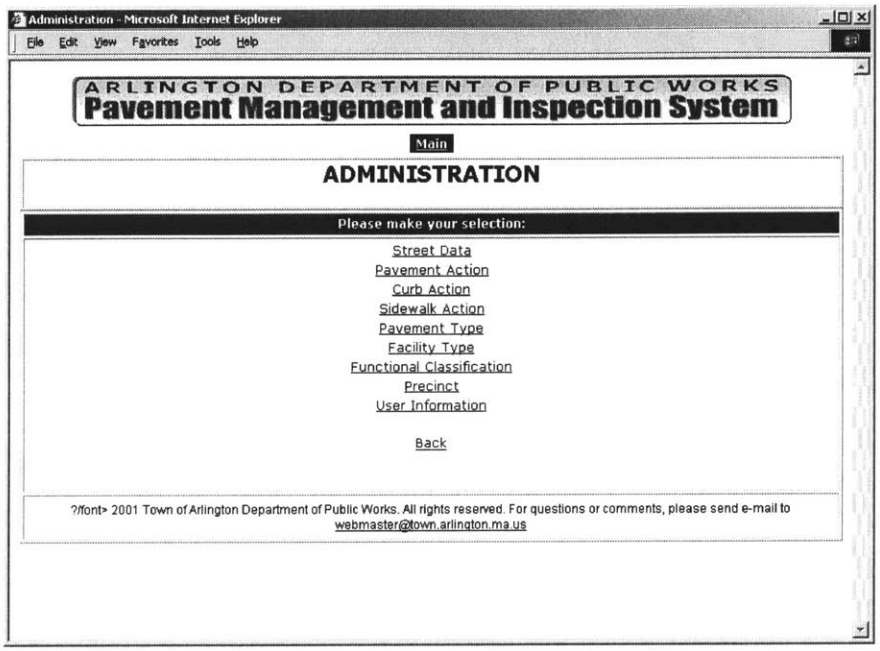
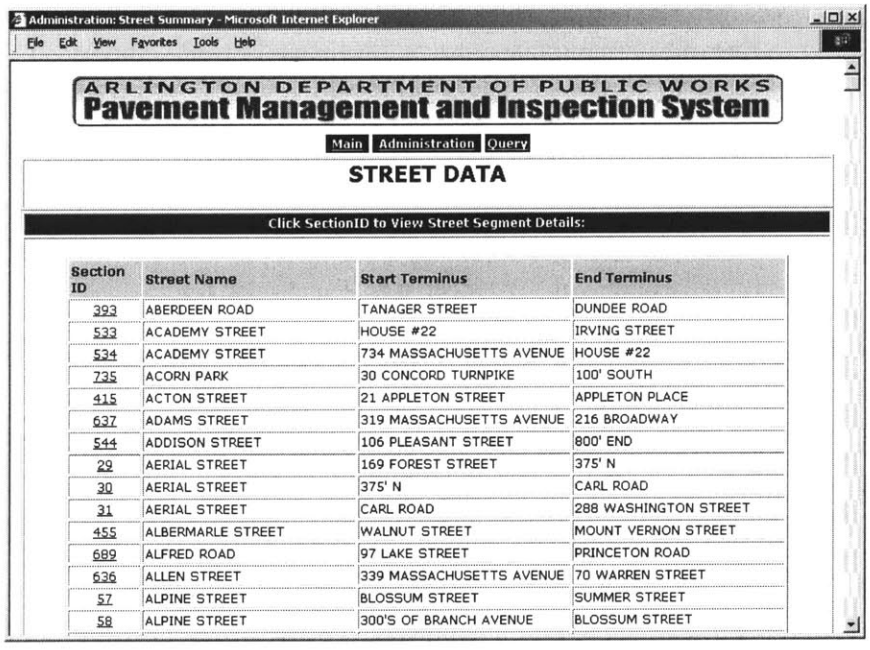


Figure 5-6 Arlington PMIS: Administration Page



Section ID	Street Name	Start Terminus	End Terminus
393	ABERDEEN ROAD	TANAGER STREET	DUNDEE ROAD
533	ACADEMY STREET	HOUSE #22	IRVING STREET
534	ACADEMY STREET	734 MASSACHUSETTS AVENUE	HOUSE #22
735	ACORN PARK	30 CONCORD TURNPIKE	100' SOUTH
415	ACTON STREET	21 APPLETON STREET	APPLETON PLACE
637	ADAMS STREET	319 MASSACHUSETTS AVENUE	216 BROADWAY
544	ADDISON STREET	106 PLEASANT STREET	800' END
29	AERIAL STREET	169 FOREST STREET	375' N
30	AERIAL STREET	375' N	CARL ROAD
31	AERIAL STREET	CARL ROAD	288 WASHINGTON STREET
455	ALBERMARLE STREET	WALNUT STREET	MOUNT VERNON STREET
689	ALFRED ROAD	97 LAKE STREET	PRINCETON ROAD
636	ALLEN STREET	339 MASSACHUSETTS AVENUE	70 WARREN STREET
57	ALPINE STREET	BLOSSUM STREET	SUMMER STREET
58	ALPINE STREET	300'S OF BRANCH AVENUE	BLOSSUM STREET

Figure 5-7 Arlington PMIS: Street Data Summary page

Figure 5-7 shows the Street Data summary page after the user clicks on the Street Data option in the administration page. The page shows a list of pavement sections sorted alphabetically with the start and end termini indicated. Other pavement data summary pages have similar layouts. The user can click on a specific street section and look at the details, as

shown in Figure 5-8. If the user wants to update any data for that street section, he can click on the "Edit" link and make the modifications and send to the database for update, as shown in Figure 5-9. The user can navigate into other tables through the administration page and carry out similar viewing and modification activities.

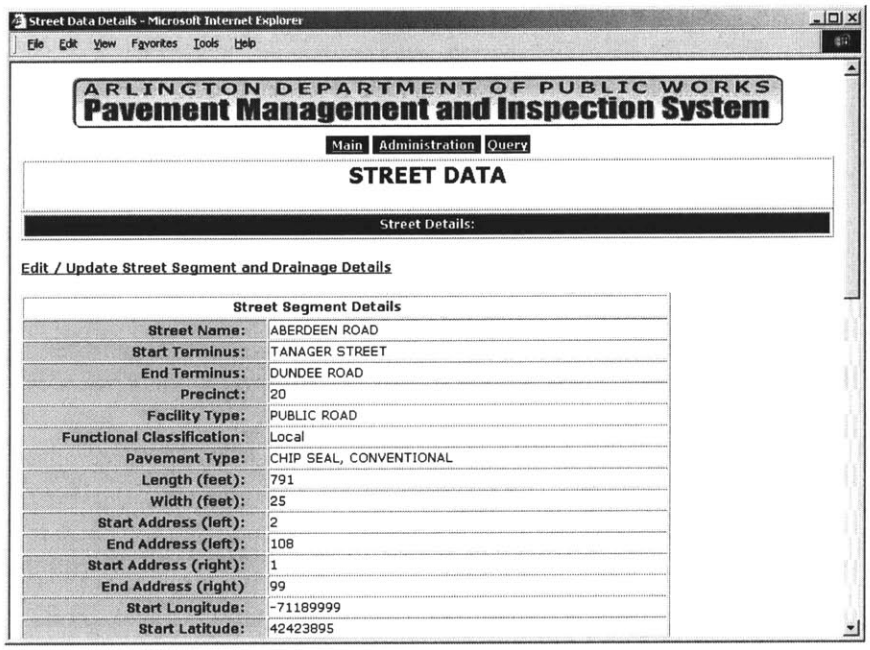


Figure 5-8 Arlington PMIS: Street section details page

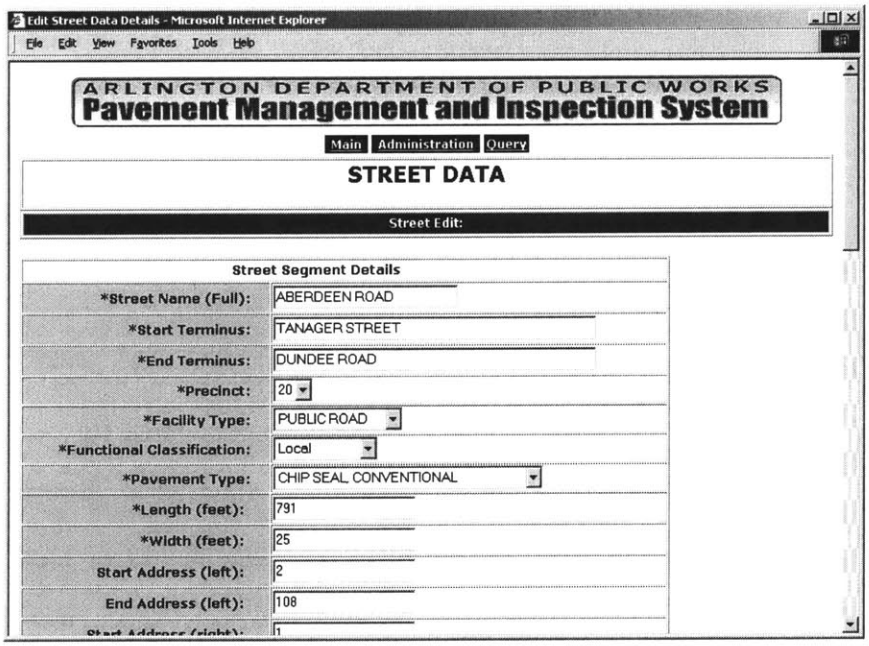


Figure 5-9 Arlington PMIS: Street section detail modification page

5.3.4.4.2 Pavement Condition Inspection

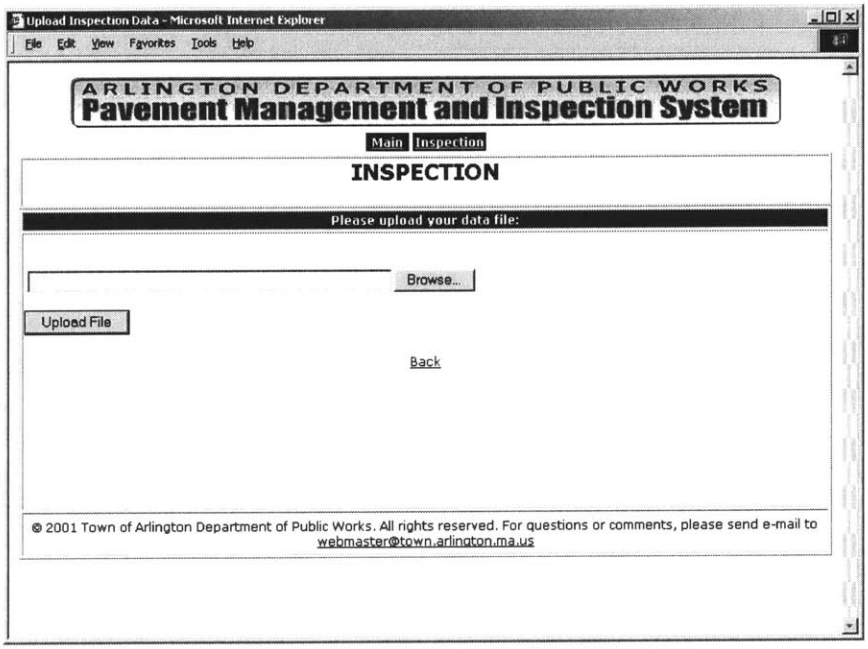


Figure 5-10 Arlington PMIS: Inspection file upload page

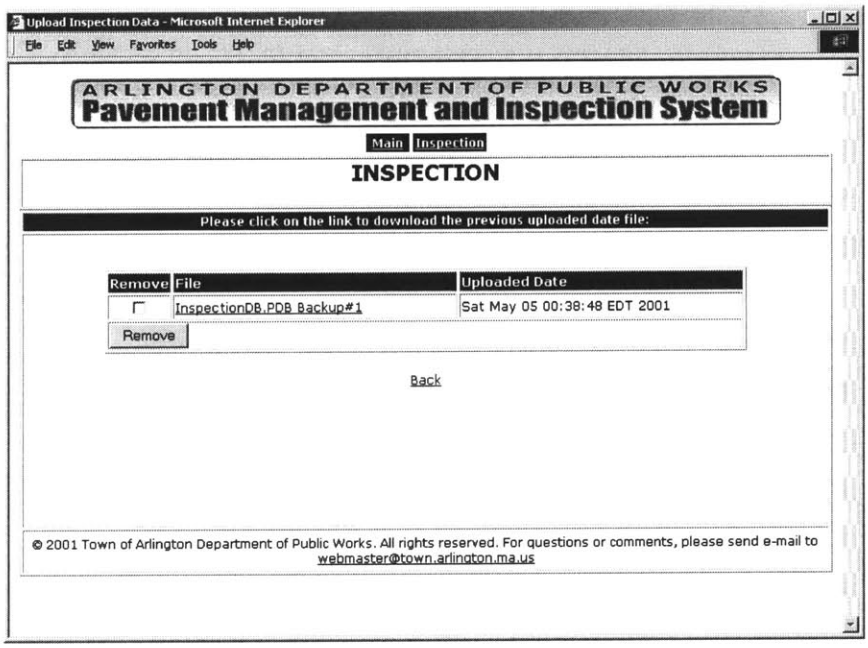


Figure 5-11 Arlington PMIS: Inspection file management

Figure 5-10 and Figure 5-11 show two of the functions in the Inspection module. After pavement inspection is done on the Palm device, the inspection file can be downloaded to a personal computer and uploaded to the pavement management system through the inspection data upload page. The system also allows the user to edit inspection

data after uploading the file so that any defect data inputted by mistake can be corrected before the data is updated into the database. The system also provides a log of inspection data files so that the user can download and view the details of inspection carried out on a specific date.

5.3.4.4.3 Pavement Analysis

After inspection, the user can carry out pavement analysis on the street sections to obtain PSI values of the pavements and also determine maintenance and repair strategies.

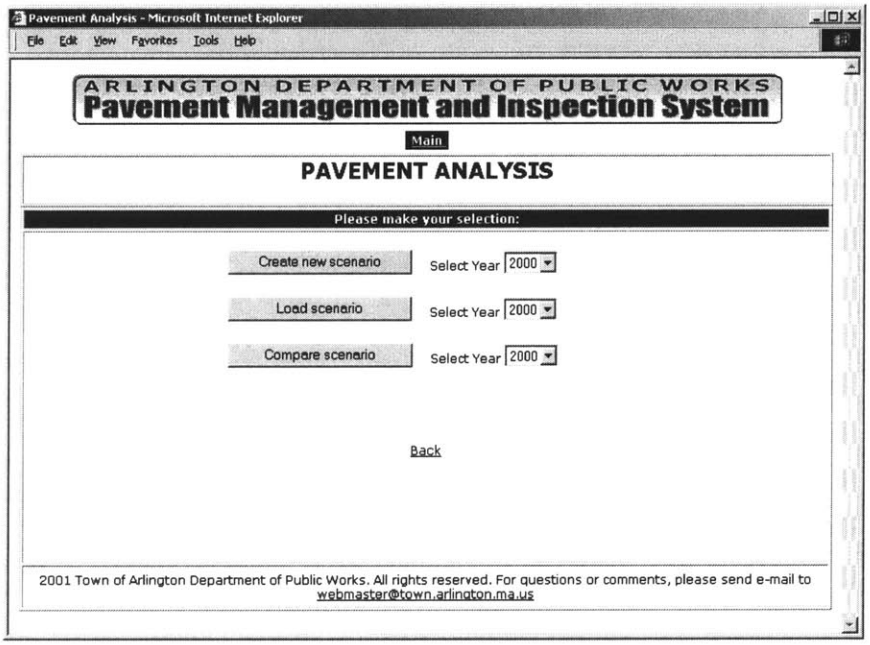


Figure 5-12 Arlington PMIS: Pavement analysis main page

Figure 5-12 shows the main page in pavement analysis, where the user have the option to create a scenario, load a scenario, or compare scenarios for a specific year. A scenario contains a set of street sections where M & R actions are planned for future implementation.

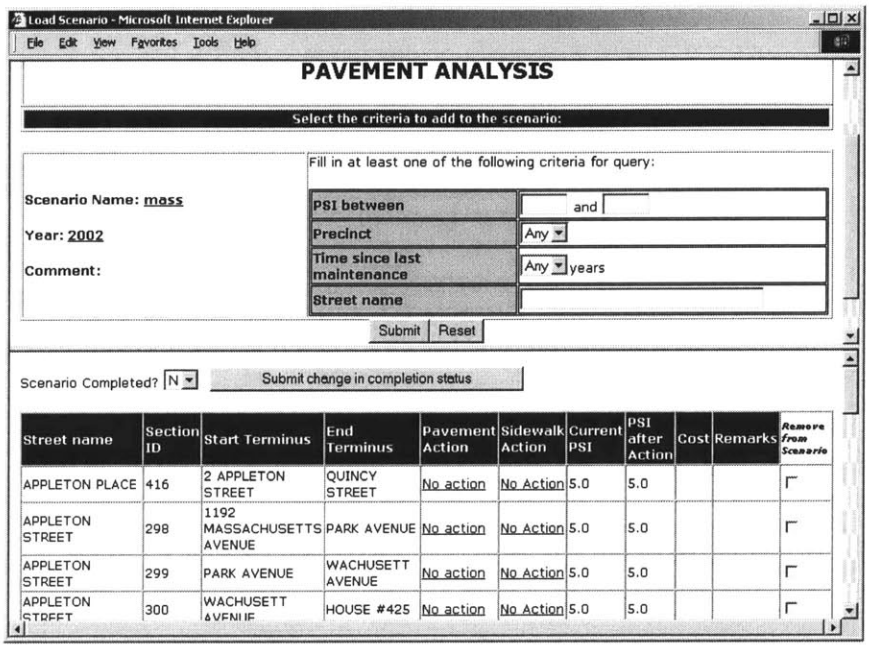


Figure 5-13 Arlington PMIS: Pavement analysis – scenario query and summary page

Figure 5-13 shows the page after a user creates or loads a scenario. The top frame is an area where the user can specify the criteria to add streets to the scenario, such as PSI range, precinct, last maintenance date, and street name. In the diagram, the user wants to look at the conditions of Appleton Place and so he enters the keyword "Appleton" in the Street Name field and leaves the other fields unchanged. The bottom frame then shows a list of streets that match the inputted query criteria. Selected details of the street sections are displayed, including the start and end termini, the selected Pavement Action and Sidewalk Action in the current scenario, the present PSI level and the predicted PSI improvement with the selected action, as well as any cost considerations. Since the user only wants to look at Appleton Place, he can check the "Remove from Scenario" buttons next to all the other street sections and remove them from the scenario.

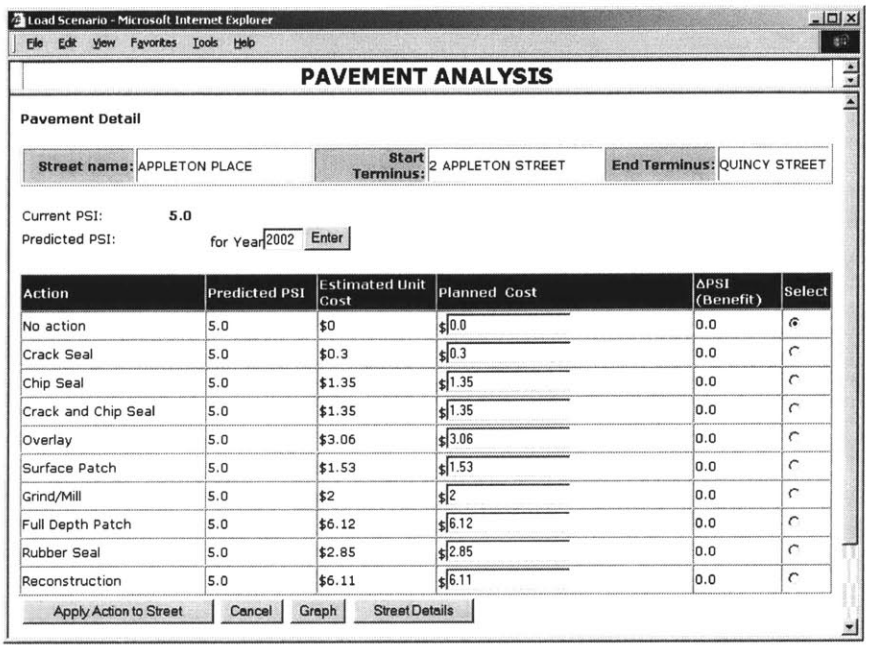


Figure 5-14 Arlington PMIS: Pavement Action selection

The user then decides to look at the various pavement action possibilities for Appleton Place. He clicks on the section number, and the system displays the pavement action selection page, as shown in Figure 5-14. The page shows the street name as well as the start and end termini, so that the user can make sure he has selected the desired street section. The action table shows a list of pavement actions, along with the predicted PSI of the pavement section after each action, the estimated costs for carrying out the action, and the benefit of that action into the future after taking into account of deterioration. The user can evaluate the benefits and costs of the array of action options and choose the one that fits his budget. After choosing the action, the user can save the selected action and go back to the Scenario Summary page and carry out more maintenance strategy planning.

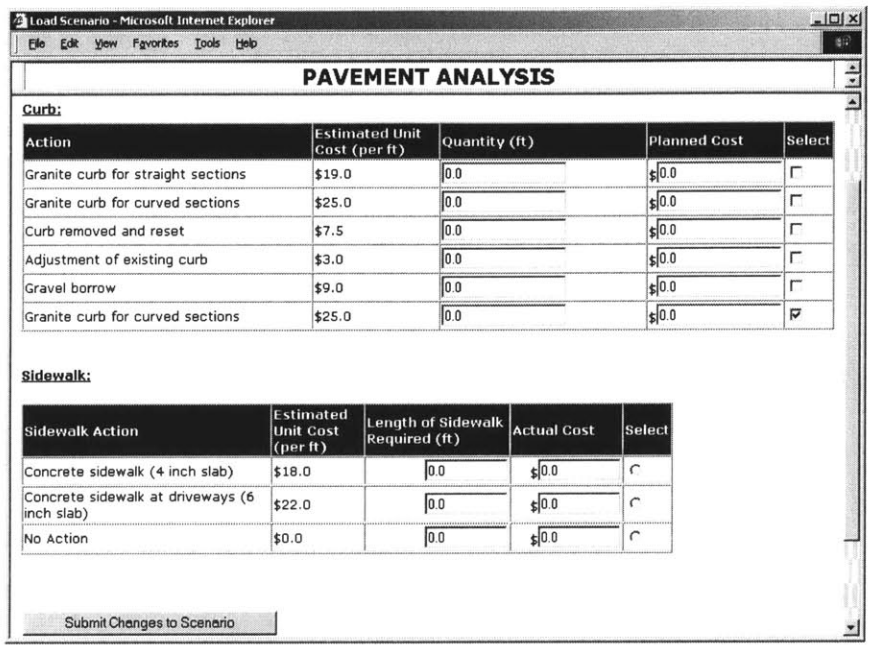


Figure 5-15 Arlington PMIS: Curb and Sidewalk maintenance action selection page

Apart from pavement actions, the user can also apply curb and sidewalk maintenance actions to the pavement sections. The user can go back to the Scenario Summary page and click on the Sidewalk action link to access the Curb and Sidewalk maintenance action selection page, which is shown in Figure 5-15. The page contains action tables for curb and sidewalk actions respectively. The system allows the user to plan out strategies and preview the cost needed, so that the user can make budget analysis based on different strategies. After the user select the desired curb and sidewalk actions, he can save the selection to the database and the summary page would automatically update with the latest selections.

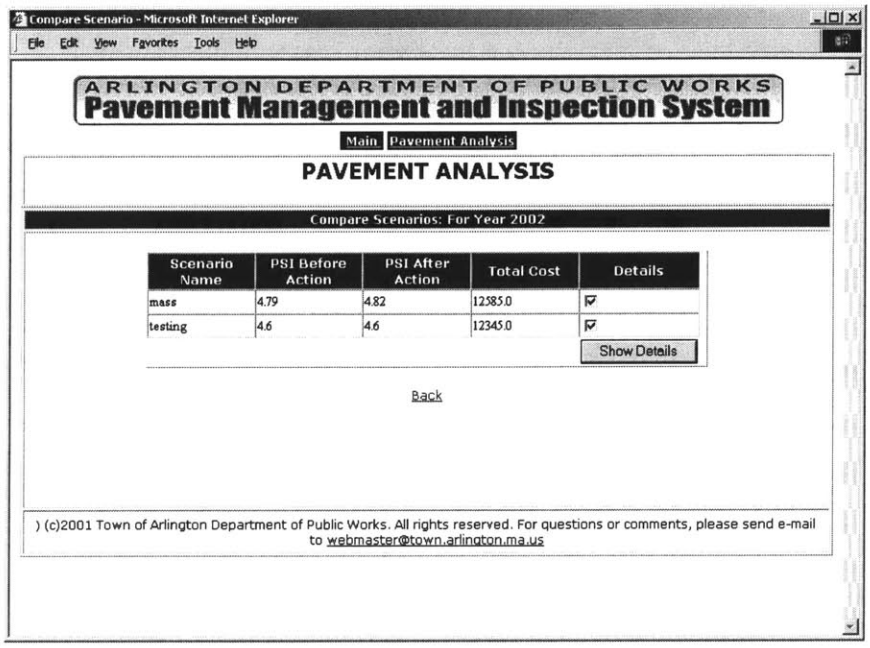


Figure 5-16 Arlington PMIS: Scenario Comparison summary page

A user can create various scenarios based on different criteria and at the end, he would want to look at all the scenarios created and decide on which scenario would actually be put into real action. The user can select among the scenarios created for a specific year in the pavement analysis main page and use the "Compare Scenario" to make a direct comparison across different scenarios based on overall PSI improvements as well as the total costs required, as shown in Figure 5-16. All the scenarios created for a specific year are shown, and the user can select one or more of the scenarios to look at the details of the scenarios by checking the boxes next to the desired scenarios. As shown in Figure 5-17, the scenario details page shows the details of the selected scenario, grouped by the street sections and listing the selected actions, estimated costs, and benefits. If the user wants to carry out further data manipulation, he can click on the "Export" button and export the scenario to an Excel spreadsheet, as in Figure 5-18.

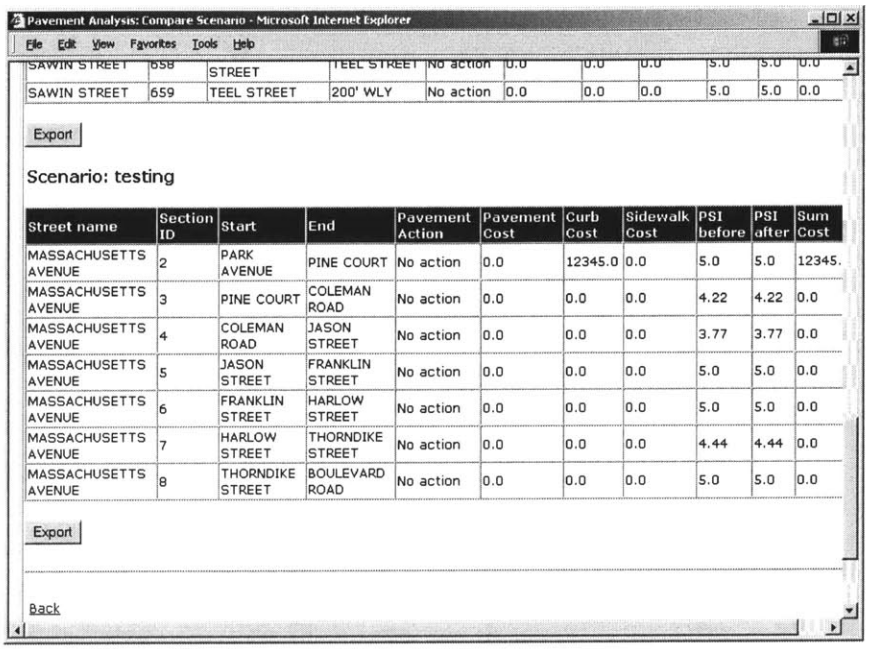


Figure 5-17 Arlington PMIS: Scenario Comparison details page

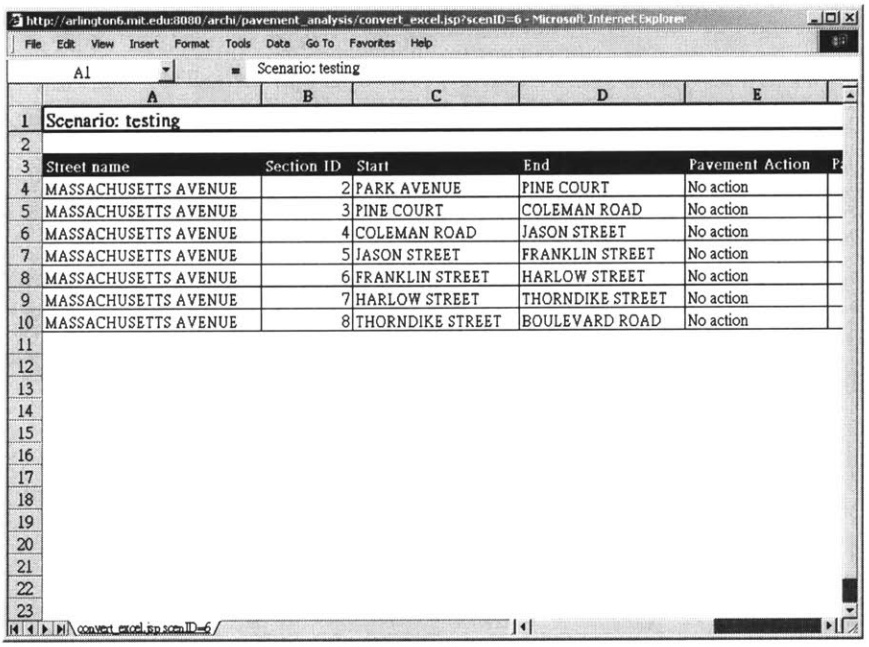


Figure 5-18 Arlington PMIS: Scenario details exported to Excel spreadsheet

6 Further Developments

6.1 System Level

At system level, there is an increased trend to integrate maintenance management with other departmental management functions, and to explore potential applications of new management capabilities and technology. The PMS should be integrated more with other high-level decisions related to capital improvements and operations. It should also be able to have adjustable work plans and schedules to reflect changing conditions, for example, in composition and funding of maintenance program, and federal legislation.

6.2 Application Level – Arlington PMIS

At the application level, the following developments are some of the features that can be integrated in the future to improve the functionality of Arlington PMIS:

- Graphing function that shows the deterioration of pavement sections.
- Implementation of more realistic deterioration models for different pavement sections based on the usage pattern.
- Automated budget optimization based on the IBC method.
- Project scheduling tool for monitoring construction and maintenance activities.

7 References

Cheung, W. (2001). Evaluation of Infrastructure Monitoring System Using PDA and GPS Technologies.

Durongdej, W. (2001). Software Development Process: Web-based Pavement Management System as Case Study.

Haas, R., & Ronald Hudson, W. (1978). Pavement Management Systems. McGraw-Hill.

Hall, M. (2000). Core Servlets and JavaServer Pages. Prentice Hall.

Hunter, J., & Crawford, W. (1998). Java Servlet Programming. O'Reilly.

Loyaerts, Y. (1997). Analysis of Road Database Management Structure. In Advanced Vehicle and Infrastructure Systems: Computer Application, Control and Automation (pp.371 – 390). John Wiley & Sons.

Markow, M.J. (1993). Highway Maintenance and Integrated Management Systems. In Infrastructure Planning and Management (pp.127 – 131). American Society of Civil Engineers.

Mohseni, A., Darter, M.I., & Hall, J.P. (1993). Benefits from Improved Management of Pavement Facilities. In Infrastructure Planning and Management (pp.21 – 25). American Society of Civil Engineers.

Shahin, M.Y. (1994). Pavement Management for Airports, Roads, and Parking Lots. Chapman & Hall.

Yim, W.K. (2001). Data Modeling in a Pavement Management System.

IBM Application Framework for e-business: Web Application Client Programming Model. <http://www-4.ibm.com/software/ebusiness/clientwp.html> (cited 1 May 2001).

IBM Application Framework for e-business: The Web Application Programming Model. <http://www-4.ibm.com/software/ebusiness/pm.html> (cited 1 May 2001).