Mississippi State University
Scholars Junction

Theses and Dissertations | Theses and Dissertations

12-11-2004

GML Represntation for Interoperable Spatial Data Exchange in a Mobile Mapping Application

Venu Madhav Singh Kanaparthy

Follow this and additional works at: https://scholarsjunction.msstate.edu/td

Recommended Citation

Kanaparthy, Venu Madhav Singh, "GML Represntation for Interoperable Spatial Data Exchange in a Mobile Mapping Application" (2004). *Theses and Dissertations*. 2427.
https://scholarsjunction.msstate.edu/td/2427

This Graduate Thesis - Open Access is brought to you for free and open access by the Theses and Dissertations at Scholars Junction. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholars Junction. For more information, please contact scholcomm@msstate.libanswers.com.

GML REPRESNTATION FOR INTEROPERABLE SPATIAL DATA EXCHANGE IN A MOBILE MAPPING APPLICATION

By

Venu M Singh Kanaparthy

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2004

Copyright by

Venu Madhav Singh Kanaparthy

2004

GML REPRESNTATION FOR INTEROPERABLE SPATIAL DATA

EXCHANGE IN A MOBILE MAPPING APPLICATION

By

Venu M Singh Kanaparthy

Approved:

_____	_____
Charles G. O'Hara Associate Professor of GeoResources Institute (Director of Thesis)	Tomasz A. Haupt Associate Professor of Computer Science and Engineering (Academic Advisor)
_____	_____
Roger L. King William L. Giles Distinguished Professor Professor of Electrical and Computer Engineering (Committee Member)	Nicholas H. Younan Professor of Electrical and Computer Engineering (Graduate Coordinator)

_____

Robert P. Taylor
Interim Dean of the College of Engineering

Name: Venu M Singh Kanaparthy

Date of Degree: August 07, 2004

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Tomasz A. Haupt

Title of Study: GML REPRESNTATION FOR INTEROPERABLE SPATIAL DATA EXCHANGE IN A MOBILE MAPPING APPLICATION

Pages in Study: 109

Candidate for Degree of Master of Science

Geographic information is critical to GIS applications located remotely for executing business operations. GIS applications need to interoperate to be able to share information for analysis and decision making process. Heterogeneity and complexity of information models and structures limit the data flow and application interoperation. Advancements in Internet technologies provided new opportunities for delivering spatial information to remote users. However, spatial data delivered is in proprietary structures, limiting the utility to GIS applications. To enable information flow between GIS applications a portable data modeling approach is necessary. However, geographic information is inherently complex to model. A comprehensive and standardized vocabulary to model characteristics of geographic entities is required. Furthermore applications with the need to share information should have an agreement on information structure and content exchanged. This research presents GML representation to provide interoperable spatial data services. The objective is achieved by providing an open framework to model, encode and delivery geographic information. The results of this research show that it is

possible to develop interoperable spatial data services through service oriented architecture.

ACKNOWLEDGMENTS

This work would not have possible without the support and assistance from a lot of people. To begin with, I would like to thank my thesis director, Dr. Charles O'Hara for his constant guidance and support during critical hour of need. I would like to thank major professor Dr. Tomasz Haupt without whom I could not have gained knowledge in web computing technologies. I would like to thank my committee member, Dr. Roger King for his encouragement and support. I would also like to thank technical support at the Engineering Research Center for timely installation of software tools. Finally, I would like to thank my family for providing me opportunity to pursue higher education and friends, who made my stay in Starkville, Mississippi enjoyable.

This publication was developed under Assistance Agreement No.X828421010 awarded by the U.S. Environmental Protection Agency. It has not been formally reviewed by EPA. The views expressed in this document are solely those of recipient and EPA does not endorse any products or commercial services mentioned in this publication.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

LIST OF ACRONYMS

COM	Component Object Model
CORBA	Common Object Request Broker Object
DCOM	Distributed Component Object Model
DIME	Direct Internet Message Encapsulation
DOM	Document Object Model
ESML	Earth Science Markup Language
FGDC	Federal Geographic Data Committee
GDF	Geographic Data Format
GIS	Geographic Information System
GML	Geographic Markup Language
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
IIOP	Inter-Internet Object Protocol
ISO	International Standards Organization
JMS	Java Message Service
ODBC	Open Database Connectivity
OGC	Open GIS Consortium
RMI	Remote Method Invocation
SAIF	Spatial Archive Interchange Format
SAX	Simple API for XML
SDTS	Spatial Data Transfer Standard
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics

VPF	Vector Product Format
UDDI	Universal Data Discovery Interface
URL	Universal Resource Locater
WKB	Well Known Binary
WKT	Well Known Text
XML	Extensible Markup Language
XMML	
W3C	World Wide Web Consortium
WMS	Web Mapping Server
WFS	Web Feature Service
WCS	Web Coverage Service
WSDL	Web Services Description Language

CHAPTER I

INTRODUCTION

1.1 Geographic Information Interoperability

The utility of geographic information in government, agriculture, utilities and transportation organizations has increased significantly. Geographic information is critical to applications located remotely for executing business operations. Heterogeneity and complexity of information models and structures limit the data flow and hence application interoperation. GIS applications need to interoperate to be able to share information for analysis and decision making process. Efforts to provide spatial data interoperability include data converters, spatial data transfer standard (SDTS, DXF and SAIF), direct data access API's, Simple features specifications SQL and COM/OLE and recently web API [3]. Advancements in Internet technologies provided new opportunities for delivering spatial information to remote users through HTTP protocol [9,10,11,12,19], but did not address the interoperability. Spatial data delivered through Internet is limited to static and proprietary structures. To enable information flow between GIS applications a standard and portable data modeling approach is necessary. XML [13,14] fits the equation since it provides portable and extensible information modeling. XML is flexible and extensible data modeling language unlike HTML. Use of

XML for data representation and transport has created new opportunity to transport custom data types over the Internet to user applications. However, geographic information is inherently complex to model. A comprehensive and standardized vocabulary (schema) to model characteristics of geographic entities in XML is required [5,6]. Furthermore applications with the need to share information should have an agreement on information structure and content. This agreement should provide applications knowledge to validate and utilize spatial information. Geographic information services provided are often bound to specific platform, server interface (CORBA, Java RMI, DCOM) and data transport and encoding protocols (HTTP, IIOP) limiting users applications to specific programming language or software package [16,17]. An interoperable spatial data service should provide open interface to access distributed geographic information services from diverse user applications without any restrictions on platform or programming language [1,2,15].

1.2 Web Services

Web Services are self-describing, self-contained software components that can be discovered and invoked over the network. They are built around set of industry standard specifications and protocols such as WSDL, UDDI, SOAP and XML [3]. WS extend the client/server architecture by enabling broader accessibility to diverse applications and reusability of the web components delivering the functionality. Services can range from weather data, sales tax and flight-time providers to computational geo-processing operations. Using WS framework, standalone applications can deliver their functionality to other applications located remotely through an open interface. All public interface

definitions are described in Web Services Description Language (WSDL) [4]. This interface describes the operations an application supports, input and output parameters for each operation, transport mechanism (HTTP, SMTP or JMS) and the web location (URL) to invoke the service. Application delivering services through WS mechanism can be orchestrated into a workflow to achieve an aggregate business objective. Simple Object Access Protocol (SOAP) [5] is central to WS architecture for facilitating information interoperability. It specifies standard envelope structure and set of information encoding rules for transport. XML [6] is markup language formulated by W3C for information representation unlike HTML, which was developed for information presentation. XML is a flexible and extensible data modeling language that is programming language, platform and device neutral language. XML provides features for representing the primitive data types. Custom data type can be built by extending primitive types. With so many features, XML has emerged as data exchange standard on the web. Information represented in XML can be transformed into text, HTML, XML, and SVG using XSLT style sheets. Tools for operating on XML documents are available in almost every programming language and platform.

1.3 Motivation

The motivation of this research is by the requirement of GIS applications to exchange geographic information portably. Geographic information is critical to applications that need access from different locations to achieve business objectives. Access to information created by applications located geographically is essential for decision-making process. Often these applications are developed in different programming

languages and operating on desktop and mobile platforms. Heterogeneity of user applications, data models and structures limits the flow of information and application interoperation. Use of proprietary structures for spatial data transport not only restricts spatial data services access to other GIS applications but also places constraint on software packages and spatial databases of choice for developing the user application itself. Level of knowledge (structural, schematic and semantic) associated with the information exchanged is a measure of interoperability. Geographic information representing real world features is inherently complex. Without a standard data modeling approach, the quality and accuracy of information is not preserved. Lack of standard data representation and exchange standard leads to limited sharing and hence poor utilization of data. Further, the use of web technologies, server interfaces and communication protocols dictate the level of spatial data interoperability [1,2]. Hence, there is a need to model and encode geographic information exchange structure to be interoperable across GIS applications.

1.4 Hypothesis

An interoperable spatial data service should provide information access to GIS applications located geographically. It should provide a standard for spatial data representation, portable transport and simplicity and ubiquity of access. Spatial data representation should provide flexible and extensible data modeling approach, common information exchange structure and explicit knowledge of the data structure exchanged. Information encoded and transported through such a mechanism should be easily accessible to GIS applications through public and extensible services interface or API.

Services delivered should be available to desktop and mobile applications developed in different programming languages on various platforms.

This thesis puts forth the following hypothesis:

- **It is possible to develop Interoperable GML data services using a service-oriented framework.**

This work proposes Interoperable GML data services for providing spatial data services to desktop and mobile platforms. The application framework provides spatial data interoperability through standard data representation, portable data encoding and transport protocol. Through common data representation, the content and structure of the information exchanged is standardized. User applications should be able to query, interpret and validate spatial data from other applications based on common information exchange structure. Geographic Markup Language (GML) [7], Open GIS geographic data encoding standard, is used to represent the geographic information exchange structure. Application schema is developed to model the GIS entities or objects exchanged. Semantics of the information structure utilized to transport spatial data between user applications is standardized through use of SOAP protocol. Use of standard service interface to provide spatial data and processing services increases the service accessibility to user applications. An open services interface is developed to provide users with information related to data structures exchanged, operations supported and location of services. The hypothesis, if realized will provide open framework to deliver spatial data and processing services to user applications in a platform and programming language independent way. Information providers will have a standard and easy approach for modeling and transporting spatial data services over the Internet. User applications can

utilize open source tools to access spatial data and processing services. The scope of this research is limited to providing interoperable access to vector and raster datasets.

1.5 Organization Of This Document

Chapter 2, *Literature Review*, describes the spatial data encoding, storage, transport, integration and interoperability employed by past and current applications with emphasis on geographic information integration. Chapter 3, *The Methodology*, presents the approach taken to validate the hypothesis and evaluation of the approach based on criteria for data interoperability. Chapter 4, *Proposed Test Bed*, presents the description of application workflow, architecture and implementation details. Chapter 5, *Results and Analysis*, will present the results of the implemented test bed and their analysis. Chapter 6, *Future Work*, outlines the future work.

CHAPTER II

LITERATURE REVIEW

2.1 Geographic Information Systems

Geographic Information System is an electronic system for storing, visualizing and analyzing geographic information or spatial data. The system primarily comprises of GIS application and data storage system. GIS applications were desktop-based with tools to view, analyze and make maps. A proprietary file structure or spatial database often stores geographic information. Geographic data is captured in two basic types, vector and raster. Raster data types are image based and store information as array of pixels. Vector data types store information as objects. Each type has specific utility in capturing the real world feature characteristics. Vector data types can model topological relationship among geographic features. Raster datasets represent continuously varying phenomenon such temperature gradient and land cover changes. Raster data types need large computing power and space for processing and storage respectively.

The utility of geographic information in government, agriculture, utilities and transportation organizations has increased significantly. Organizations independently developed information over decades resulting in data islands. Each island of data has its

own data model and structure along with the software packages for analysis.Heterogeneity of data models and formats limited data flow geographic information across organizations and hence the utility. Conversion tools were developed to transport information from one system to another at additional cost. Initiatives for interoperable geographic information exchange resulted in standards such as Spatial Data Transfer Standard (SDTS), Spatial Archive Interchange Format (SAIF) and Geographic Data File (GDF) [3,4,5]. Complexity and ambiguity of these standards limited their use. Innovations in web technologies provided new opportunity to deliver geographic information through Internet to remote users. Through client-server architecture, information provider can service requests from users by transporting data over HTTP protocol. However, information delivered is static (GIF, PNG and JPEG) or proprietary file structure (ESRI, and MapInfo). Open GIS standards such as WMS, WFS and WCS specifications [2] are developed to address spatial data interoperability problem.

2.2 Mobile GIS

Traditionally GIS applications were limited to desktop. New technological innovations introduced mobile devices. As a result, numerous software tools and applications have been developed to support mobile user requirements. Integration of GPS (Global Positioning System) module provides location-based intelligence to mobile applications. Wireless technology has enabled real-time data delivery from users located remotely, triggering the demand for emergency decision-making and resource dispatch applications. Traffic routing, forest fire and chemical spill hazards prevention are such critical applications. However, mobile devices are characterized by limited capabilities

such as low storage, computing power, battery and screen resolution. The connectivity of wireless devices can range from fully connected to intermittent to disconnected modes. Low bandwidth of the wireless connection limits the size of information transported. Moreover the complexity of tasks executed in field environment dictate the design requirements of a mobile application. Increasingly mobile devices are replacing the paper-based approach employed for field operations and data collection. Information gathered from remote mobile applications is valuable in process of decision-making. Mobile applications improved productivity by cutting down the time required to execute a business process or task. They provided efficient and accurate methods for data collection, validation and integration to certain extent. Digital data collection methods ensured quality of geographic information. In GIS context, initially form-based data entry applications without maps were developed. More advanced applications provide the ability to view, edit and create maps on mobile devices extending the desktop GIS to remote locations.

2.3 Spatial Data Transport

Traditionally, GIS systems were confined to desktop and provided tools to visualize and analyze geographic information. The geographic information is often stored in a file based proprietary structure on a standalone machine. Sharing of information was possible through network file system. Advent of Internet technologies provided new opportunities and approaches to deliver geographic information through Web GIS systems [6,7,8,9,10]. Web GIS systems are based on client-server architecture.

2.3.1 Client-Server

Through Web GIS, data providers can serve spatial information could be delivered to remote users over the Internet using HTTP protocol. Server processes basic requests from client applications for data. Advanced servers can process spatial operations such as buffer, mosaicking and overlay. User can access the geospatial information in a web browser from any remote location without the need for intensive training program for utilizing the complex software packages and data. Cost of complex and expensive software and hardware installations can be eliminated. User is relieved from burden of maintaining the datasets. Centralized control will enables changes to the datasets near the providers visible to users immediately. Through web GIS systems data types such vector (ESRI Shape files and DXF) and raster (GIF, TIFF and PNG) could be transported to user applications. The extent of utility of datasets to users depends on the format of delivery. If the datasets are delivered in a format other than required, the user applications needs to convert the dataset into a usable format. Limitation on size of data transported over the Internet exists and depends on the connection speed. Unlike vector types, raster images have large data volume and hence need high speed Internet connection for transport. Raster images provide limited information and interaction capabilities to the user applications, while vector data provides rich features and geometry information. The user can interact with objects in vector data through query, zoom in/out and buffer operations locally without need for server connection. Raster data types do not provide user interaction capabilities such as query and zoom in capabilities. Size of vector datasets increases with the number of geographic features delivered over the web. Methods such as scale dependent transport are possible for huge datasets. Features visible for a given

scale are only transported to user applications to effectively utilizing the Internet bandwidth [11]. Three types of client server architecture are possible based on data type support and functional operations such as spatial and non-spatial query, transformation and rendering delegated between the client and server processes. Data requirements are addressed by providing support for diverse dataset types (image, vector, CAD, GDF and SVG). Functional requirements are addressed by providing support for spatial (buffer, intersect and overlay) and non-spatial operations (feature attribute query).

2.3.1.1 Thin Client

In thin client architecture, server hosts most of the processing operations and data. Users are relieved from cost of software packages and management of datasets. Centralized control of the datasets enables immediate propagation of changes to remote users. Type and format of the dataset delivered from the server can limit the utility to users. Vector datasets such vector (Shape files, VPF and CAD) cannot be delivered to thin client applications. Interaction with the datasets such as zoom and pan operations require server processing. Thin client applications are web browser based applications built with server technologies such as CGI, ASP and JSP.

2.3.1.2 Thick Client

In thick client architecture, the client application provides features such as support for vector and raster dataset types (VPF, ESRI Shape file and SVG,) and local interaction operations with datasets. Thick client are suitable for location-based services or fieldwork applications requiring limited communication with server. Since field based applications

on mobile devices have limited processing power, memory and communication bandwidth. The primary goal is to minimize server-client communication traffic. Clients only communicate with the server to download the datasets and mostly rely on local processing tools and resources for operating on the datasets. To support rich client features, applets or browser with plug-in are necessary. Client application can be developed to communicate for data and processing services through custom protocols such as Java RMI or CORBA IIOP in addition to standard HTTP and CGI. Java applets, HTML based application with Java script and plug-in are examples of thick clients. Users require additional software packages or plug-in and hardware.

2.3.1.3 Intermediate Client

In intermediate client architecture, the processing functions such as data conversion, overlay, transformation, styling and rendering are split among client and server processes depending on user application and business requirements.

2.3.2 Multi-tier

A scaled version of client-server architecture is a multi-tier architecture, where user applications have ubiquitous access to spatial data and processing functions located on geographically distributed servers. In a multi-tier architecture, data and processing functions are separated into a database and application logic tiers respectively. Application tier handles user requests and processing logic. Database tier handles processing of data oriented operations. Distributed object technologies such as CORBA, DCOM and Java RMI can be utilized to provide open interface to access distributed geospatial objects. Limitations such as platform and dependence complexity limit the use

of such distributed object technologies to certain user applications. Advantages of multi-tier architectures include scalability and fault tolerance.

2.4 Spatial Data Interoperability Approaches

2.4.1 Spatial Data Interoperability

Standards to address geographic information interoperability have been developed by various organizations. Information interoperability is the ability of a system or components of a system to provide information portability across multiple applications. Traditionally GIS systems were limited to desktop and could only be utilized by a scientist or trained expert. Spatial data or geographic information is often stored in proprietary format and accessed through a complex and vendor specific GIS software packages. Through this approach only certain GIS software could utilize geographic information. Increase in demand for spatial information by government, public works, agriculture and transportation organizations resulted in need for geographic information interoperability. Open GIS Consortium is an organization of government, private, GIS vendors and industry partners formed to defined requirements and implement open and industry standard specifications. Organizations include ISO, ESRI, W3C and FGDC. The evolution of standards to provide GIS interoperability [1] are described in the table below,

DLG, MOSS, GIRAS	Data Converters
SDTS, SAIF, DXF, GML	Standard interchange formats
VPF, Shape files	Open file formats
ArcSDE API, CAD Reader, ArcSDE CAD Client	Direct Read API
OGC Simple Feature Specification for SQL, COM/OLE and CORBA	Simple feature sharing in DBMS
WMS, WFS, WCS	GIS Web services

Table -1 Interoperability Initiatives

Initial approaches to provide spatial data interoperability resulted in data conversion tools (DLG, MOSS, GIRAS) to Import/Export data from and to different proprietary formats. Data conversion process is expensive and can result in loss of information during translation. The next step was to develop a standard spatial data interchange format such as SDTS, DXF and SAIF. The complexity and loss of information associated with the above interchange formats limited their use. Further open file data structure such as VPF and ESRI shape file [12,13] were developed to provide geographic information interoperability. A limitation such as lack of topological modeling is associated with ESRI shapes files. Till the early 1990 geographic information primarily was stored in file base proprietary formats such as ESRI ArcInfo. Technological advancements in DBMS systems resulted in support for storing geographic information in a database. Vendor products such as ESRI ArcSDE, Oracle Spatial 9i and Informix Spatial Database [14,15] provide support to store spatial data. Spatial databases provided a new opportunity for enterprise wide GIS data storage and access. Application Programming Interface's (API) were solution to the GIS data interoperability. API's such as ArcSDE Java and C were

developed to provide direct data connection to user applications. However interoperability of GIS systems could not be achieved since each spatial database could have different schema structure for storing information. A consensus among organizations such as ISO, OGC and FGDC to share geographic information through spatial data standards resulted in Simple Feature Specification. The specification provides rules to represent and share simple features such as point, line and polygons in spatial databases. Two specifications emerged namely, Simple Features Specification for SQL and OLE/COM [2]. The Simple Features for SQL provides a framework for accessing geometry and functions to access, query and update spatial data. WKB specification, is central to Simple Features Specification, dictates how geometry for simple features should be stored in the database. This specification can be implemented using two approaches namely, SQL 92 implementation of Feature Tables either uses normalized table and standard numeric types to store geometry or access geometry through WKB representation. SQL 92 with Geometry Types provides functions for accessing geometry in WKB or WKT and functions to access, manipulate and query spatial datasets. Not many vendors implemented Simple Feature for SQL specification except few such as Oracle, Informix and IBM DB2. Microsoft's ODBC and OLEDB provide OLE interfaces to uniformly access multiple data sources. Increased implementation of OLEDB to provide universal data access to many database vendors resulted in Simple Feature Specification of COM/OLE. This specification provides additional COM interfaces for accessing geometries in WKB, along with standard OLEDB interfaces. This specification has been adopted by number of vendors such as ESRI, AutoDesk and Cadcorp. Simple Features for CORBA is developed to provide functionality to access and manipulate simple geographic features. CORBA specification for simple features has been designed

to provide access to geospatial data and applications through a language, operating system and vendor independent approach. This specification consists of two modules, Feature and Geometry modules. Feature model is used to create, access and query simple geospatial features and feature collections. Geometry model provides set of interfaces for accessing the geometries. The CORBA specification for Simple feature has been designed to interoperate with SQL and OLE/COM specifications. Advancement in Internet technologies and standards enabled new opportunities to provide Internet based spatial data interoperability. OGC Web Map Service (WMS) specification is developed first to provide image service to user through the web. WMS provides georeferenced maps over the Internet in GIF, JPEG, TIFF, PNG and SVG formats. The specification provides communications semantics for request and response operations such as GetCapabilities, GetMap and GetFeature. Vendors implementing a specification need to support basic set of operations such as GetCapabilites and GetMap for OGC WMS. Web Feature Service specification provides exclusively vector data services over the Internet. Like WMS, WFS defines request and response communication semantics. Vendor implementing WFS specification must use GML for representing geographic information and support basic operations such as GetCapabilities, DescribeFeature and GetFeature. WFS specification provides support for spatial and non-spatial query operations. Other OGC specifications such as Web Converge Service and Web Catalog Service specification have been developed to provide raster dataset and data source cataloging services respectively. OGC Web Services Initiative is the latest effort to provide distributed geoprocessing services through service-oriented framework.

2.4.2 XML for Information Interoperability

XML is text based, platform and programming language independent markup language developed by W3C for data modeling and storage. XML stores information in plain text allowing applications and humans to easily interpret. XML is human and machine readable, extensible and portable language. DOM and SAX API's provide methods to access and manipulate XML data in terms of node, tree and graph representation. Query language such as XQuery has been developed to query XML documents. Database vendors such as Microsoft and Oracle are providing support for XML output. XML separates data representation from presentation. Multiple views can be generated by apply XSL transformation to same XML data. XML Schema specification provides semantics to the information stored in XML structure. Schema provides features to model characteristics of simple and complex data types. Complex data types can be built by extending the simple types. The content (default, fixed) and range of values for a data type can be specified explicitly. XML schema supports object-oriented features such as inheritance and encapsulation [35]. From data sharing perspective, XML schema can be defined to create a common and neutral data exchange model. Data model will provide data structure information to user applications for validation and use. Organizations interested in sharing geospatial information can define a standard for data exchange. The standard will contain feature attribute and geometry characteristics of geographic features that need to be shared [36,37,38]. To share information across heterogeneous databases, integrators are required to map local database schema to common schema.

2.5 Spatial Data Interoperability Criteria

Methods or architecture providing geospatial information interoperability can be evaluated based on following criteria [26],

2.5.1 Spatial Data Semantics

This classification is based on the level of knowledge associated with a dataset. Higher level of knowledge representation of a dataset provides maximum interoperability dataset among diverse applications. At structural level, knowledge about the physical organization of dataset is available. User applications will have information on structure of dataset such as array or list. This level does not provide knowledge to user applications about what data is encoded or represented in the data structure. E.g. array of values can mean pixel values of an image or temperature gradient. At schematic level, knowledge about the structure and characteristics of datasets are available. Characteristics are dataset attribute information that can be queried. At the top is semantic level that provides common vocabulary to model feature characteristics of the datasets across different GIS systems. Common vocabulary or grammar is called ontology. Organizations interested in data sharing can create ontologies tailored to their application domain. Semantic level is the highest level of dataset integration possible.

2.5.2 Spatial Data Representation

Heterogeneity of data models and structures is due to organizational data requirements and lack of data standards. Heterogeneity of data models limit interoperability for geographic information across applications or GIS systems. Data exchange standards

such as SDTS, SAIF and GDF are developed to address spatial data interoperability problem. These standards formalize the information exchange structure and content. Complexity and loss of information associated with theses models limited their use. GML is the recent (2002) data exchange standard developed by OGC. However, to share geographic information across multiple applications uniformly a common data model is necessary. Use of a standard data model provides spatial data interoperability across user applications. Since schema structure for individual databases is often different, data providers need to develop mapping from local spatial database to a common and standard schema.

2.5.3 Spatial Data Format

Spatial data delivered in formats usable to client applications provides greater utility. Data conversion is necessary if the service provider's data structure does not match the user application requirements. Since user applications are developed using free or proprietary GIS software packages that may not support all the geographic data formats. Either user applications or the data provider need to convert the data into a usable structure. An interoperable spatial data service should support data delivery in diverse formats (both raster and vector). Tools [Spatial Direct, FME] have been developed that convert data into format desirable (such as GIF, MapInfo MIF, GDF, ESRI Shape files, ArcInfo, AutoCad DXF and GML) to user applications.

2.5.4 Service Interface

Interoperable service interface should provide public, standard and extensible server interface [23]. Distributed computing technologies such as CORBA, DCOM and Java

RMI provide ability to publish and access geoprocessing functions and spatial data. CORBA provides flexibility of implementation in different programming languages such as Java and C++. Client applications could interact with CORBA objects irrespective of the platform, language and data model involved in implementation of the objects. DCOM is Microsoft based distributed computing technology limited to windows platform and CORBA is complex to implement. Java RMI succeeded in providing distributed computing solution through easy to use and platform neutral solution. Java RMI requires server and clients implemented in Java language. Server side technologies such as Java Servlets, Java Server Pages, CGI and Active Server pages provide dynamic processing and presentation of information to clients through web-enabled browsers. These technologies use HTTP for data transport across the Internet. Standard initiatives such as WMS and WFS by OGC provide geographic images and feature data service to remote applications by leveraging web technologies. Use of standard specifications for implementing geospatial data services provides interoperability. Latest initiative to provide spatial data interoperability is to provide flexible and extensible service interface using web services standards. SOAP API's are developed that can provide user applications to quickly leverage the geospatial data and processing services.

2.6 Geographic Markup Language

Geographic Markup Language (GML) [21] is an XML encoded language for modeling the structure and content of geographic information. It defines vocabularies for representing geographic features characteristics such as topology, direction, elevation and units. GML is an OGC (Open GIS Consortium) data exchange standard, primarily

targeted to address geographic information interoperability among GIS applications. GML is used to model, encode and share vector datasets. GML is an open source data exchange standard and hence freely available to everyone. GML encoded information is portable and can be transported over the Internet to diverse applications on desktop and mobile platforms. Users of GML data do not require investment in complex and expensive software such as Intergraph GeoMedia Viewer, ESRI ArcIMS. An open source map viewer such as Map Server can be utilized to create maps from GML data. Information encoded in GML document is in plain text format and hence can be converted and integrated with other formats. Increasingly large numbers of GIS vendors such as ESRI, Intergraph, and FME are supporting GML data exchange and storage. The cost associated with providing GML based information integration approach is low. Since GML uses XML encoding, it inherits all the features such as extensibility, portability and human readability. XML provides extensibility by enabling user defined custom data types. XML is not dependent on specific platform, programming language or a device. Diverse software tools are available for reading and writing XML data in different languages.

Geographic features are encoded as collection of feature of certain shape type such as point, curve or surface [21] in a GML document. GML provides user applications with feature level access to data layers. This will facilitate query and extraction of subset of features required instead of accessing all the layer features. Feature level transactions such as insert, update and deletion operations can be executed on GML data. Transactional operations are restricted to authorized users. GML schemas provide support for XLink and XPointer mechanism that will allow features from remote data

sources to be integrated into a GML document. The result dataset for a query request with certain criteria may originate from multiple data sources. GML provides integration of data from multiple data sources into a single result dataset enabling real-time integration of geographic information. Since GML is an XML based markup language, transformations can be applied to datasets. Each transformation can be tailored to created customized view of the GML encoded dataset. GML delivered to clients can be converted into different usable formats such as SVG, ESRI Shape files. Tools for creating, transforming and editing are available at no cost in almost every programming language. Open source API's are available freely for manipulating GML documents. Additionally, XLST processors can be created to create SVG maps from GML documents. The size of the GML document increases with number of geographic features encoded. This can be problem for mobile applications with limited computing and memory resources. Compression techniques are required when transporting data to mobile user applications. Since GML offers feature level access to geographic data, queries to access subset of features in a datasets will provide faster transport to mobile applications. GML provides base schemas to model attribute and geometry of geographic features such as roads, rivers and parcels. Schemas provide support to model complex features, feature collections and associations. GML schemas can be extended to model domain specific objects characteristics. Wastewater network features such as sewer lines and manholes can be modeled from base schema. At higher level, application schema can be developed by extending base schema. Application objects or entities such as intersection, drill hole and work order can be modeled. The application schema represents the structure or placeholder for data that will be populated during the data exchange operation. The application schema is used by organization or applications interested in sharing

geographic information. Further communities interested in sharing geographic information in GML can custom develop GML schemas based on data requirements. Since major data providers are increasingly adopting GML, use of GML to model the geographic information structure will provide interoperability across multiple GIS systems. However, GML does not provide semantic interoperability. Since data providers can model features such as lakes or rivers through common exchange GML structure, but might identify the features with different names. GML encoded spatial information can be parsed using DOM or SAX based parsers. DOM based approach creates in-memory tree structure representation of GML data for parsing. DOM based parsing of GML information is more memory and processor intensive since the size of the GML data increases drastically with number of features. SAX approach provides event based parsing of GML data. It provides callbacks to applications to handle specific events such as begin or end of a XML element. Hence, SAX approach is suitable to parse GML documents but require more effort from application developers to utilized callbacks to transform GML data. GML data can be transformed into other formats such as SVG, PNG, GIF and ESRI Shape files using open source XML parsers such as Xalan, and Batik.

2.7 Web Services

2.7.1 Distributed Computing Architecture

Distributed computing technologies such as Microsoft's DCOM, OMG CORBA and Java RMI [22] provide remote and transparent access distributed objects. CORBA interfaces can be defined in programming language neutral language such as IDL and implemented

in languages such as Java and C++. ORB handles all the marshalling and unmarshalling of CORBA objects are transported using IIOP protocol. Client applications could interact with CORBA objects irrespective of the platform and language. CORBA specification is complex to implement. DCOM is Microsoft technology for accessing distributed remote objects. Access to remote objects using DCOM is limited to applications operating on windows platform. Java based distributed computing solution, Remote Method Invocation, provides easy to use and cross platform compatible solution to remote Java applications and applets. But client and server applications have to be in Java language and cannot interoperate with application developed in other programming language.

2.7.2 Service Oriented Architecture

Web services have been created to solve the problem of interoperability among applications by providing a framework of searchable, reusable, interoperating software components or services. WS provides applications or systems interested in interoperation with communication semantics, transport protocol and open interface. The communication semantics specify XML message structure that represents the data exchanged between any two applications. Data exchange operations can be specified independent of the underlying transport protocol. WS provides flexibility to use plug-in multi transports such as HTTP, SOAP and JMS for data transfer. Web services are built around set of industry standard protocols such as WSDL, SOAP, XML and UDDI [16].

2.7.3 Web Services Description Language

WSDL [17] is an XML language for describing a web service in a portable approach. It describes operations, input and output parameters for each operation, transport

mechanism and the web location to invoke the service. Different parts of the WSDL document that describe a complete web service are Types, Message, PortType, Binding and location. Types part represents data types that can be transported to other applications. Schema documents containing both primitive and complex data types from different namespace can be included. Message part represents the structure of the message and types of data exchanged. Messages often represent request and response structures and are associated with a certain operation. E.g. getImage operation can have input message that specifies the image width, height and extents and output message with the image location. Figure 1 shows XML request for Image.

Input Message

<**getImageInput**>

<parameters **width**="**500**" height="**500**"

Xmin="**-122.524464**" Ymin="**37.690002**" Xmax="**-122.349621**" Ymax="**37.834193**" />

</**getImageInput**>

Output Message

<**getImageOutput**>

<imgLoc>**http://venu-pc.erc.msstate.edu/output/basemap.jpg**</imgLoc>

</**getImageOutput**>

Figure 1 Image Request

PortType part represents operations supported by a web service. Multiple PortTypes can exist within a single web service, where each PortType groups related operations. Each operation part will have input and output part representing the input and output messages for the specific operation. Binding part binds the PortType and hence the operations to

specific transport protocol and a web location. Supported protocols are HTTP GET/POST, SOAP, SMTP and JMS.

2.7.4 Simple Object Access Protocol

SOAP [18] is XML based open standard for encoding data for transport over the Internet. HTTP enabled computers to network and publish web pages for presenting the data in HTML. Limited data types such as text, images and multimedia files can be represented in HTML. SOAP specifies the data encoding for transporting information between applications in XML. Explicit mapping from data objects represented in a programming language to XML types is provided. Since XML provides a platform and device independent approach to represent information, SOAP encoded information is portable across applications. SOAP is an application level protocol like HTTP that can be implemented over TCP. Since firewalls and proxies on the Internet do not limit HTTP traffic, SOAP is implemented over HTTP. SOAP specifies the standard structure and content for messages exchanged between applications. Figure 2 shows SOAP request for an image.

```
<?xml version='1.0' encoding='UTF-8' ?>
  <s:Envelope>
    <s:Body>
    <e:GetImage xmlns='http://www.mynamespace.com'>
        <e:parameters width='500' height='350'
                      minx="-122.524464" miny="37.690002"
                      maxx="-122.349621" maxy="37.834193"
        </e:parameters>
   <e:GetImage>
  </s:Body>
</s:Envelope>
```

Figure 2 GetImage SOAP Request

A SOAP message consists of an envelope containing different parts such as body and header. The SOAP Body part contains XML payload representing the data. The Header part is used to specify application specific information such as transaction, routing and encryption that needs to be included for processing the SOAP message. Additional, Fault part is used to notify a user application in the event of failure of certain operation provided by a web service. SOAP messages have two different styles, Document and RPC. In Document style, there is no predefined structure for data in the Body part. The structure can be defined by creating schema documents. XML documents such as processing orders, product information and business documents can be transported through this approach. In RPC style, Body part contains the name of the method or remote procedure and an element for each parameter of the procedure. Data serialized into SOAP body is in two formats, encoded and literal. In encoded format data is serialized using encoding rules in the SOAP specification. The specification defines how objects and arrays can be serialized. Data is exchange as objects and structures. In literal format data is serialized according to the schema definitions. Document/literal approach,

using document style with literal serializing, is more practical for applications exchanging XML documents with a pre-agreed structure. Transporting binary data such as images, videos and graphic files using SOAP require custom encoding method. Base64 encoding scheme encode bits to character representation suitable for small binary data. Base64 encoding increases the size of the payload and hence needs intensive computing resources. SOAP with Attachments (SAAJ) specification defines how SOAP messages can transport arbitrary binary data in one document retaining the native format of data. MIME (Multipart/related message extension) represents several parts with a document. Each part can be a binary content or a file with a unique content identified. The identifier is used by client applications to locate binary data within the SOAP message. MIME suffers from buffering and searching for message delimiters. DIME (Direct Internet Message Encapsulation) [23] is approach for encapsulation data payloads into single message. DIME approach is similar to MIME but the with less computing overhead associated with processing data. Use of DIME to transport binary images is demonstrated in this work.

CHAPTER III

METHODOLOGY

This section will describe methods developed to achieve spatial data interoperability across desktop and mobile applications. The description of components of the proposed architecture is provided and methods are evaluated based on criteria for interoperable data solution. Information interoperability is the ability of a system or components of a system to provide information portability across multiple applications. Proprietary of data structures, models and service interfaces limit the utility of geographic information to user applications. This thesis proposes following methods to provide spatial data interoperability across desktop and mobile applications.

The interoperable methods depend on the following approaches:

- Use of standard data model
- Use of flexible, accessible and extensible service description interface
- Use of portable and extensible data encoding and transport protocol

3.1 Common Data Model

Use of standard data model to represent geographic information exchanged between applications enables improved sharing and hence interoperability. A standard data model

will provide users knowledge about dataset representation and data providers to serve geographic information. Geographic information is inherently rich and complex to model. Standards such as SDTS, SAIF and DXF have been developed in the past to provide offline data interoperability. OGC GML standard is the latest developed to provide online access to spatial data from multiple data sources. Applications [ref] have been developed that utilize XML to model and share geographic information. These applications need to custom defined comprehensive grammar or schema to model unique and complex characteristics of data shared. Further it is difficult to get data users and providers to agree on custom developed grammar for data modeling. Since GML is an OGC standard, geographic information representation built over GML schemas provide common data model to large group of data users and providers. To utilize geographic information applications need to know the structure of data exchanged. GML application schema provides knowledge about the structural and schematic information. UML diagram representing manhole and sewmain features is show in figure 3.

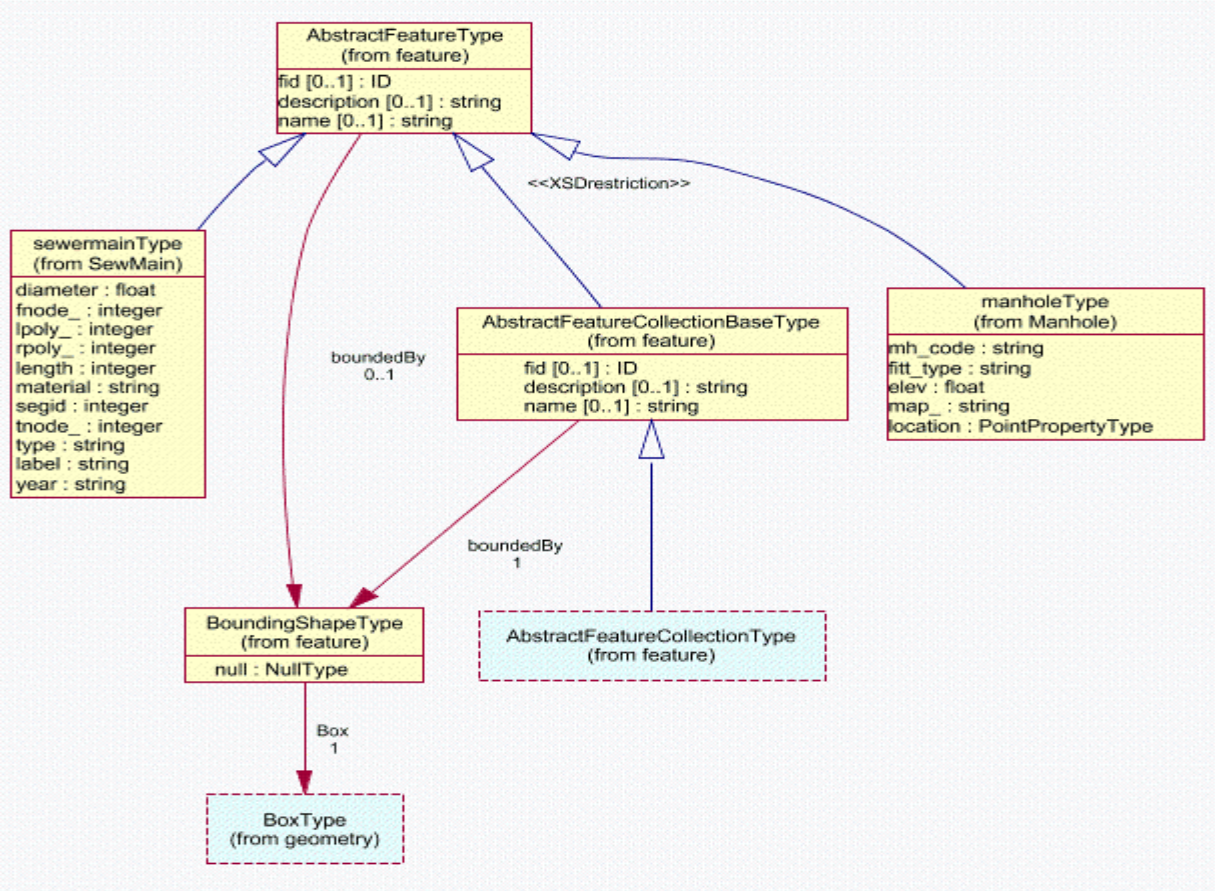

Figure 3 UML Diagram

Figure 4 show XML schema representation of manhole feature. “ManholeFeatureType” is a complex XML data type that defines properties unique to a manhole and inherits all the properties such as geometry from GML base class, “AbstractFeatureType”.

```
<xs:complexType name="manholeFeatureType">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element ref="gml:location"/>
          <xs:element name="mh_code" type="xs:string"/>
          <xs:element name="fitt_type" type="xs:string"/>
          <xs:element name="elev" type="xs:float"/>
          <xs:element name="map_" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Figure 4 Manhole Feature Schema

Figure 5 represents XML instance document with manhole information.

```
<gml:manholeFeatureType i:type="gml:manholeFeatureType">
        <gml:boundedBy i:nil="true"/>
        <gml:description i:nil="true"/>
        <gml:fid i:type="d:string">607</gml:fid>
        <gml:name i:nil="true"/>
        <gml:elev i:type="d:float">102.0</gml:elev>
        <gml:fitt_type i:type="d:string">S</gml:fitt_type>
        <gml:location i:type="gml:PointPropertyType">
          <gml:Point i:type="gml:PointType">
            <gml:coord i:type="gml:CoordType">
              <gml:X i:type="d:decimal">1603355.875</gml:X>
              <gml:Y i:type="d:decimal">647000</gml:Y>
              <gml:Z i:nil="true"/>
            </gml:coord>
          </gml:Point>
        </gml:location>
        <gml:map_ i:type="d:string">8319SW</gml:map_>
        <gml:mh_code i:type="d:string">S05908</gml:mh_code>
    </gml:manholeFeatureType>
```

Figure 5 Manhole Data Instance

GML schema enables data providers with flexibility to build application schema documents that represent characteristics of geographic features related to certain domain. Application schemas are built by extending the base schema from the GML specification. User can specify constraints on type, range and default values for feature attributes. Application domains such as transportation, wastewater or geology [36,37,38] can be modeled. E.g. Transportation objects can be modeled from road, bridge and railroad objects. Communities interested in sharing spatial data can represent domain specific data structures through GML schemas. Users interested in utilizing the dataset share the application schema documents. Use of such a common data standard to represent geographic information will enable maximum spatial data interoperability.

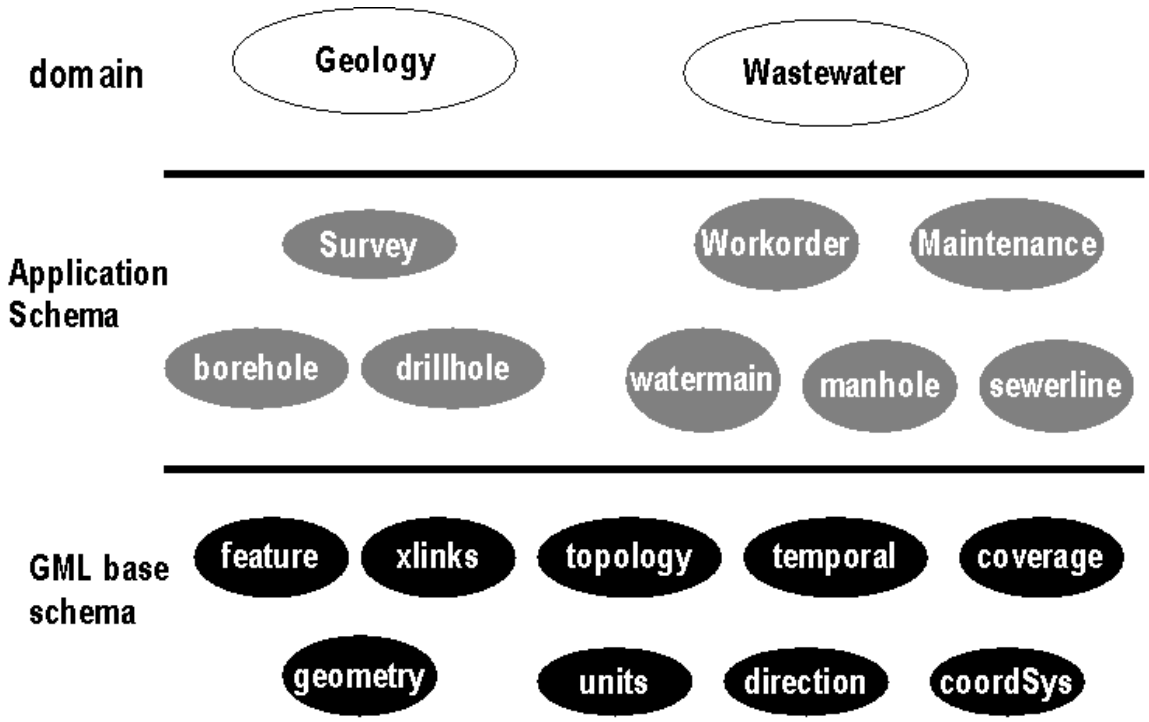

Figure 6 GML Spatial Data Object Modeling

Increasingly major GIS vendors such as OGC, ESRI, Intergraph, Oracle and FME are supporting GML data representation. Thus GML application schemas enable schematic level interoperability by providing user applications with the knowledge on the data structure and attributes. User applications can interpret and validate the structure of the dataset exchanged based on schema information provided. Further spatial and non-spatial operations such as buffer, intersect and subset are possible.

3.2 Portable and Extensible Data Encoding

Traditionally GIS datasets have been developed by different organizations in different formats resulting in data islands. Data sharing was possible through direct data conversion, open file formats and API's. To use proprietary formats user applications required data conversion or use of software packages that could interpret certain or collection of data formats at additional cost. Revolution of Internet provided opportunity to deliver share spatial information through web. HTML is used primarily to publish data on the Internet. HTML pages are static and provide limited support for text, pictures and multimedia content. XML unlike HTML offers flexible and extensible modeling features. XML can be used to transport geographic information in a portable and extensible way. To exchange data in XML, applications need an approach to serialize data types before transporting and de-serialize at the receiving end to use. Without standard mapping from applications data types to XML types, Serializing and de-serializing logic needs to be embedded into client and server applications. This thesis employs SOAP to encode the spatial information for transport over the Internet. SOAP messages contain XML

representation of spatial objects such as manholes and sewer lines [B2, B4]. The figure 7 represents the SOAP message with XML encoded manhole feature information.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
  <pointFeature>
     <multiLocation xmlns="http://www.opengis.net/gml">
          <MultiPoint>
            <pointMember>
              <coord>1602858.25,648012.5625</coord>
               <fid>503</fid>
               <mh_id>U0033-S11</mh_id>
               <fitt_type>S</fitt_type>
               <elev>97.53</elev>
               <map_>8319SW</map_>
             </pointMember>
            <pointMember>
            <coord>1603171.828418,647936.206351</coord>
               <fid>509</fid>
               <mh_id>S08402</mh_id>
               <fitt_type>S</fitt_type>
               <elev>0</elev>
               <map_>8319SW</map_>
             </pointMember>
            <pointMember>
            <coord>1603675.864721,647920.721049</coord>
               <fid>512</fid>
               <mh_id>S08416</mh_id>
               <fitt_type>S</fitt_type>
               <elev>0</elev>
               <map_>8319SW</map_>
             </pointMember>
          </MultiPoint>
         </multiLocation>
       </pointFeature>
  </soap:Body>
 </soap:Envelope>
```

Figure 7 SOAP Message with XML Encoded Spatial Entities

3.3 Open Service Interface

Description of service interfaces to datasets independent of underlying communication protocol and data encoding provides transparent accessibility. Past efforts to provide access to dataset include proprietary API's. CORBA and DCOM interfaces for delivering spatial data services are complex and limited to certain platforms respectively. CORBA provide platform and programming language neutral access to distributed objects. Limitation such as access to CORBA objects past the firewall exists. This thesis proposes use of web service description language to openly describe spatial data services. WSDL is an XML based language to describe web services in a platform and transport independent way. WSDL provides publicly accessible service interface to spatial data services and flexibility to plug-in different transport protocols such as HTTP, SOAP or JMS. Services defined through WSDL are easily extensible without requiring major changes to user applications. WSDL provides vocabulary to define operations and data types supported by a service along with the transport protocols and service location. User applications interested in accessing spatial services will use the WSDL document to create requests for required services. User sends a SOAP request message to the service. The service processes the SOAP message, performs operations and returns the result with the SOAP response message. Spatial services can be extended to add support for new operations and data types.

3.4 Architecture Design

The design decisions of the architecture were driven by requirement to provide spatial data to desktop and mobile platforms. Architecture solution implemented provides

interoperable geospatial data and processing services. Interoperable methods for spatial information interoperability are based on web services architecture. The architecture is a multi-tier solution with backend spatial database, middleware service framework and front-end user applications. Following paragraphs describes various distributed computing technologies, data exchange formats and databases available for implementation and approach taken in this thesis.

3.4.1 Service Interface

WSDL specification describes the open service interface to access distributed services by user applications. Other distributed computing technologies such as CORBA, DCOM and RMI are candidates for providing distributed services. The table below provides comparison of DCE's (Distributed Computing Environments),

DCE	Open & Public	Cross platform compatible	Mature	Language Dependency	Tools availability	Performance	Multi transport protocol
WS	Yes	Yes	Yes	No	Yes	No	Yes
CORBA	Yes	Yes	Yes	No	Yes	Yes	Yes
DCOM	Yes	No	No	Yes	Yes	No	No
RMI	Yes	Yes	Yes	Yes	Yes	Yes	No

Table - 2 Distributed Computing Technologies

All the DCE (WS, CORBA and RMI) are mature, public and open industry standards except DCOM, which is not a mature standard. WS and CORBA provide mapping of interface definitions to different programming languages. CORBA is ideally suited to provide interaction between legacy systems. Software products and tools are available in

popular programming language such as Java, C++ and Visual Basic for implementing the DCE client-server components. Most of the DCE's in the table 2 are OS independent except for DCOM that is windows platform dependent. RMI is bound to Java and hence language dependent. Performance of WS is limited due to XML processing required at server and client locations for serializing and de-serializing the objects to and from XML. CORBA services provide higher performance compared to WS, DCOM and RMI.

WS provide support for multi-transport protocols such as HTTP, SOAP and JMS while CORBA provides TCP and HTTP support. RMI uses native JRMP (Java Remote Method Protocol) for transport. Unlike HTTP that supports limited data types, SOAP provides support for transporting custom data types. JMS provide asynchronous transport providing scalability and throughput while reducing the latency. SOAP is application protocol that can be implemented over HTTP and JMS. WMS and WFS are open GIS consortium standards for delivering image and feature data over the Internet using HTTP protocol. WFS implementation must use support set of basic operations (GetCapabilities, GetFeature and DescribeFeatureType) and GML to model and encode data exchanged. This thesis uses GML to model and transport geographic information like WFS but utilizes SOAP to transport information to user applications. WS framework is used to provide spatial data and processing services to user applications.

3.4.2 SOAP Toolkits

Java and .NET are major platforms where majority of WS are implemented. The spatial data services implemented in thesis provide Java and .NET services implementation. SOAP packages perform the task of serializing and de-serializing SOAP message from

application objects to XML types and vice-versa. The table below provides SOAP packages for major platforms,

SOAP Toolkit	Public	Mature	Platform
WASP	Yes	Yes	Windows, Solaris
Glue	Yes	Yes	Windows, Solaris
Axis	Yes	Yes	Windows, Solaris
.NET SDK	Yes	Yes	Windows
MS SOAP	Yes	No	Windows
PocketSOAP	Yes	Yes	Windows CE
PSOAP	Yes	No	Windows, Linux CE

Table - 3 SOAP Tookits

Popular SOAP packages available for Java are Axis, WASP and Glue. Likewise of .NET platform Microsoft .NET SDK that provides flexibility to program services in multiple languages such as C#, VB.NET and C++. WASP and .NET SDK are utilized to implement Java and .NET services for desktop and mobile clients. Mobile client utilizes PocketSOAP to produce and consume SOAP message from spatial data services.

3.4.3 Data Exchange Format

Data exchange format used to sharing spatial data between applications is measure of data interoperability. Use of proprietary data structures limits the use. The table 4 provides some of common data exchange formats in GIS.

Data exchange format	Public	OGC Standard	Level of Knowledge	Dynamic data integration
ESRI Shapefiles	Yes	No	Schematic	No
GML	Yes	Yes	Schematic	Yes
XML	Yes	No	Schematic	Yes
Raster/grid (GIF/JPG/TIFF)	Yes	No	Structural	No
DXF	Yes	No	Schematic	No
ESRI ArcInfo	No	No	Schematic	No

Table - 4 Common GIS Data Exchange Types

ESRI Shapefiles, ArcInfo files are proprietary formats from ESRI. GML and XML are platform neutral open standards for representing information as text. DXF is data exchange format for CAD files. Most of the data types except XML and GML formats do not provide dynamic integration of data from multiple data sources. XML based formats provide explicit knowledge thorough schema. A comprehensive and standard vocabulary in XML is required to model geographic entities. Raster data types such as coverages and grid provide structural knowledge and do not provide schematic information. This thesis uses GML as data exchange structure between server and client components. GML schema already provides structure to model geographic entities but to be able to exchange data between applications a mechanism for transporting GML encoded data is required.

3.4.4 Database

Database is a critical data storage component of multi-tier systems. The table 5 provides databases considered.

DBMS	Public	Spatially enabled ?	Platform dependency	Spatial operations
PostgreSQL	Yes	Yes	No	Yes
MySQL	Yes	Yes	No	No
SQLServer	No	No	Yes	No
Oracle	No	Yes	No	Yes

Table - 5 Databases

MySQL and Oracle are candidates but Oracle is proprietary and MySQL does not support spatial operations. SQL Server is Microsoft based proprietary DBMS bound to windows platform. PostgreSQL is used in this thesis to store spatial data types. The database supports spatial operations such as buffer, intersect and re-projection. The table below shows the summary of server and client components chosen for implementation in this thesis.

Components	Software Products/Public	Platform	SOAP Packages
Server	WASP Java - Public .NET - Proprietary	Windows XP	WASP API .NET SDK
Desktop	Map Objects - Proprietary	Windows XP	WASP API
Mobile	ArcPad Proprietary	Windows CE	Pocket SOAP
Database	PostgreSQL – Public	Windows XP	NA

Table - 6 Summary of Components

3.4.5 Architecture

Service framework describes geospatial web services using web service description language. The service description imports GML schema and binds spatial and non-spatial operations with spatial data types exchanged. GML application schema models the feature attribute and geometry characteristics of spatial objects transported to user applications. The user application analyze WSDL document to identify the spatial or non-

spatial operations and create SOAP requests to the web server. The web services processes the requests against the spatial database and returns the SOAP response to user applications. Custom protocols such as ArcXML, JDBC are utilized by web services to extract data from different data sources.

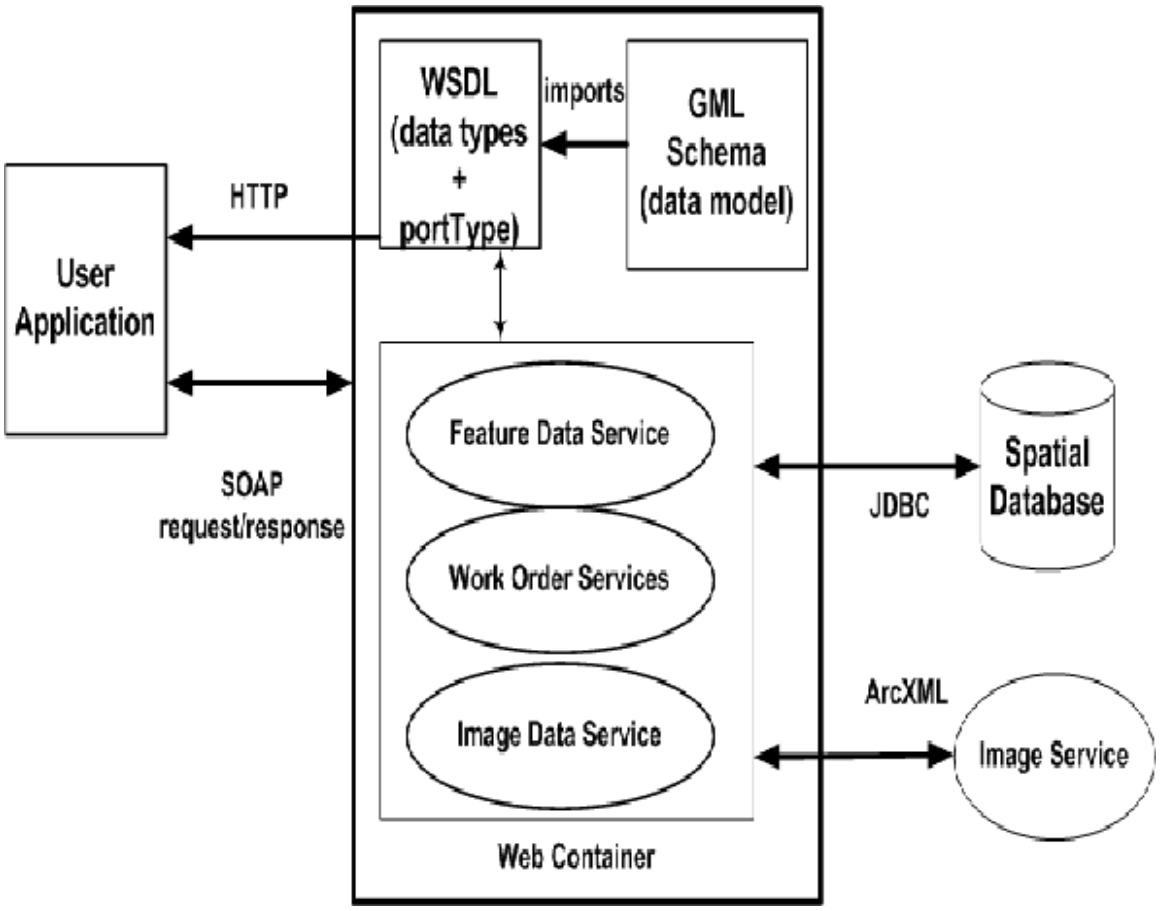

Figure 8 Component Diagram

3.5 Evaluation Criteria for Interoperable Data Services

3.5.1 Data Modeling and Encoding

A common data model is necessary to enable spatial data interoperability across users. The implementation provides GML application schema to model the geographic data exchanged. Application schemas representing the feature geometry and attributes of the

spatial data is developed by extended base GML schema. The implementation encodes spatial datasets as GML document and utilizes SOAP protocol to transport over HTTP to user applications. Information encoded and transported in XML based SOAP message is available to applications on desktop, Internet and mobile platforms.

3.5.2 Data Semantics

The level of data interoperability depends on the extent of knowledge available to user about a dataset [26]. Three levels of data interoperability are possible based on the knowledge about structural, schematic and semantic aspects of a dataset. This implementation provides schematic level interoperability. The structure and content of information exchanged is available to user applications as schema document. The schema document will enable the user applications to interpret and validate the information structure.

3.5.3 Data Conversion and Types

An interoperable data service should deliver data in a format usable to majority of users without explicit conversion. The spatial services implemented deliver spatial data in GML and JPEG representations. The user applications convert GML encoded spatial information into usable structure.

3.5.4 Service Extensibility and Accessibility

The implementation provides spatial data delivery service. Request for specific dataset is processed against the spatial database and GML encoded vector layer is delivered. Other services such as feature redline, work creation, download and update are implemented in

context to providing wastewater work management operations. The implementation uses web services framework for providing spatial data delivery and processing. The implemented services are publicly available to users applications. The services are exposed through web services description language document. The document is both machine and human readable. Services are accessed by desktop and mobile applications through free and publicly available SOAP APIs such as WASP, MS SOAP Toolkit, .NET, Glue, Pocket SOAP, kSOAP and Apache Axis. Implemented web services can be easily extended by adding support for new data types and processing functions. The changes to the services implementation are transparent to user applications.

CHAPTER IV

IMPLEMENTATION

4.1 Testbed Framework

This section describes software components of architecture and their implementation details. Septic system failures are significant problem in fast growing rural areas. A majority of septic systems are greater than 30 years old, over 25 percent are in some sort of failure, and 10 percent overflow on an annual basis. Current methods are inadequate for mapping and managing resources as well as tracking problems and planning appropriate and effective solutions. New technologies must be identified and solutions implemented to provide information and work management solutions to address the existing problems. Wastewater services are primarily maintenance, inspection and emergency response operations. Field inspectors use paper-based maps and work orders for locating housing facilities (parcels), sewer features (such as sewer lines, manholes, meters, pumping stations) and execute tasks in the field. Completed paper work orders are finally updated to spreadsheet or database through manual data entry. Current approaches are inefficient and not cost effective for achieving wastewater work management objectives. Paper-based data collection methods introduce data inconsistency and delays in work order execution due to manual data entry methods. Changes to spatial data in the field are difficult to incorporate into paper maps.

4.1.1 Business Requirements

Web service framework should provide infrastructure and tools to execute wastewater work management operations efficiently at office and field.

4.1.2 User Requirements

- Supervisor should be able to retrieve vector and raster data layers for a specified spatial extent from a remote spatial database.
- Supervisor should be able to view and query parcels and sewer features of the complaint area.
- Supervisor should be able to issue a work order by creating task description, date to complete, inspector's name and support vector data layers.
- Field inspector should be able to query and download work order and execute the tasks in the field remotely through the Internet.
- Field inspector should be able to execute tasks in the field and update the completed work order through the Internet.

4.1.3 Data Requirements

- **Desktop application**

The desktop application will need vector layers for parcels and sewer features (like sewer lines and manholes). Additionally, information on work orders, inspectors, and complaint history in a spreadsheet or preferable a database is required.

- **Mobile application**

The mobile application will need vector layers of manholes, sewer lines and parcels. Work order related information (e.g. task description and completion date) is required.

4.1.4 Work Flow

Workflow starts with incoming complaint call (e.g. Sewer backup). The supervisor will address the complaint call by creating a work order. Each work will contain task description, creation and completion dates, inspector's name and support vector and raster datasets. The field inspector will download the work order and execute the tasks in the field. Completed work order are updated and closed. Use a picture to show the workflow.

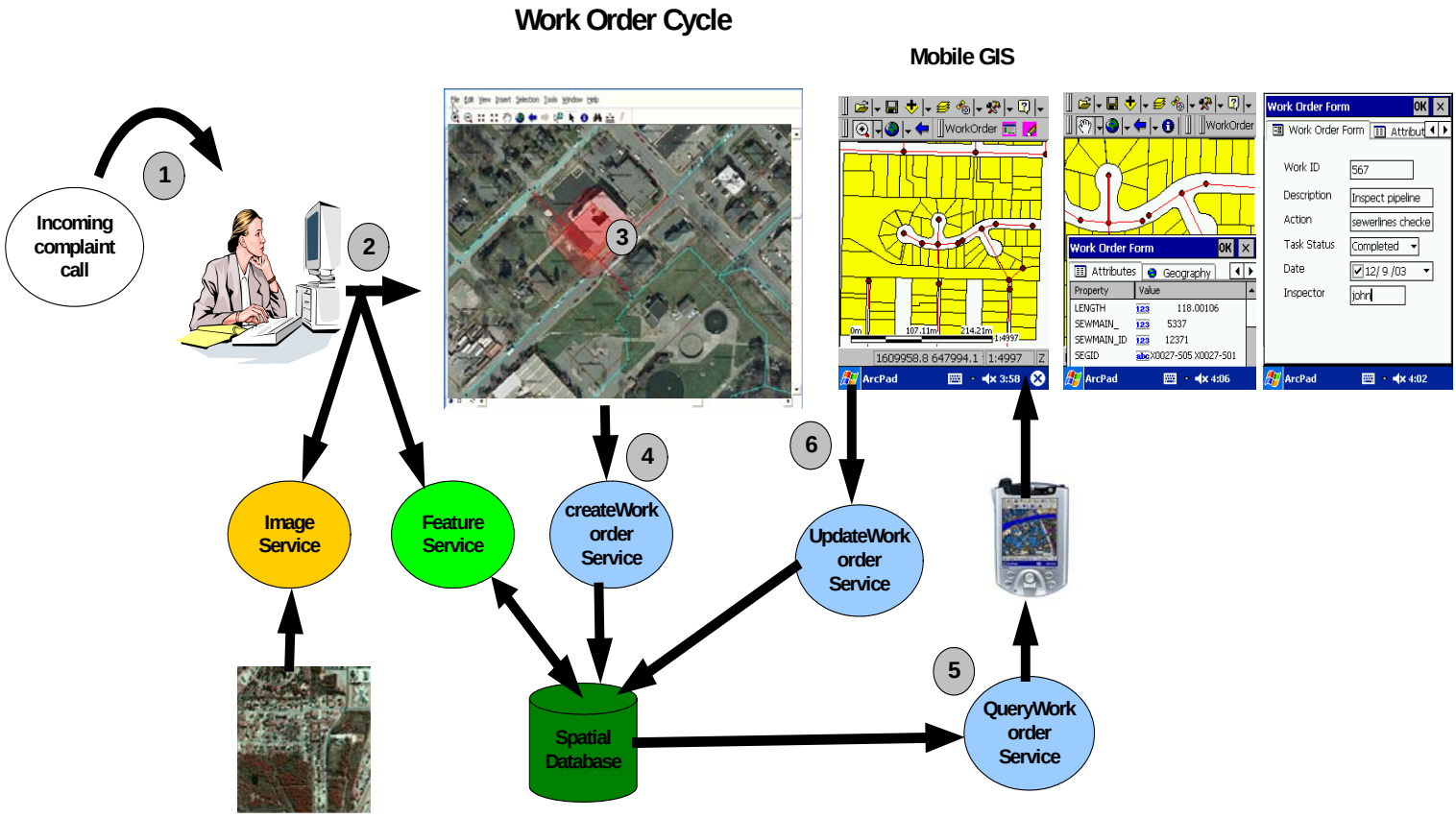

Figure 9 Work Order Cycle

4.2 Components

The components of the architecture are namely, desktop manager, services framework, spatial database and mobile application.

4.2.1 Desktop Manager

The Desktop Manager (DM) is primarily developed to assist supervisors to execute wastewater work management operations. The desktop application provides tools to view, query and transform vector layers apart from support for basic functions such as

zoom in/out, add and delete feature data layers. Advanced tools include remote vector and raster data access and overlay, redline features for field inspection, work order creation and data packaging are developed. The Desktop Manager utilizes remote spatial data web services to access and overlay vector and raster datasets for query and analysis. The primary services utilized by DM are spatial data access, feature redline and work order creation.

Desktop Manager is a Microsoft windows desktop application developed in Java programming language. Desktop Manager utilizes spatial data access, feature redline and work creation services to create a work order. The DM utilizes spatial data access service for accessing vector layers of wastewater features such as sewer lines and manholes. Two types of spatial data services are implemented to support vector and raster datasets respectively. DM uses WSDL [A.1] document for spatial data service and creates proxies at design time. Proxies are object classes that represent complex and simple data types marshaled into XML in a SOAP message. Proxies are utilized to construct SOAP messages requesting spatial data service. They will enable developers to operate on XML data in SOAP messages at Java objects level. Serializing/De-serializing of SOAP messages into java application objects and vice versa is achieved through WASP API's [42]. The DM sends a SOAP request [B.1, B.3] with the name of the vector layer and the spatial extents. The spatial data service processes the SOAP message by retrieving sewer and manhole features within the requested bounds and return a GML document in response SOAP message [B.2, B.4]. The response document will contain the requested feature attribute and geometry information encoded in GML. The DM or any application needs to convert the geographic feature information in GML to a usable data structure.

DM is implemented to use ESRI Shape files to manage wastewater features. Hence, GML encoded feature information is converted into ESRI Shape files on the fly by DM components. Image service is a wrapper around commercial web mapping software ArcIMS. The wrapper acts as proxy for DM and handles request and response operations to and from image service. Show a picture accessing image service. To access image service, DM creates a SOAP request with spatial extents as input parameters. SOAP response will contain the image for the requested extents. The desktop manager uses feature redline service to persistently store the suspected sewer features to a spatial database. The user can query for a complaint parcel and select suspected sewer features in proximity for inspection through buffer operation. Selected sewer features are persistently stored by creating a SOAP request to feature redline service. The SOAP request will contain features attribute and geometry information of redlined sewer features encoded in GML. The feature redline service inserts the selected sewer features into spatial database and returns acknowledgement of successful update.

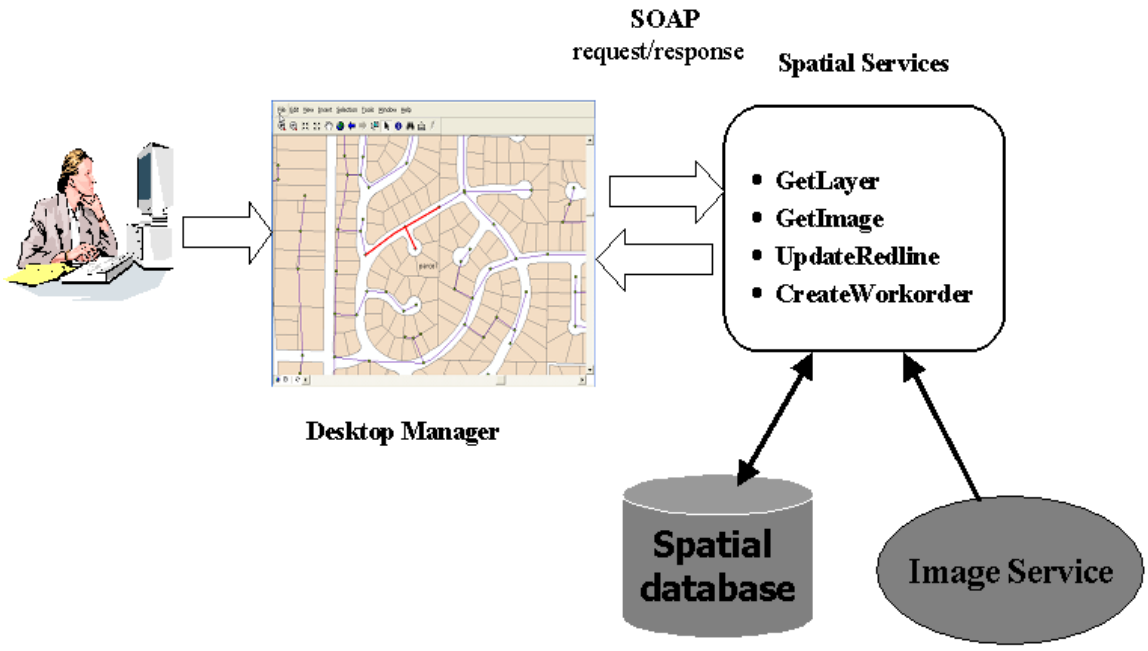

Figure 10 Desktop Manager

DM uses work order creation service to store work order information in the spatial database. The desktop application will allow managers to create task description and assign a field inspector with a work order. A work order contains information related to tasks description, inspector assigned, date created, action taken and date completed information. Work order service will store the work related, map projection, support data encoded in GML for field inspection. Tools to create, view and project spatial data layers are implemented with ESRI Map Objects API's [43]. Open source libraries as GeoTools [44] and JUMP [45] can be used to develop similar desktop application.

4.2.2 Service Framework

The following are the list of geospatial web services developed to support waster water work management services provided. Figure 7 shows the work management services.

- **GetLayer** - Spatial data delivery operation

The operation delivers vector layers in GML based on user specified spatial extent. Vector layer name and the spatial extents are input parameters and output is a GML document containing feature attributes and geometry information for the bounded region.

- **UpdateRedline** - Feature redline operation

The operation will allow users to select and store the redlined features for a work order persistently in a spatial database. Selected features (feature and geometry of suspected sewer lines) are input and output is a persistent record entry in the database.

- **CreateWorkorder** - Work order create

The operation will allow users to create and store work order information persistently in a spatial database. Work order information such as tasks, date created, inspector assigned, owner parcel and address are input and output is a persistent work order entry in the database

- **QueryWorkorder** - Query work order

The operation will deliver work order to field operators over the Internet. Inspector name is the input parameter and output is work order. Each work order will contain information on tasks, date create, owner, map projection, manholes, sewer lines and redlined features

- **UpdateWorkorder** - Work order update operation

The operation will update the work order information persistently to the database. Input is completed work order and output is update confirmation.

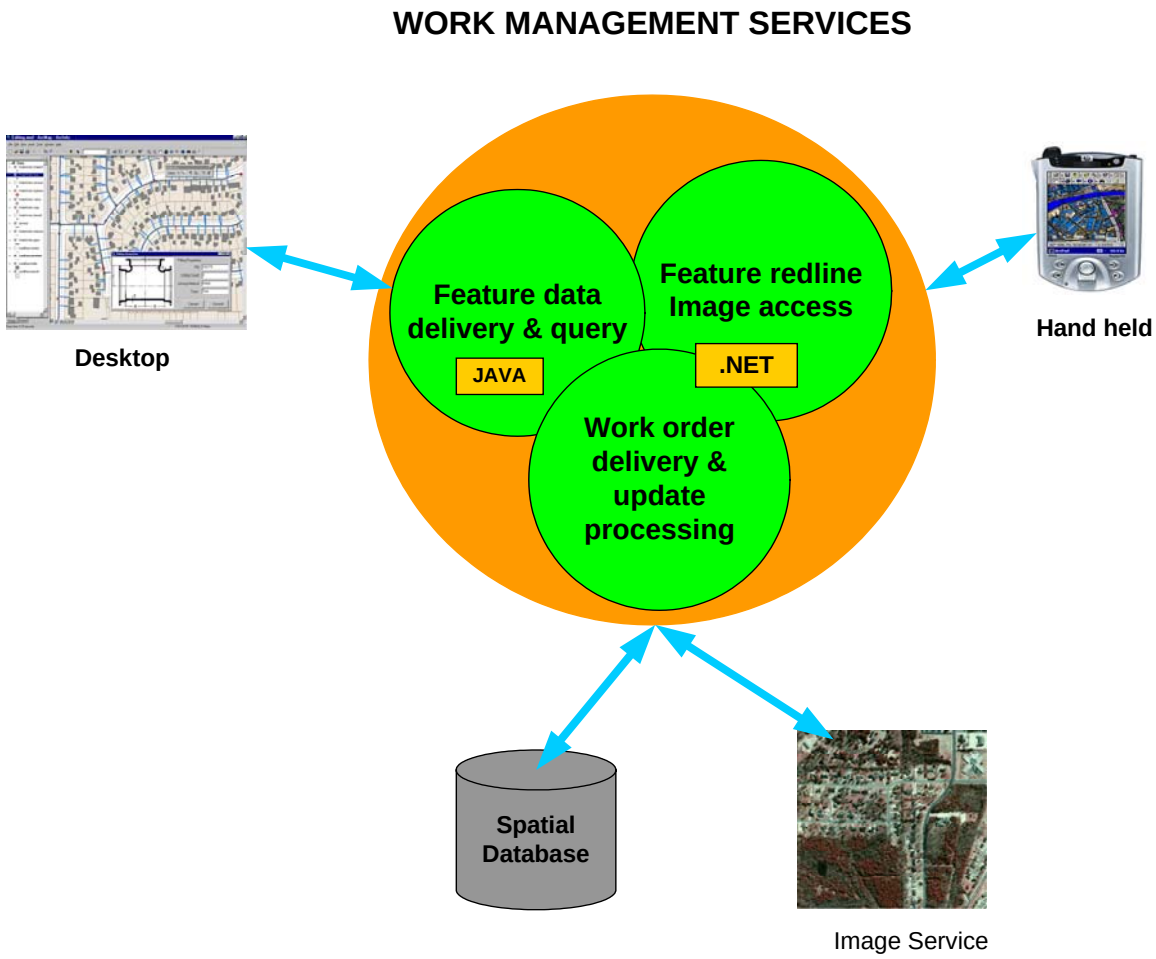

Figure 11 Work Management Services

4.2.3 Spatial Database

Spatial database is storage for geographic information. Unlike conventional databases, spatial database is designed to store geometric information. A spatial database stores vector and raster datasets. PostgreSQL (Pgsql) [39] is, an open source, database employed for storing geographic information. The database is spatially enabled through PostGIS [40], a spatial extension package. PostGIS creates necessary structures for loading, retrieving and modifying geographic data. Each vector or raster dataset in Pgsql database is stored as a table. Feature attributes and geometry of a data layer are stored in

columns field of the table. Pgsql supports storing, retrieving and editing spatial features apart from spatial operations such as intersect, union and buffer. Tools for loading ESRI shape files and coverages, administration utilities and ODBC/JDBC drivers are freely available for the database. Performance of PostgreSQL for retrieving and updating geometries is considerably fast. Tables such as work order, customers, history and inspectors are created for supporting work management data requirements in addition to tables to store waster water features.

4.2.4 Mobile Application

Mobile Application (MA) is designed to support field operations for wastewater services. The MA application assists field personnel to download, edit and update work order information remotely. Mobile Application provides field user with tools to execute field operations efficiently. MA utilizes mobile service to download and upload work orders. Field personnel can query and download work orders over the wireless Internet or through desktop Internet at the office. The field inspectors can edit feature attribute information in the field. On completion of the tasks, the work order can be transported through wireless Internet or desktop Internet in the office. MA utilizes work order upload service for updating the completed work. MA provides basic tools to view, zoom in/out, and query data layers.

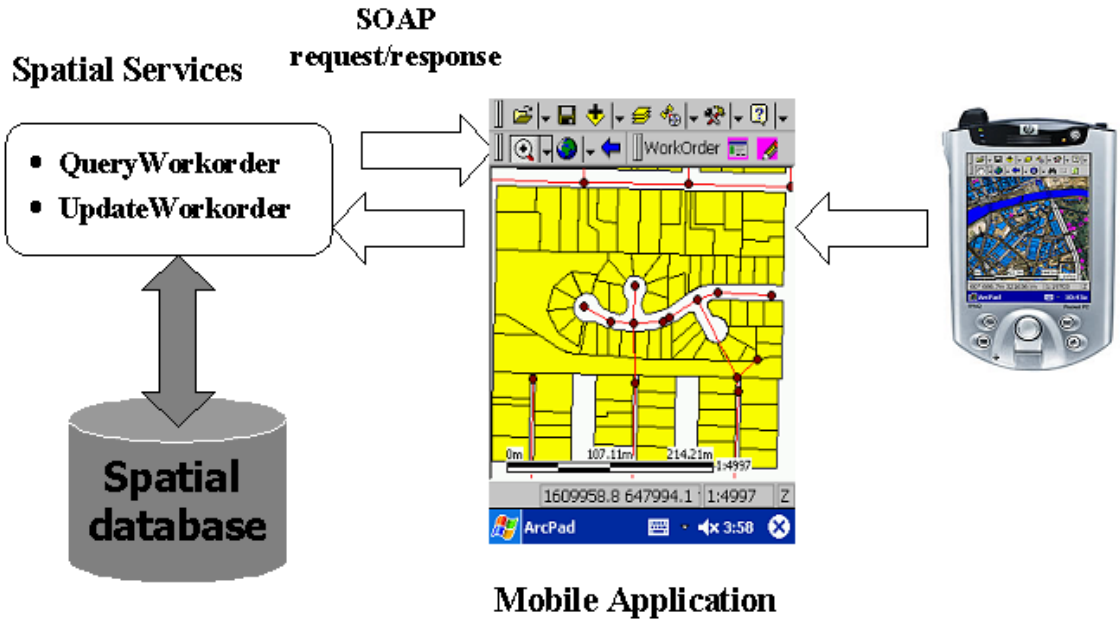

Figure 12 Mobile Application

Mobile Service interface provides web services to support fieldwork execution operations for mobile devices. The mobile service provides a WSDL document with description of operations supported, structure of input and output messages for each operations and the location of the service. MA downloads and updates work order information by binding to WSDL document of the mobile service. Three web services are supported namely,

- **CreateWorkorder**

Creates work order information such as task description, customer, inspector, date created, date to complete, action taken, spatial extents of map to be inspect. Supervisor utilizes this service from the desktop application to create a work order entry in the database.

- **QueryWorkorder**

Retrieves work order assigned to specific inspector. Field inspectors utilize this service from mobile application to query work order information. MA sends a request SOAP message [C.1] to query and downloads work order assigned to a field inspector. Each

work order contains tasks description, vector data layers of redlined sewer features, manholes and sewer features information encoded in response SOAP message [C.2]. Vector layers transported to mobile device are GML encoded and hence need to be converted into usable file structure on the client end. MA extracts and creates required geometries from GML SOAP message on the fly. Creating geometries from XML messages on mobile applications can be compute intensive with the increase in number of geographic features transported. Two approaches are implemented for transporting data layers to mobile devices. In first approach, data layers are encoded in GML and sent as part of SOAP message body. The mobile application needs to iterate through all the GML encoded data nodes for creating geometries. The second approach utilizes DIME [23] for transporting data layers to mobile device. DIME is used for transporting binary data over the Internet efficiently. It used Base64 encoding for representing the binary data on wire. This approach is suitable to transport large number of features and has low processing overhead. Since the geographic data represented in SOAP message body, the size of the SOAP packet increases with data or file structures. Unlike MIME, DIME approach transports information as an attachment that does not include the information encoded in the body part of SOAP message. This gives significant increase in performance due to little overhead associated with processing SOAP message. In MIME approach all the data is encoded into the SOAP body part that will need more computing capacity from the client end to process the message.

- **UpdateWorkorder**

Updates the work order information to the database. The completed work order information will contain action taken, work status, date completed and comments. The field inspector utilizes mobile work order update service to transport the changes made to

the work order in the field. MA is developed in ESRI Arcpad software for windows CE platform (PocketPC 2002) [46]. The software provides tools to view, edit and create ESRI shape files on mobile device.

CHAPTER V

RESULTS

Geographic information is critical to GIS applications for analysis and decision-making. Heterogeneity of data models and structures limit the information flow across applications. Web Service framework implementation provided following features,

- **Common Data Model**

Use of GML for data modeling provided a flexible and extensible approach to represent geographic feature characteristics. Common schema represenatation provides semantic information about the structure of data types exchanged. Data providers could create application schema to model domain specific geographic entities. Based on the schema structure user applications could validate and utilize spatial information. Use of GML to encode information enabled user applications to convert to desirable data structures. Agreement between the service provider and user on the schema structure representing data types exchanged is necessary.

- **Portable Data encoding and Transport**

Use of SOAP provided portable and extensible mechanism to encode and transport geographic information to user applications. XML structure of SOAP message allows custom and binary data types encoding. SOAP messages provide communications semantics between service provider and users through standard envelope structure.

- **Open Service Interface**

Use of web services description language provides open and extensible service interface. The service description document provides user applications with knowledge on the structure of data types exchanged, request/response messages and service location. Service providers are not bound to certain platform or programming language for service implementation. User applications can use open source tools and create requests in programming language of their choice for utilizing the spatial services. Hence, Service-oriented framework can accomplish the task of spatial data interoperability and and application interoperation.

CHAPTER VI

FUTURE WORK

Currently Spatial Data Services implemented are synchronous services. Asynchronous service implementation provides higher service realiability and decoupling among applications. Implementation of security features such as authentication and authorization to user applications for utilizing spatial services. With the increase in demand for data from distributed spatial databases, spatial services providing data query, integration and visualization capabilities to user applications need to be implemented. Further implementing spatial service based on Open GIS standards such as WMS, WFS and WCS for image, feature and coverage data services will enable broader utility of spatial information.

REFERENCES

1. Spatial Data Standards and GIS Interoperability, ESRI White Paper, http://www.esri.com/library/whitepapers/pdfs/spatial-data-standards.pdf
2. Open GIS Consortium (OGC) Specifications, http://www.opengis.org/specs/?page=specs
3. Spatial Data Transport Standard, http://mcmcweb.er.usgs.gov/sdts/
4. Spatial Archive and Interchange Format, SAIF, http://s2k-ftp.cs.berkeley.edu:8000/sequoia/schema/html/saif/saifHome.html
5. Geographic Data Files, http://www.ertico.com/links/gdf/gdf.htm
6. Shuxin Yuan, C. Vincent Tao, Development of a GIS Service Model in Support of online Geoprocessing
7. M. Bertolotto, J.D. Carswell, L. McGeown, P. Thijs, Geospatial Data Handling for web-based and Mobile Applications, 2002, Symposium on Geospatial Theory, Processing and Applications.
8. AA. Alesheikh, H. Helali, HA. Behroz, 2002, Web GIS: Technologies and Its Applications, Symposium on Geospatial Theory, Processing and Applications.
9. Winnie S.M. Tang, Jan Robert Selwood, 2003, The Development and Impact of Web-based Geographic Information Services, GIS development, http://www.gisdevelopment.net/technology/gis/mi03002pf.htm.
10. Aripak Panatkool, Sittchichai Laoveerakul, 2002, Decentralized GIS Web Services on Grid, Proceedings of the Open Source GIS – GRASS users conference.
11. Chen Liang, Chung-Ho Lee, Jae-Hong Kim, Hae-Young Bae, 2001, Scale-Dependent Transmission of Spatial Vector Date on the Internet, The 2nd International Conference on Information Integration and Web-based Applications & Services (Austria).

12. ESRI Shapefile Technical Description, July 1998, http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
13. Vector Product Format, NIMA, National Imagery and Mapping Agency http://ioc.unesco.org/oceanteacher/resourcekit/M3/Formats/Integrated/VPF/VPF%20Overview.htm
14. ESRI, ArcSDE, Spatial Database Management System, http://www.esri.com/software/arcgis/arcinfo/arcsde/index.html
15. Oracle 9*i*, Spatial Database Management System, http://otn.oracle.com/products/oracle9i/index.html
16. Rosen. M, Boak .J, 2002, eAI journal, January, p. 39-43, Developing a Web Services Strategy
17. Web Services Description Language, http://www.w3.org/TR/wsdl
18. Gudgin .M, Hadley .M, Mendelsohn .N, Moreau .J, Nielsen .H, 2003, W3C SOAP Version 1.2, http://www.w3.org/TR/SOAP/
19. EXtensible Markup Language (XML), http://www.w3c.org/XML/
20. Allan Doyle, Carl Reed, Jeff Harrison, Mark Reichardt, Introduction to OGC Web Services.
21. Open GIS Geographic Markup Language (GML) Implementation Specification, http://www.opengis.org/docs/02-023r4.pdf
22. Object-Oriented Middleware Technologies, http://dsonline.computer.org/middleware/projects_OOM.html
23. Direct Internet Message Encapsulation, DIME, IBM article, http://www-106.ibm.com/developerworks/library/ws-dime/
24. Universal Description, Discovery & Integration, UDDI, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec
25. Arnon Rosental, Len Seligman, Roger Costello, XML, Databases and Interoperabilty
26. Interoperable Data Services for Earth Science Data, White paper, Silvia Nittel
27. Integration of Spatial Data within a generic platform for location-based applications, Steffen Volz, Jan-Martin Bofinger, Symposium on Geospatial Theory, Processing and Applications 2002.

28. Steven H. Wong, Dilip Sarkar, Steven L. Swartz, A CORBA-based Middleware Architecture for Building Open and Interoperable GIS.
29. Charles B. Cranston, Frantisek Brabec, Gisli R. Hjaltason, Douglas Nebert, Hanan Samet, Adding an Interoperable Server Interface to a Spatial Database: Implementation Experience with OpenMap
30. Paul W. Calnan, Isabel F. Cruz, Object Interoperability for Geospatial Applications
31. Marta Wojnarowska, Bridget E. Ady, Interoperable Solutions in Web Mapping, Symposium on Geospatial Theory, Processing and Applications 2002.
32. Sanphet Chunithipaisan, Philip James, David Parker, Zainal Abdul Majeed, Simon Abele, Geospatial Interoperability via the Web: Supporting Land Administration in Kuala Lumpur, Map Asia 2003.
33. Omar Boucelma, Mehdi Essid, Zoe Lacroix, A WFS-Based Mediation System for GIS Interoperabilty
34. Nancy Wiegand, Naijun Zhou, Isabel F. Cruz, A Web Query System for Heterogeneous Geospatial Data
35. Ayesha Malik, Building XML schemas in an object-oriented framework, ftp://www6.software.ibm.com/software/developer/library/x-flexschema.pdf
36. CSIRO, XMML, http://xmml.arrc.csiro.au/
37. ESML, Earth Science Markup Language, http://esml.itsc.uah.edu/index.jsp
38. LandXML, http://www.landxml.org/
39. PostgreSQL, www.postgresql.org
40. PostGIS, http://postgis.org/
41. WKB4J, http://wkb4j.sourceforge.net
42. WASP Server for Java, Systinet, http://www.systinet.com/products/ wasp_jserver/ overview
43. Map Objects Java Edition, ESRI, http://esri.com/software/mojava/index.html
44. GeoTools, http://geotools.org/
45. JUMP, Unified Mapping Platform, http://www.vividsolutions.com/jump/
46. Arcpad, ESRI, http://www.esri.com/software/arcpad/

APPENDIX

A.1 Feature Data Service WSDL

```
<?xml version='1.0' encoding='utf-8' ?>
<wsdl:definitions name='example.webservices.gmlline.GetLayersPortImpl'
   targetNamespace='http://systinet.com/wsdl/example/webservices/gmlline/'
   xmlns:tns='http://systinet.com/wsdl/example/webservices/gmlline/'
   xmlns:ns0='http://systinet.com/xsd/SchemaTypes/'
   xmlns:ns1='http://idoox.com/interface'
   xmlns:map='http://systinet.com/mapping/'
   xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
   xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'>
      <xsd:schema elementFormDefault="qualified"
         targetNamespace="http://systinet.com/xsd/SchemaTypes/"
         xmlns:map="http://systinet.com/mapping/"
         xmlns:ns0="http://systinet.com/xsd/SchemaTypes/"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:tns="http://systinet.com/xsd/SchemaTypes/"
         xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
         xmlns:xns4="http://systinet.com/wsdl/example/webservices/gmlline/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <xsd:import
         namespace="http://systinet.com/wsdl/example/webservices/gmlline/"/>
         <xsd:element name="p0" nillable="true" type="xns4:GetLayerType"/>
         <xsd:element name="GetLayerResponseType_Response" nillable="true"
            type="xns4:GetLayerResponseType"/>
      </xsd:schema>

      <xsd:schema elementFormDefault="qualified"
         targetNamespace="http://systinet.com/wsdl/example/webservices/gmlline/"
         xmlns:ns0="http://systinet.com/xsd/SchemaTypes/"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <xsd:complexType name="GetLayerType">
            <xsd:annotation>
               <xsd:appinfo>
                  <map:java-type
                  name="example.webservices.gmlline.GetLayerType"/>
               </xsd:appinfo>
            </xsd:annotation>
            <xsd:sequence>
               <xsd:element name="extent" nillable="true"
                  type="tns:Extent"/>
               <xsd:element name="name" nillable="true" type="xsd:string"/>
               <xsd:element name="type" nillable="true" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
```

```
<xsd:complexType name="Extent">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="example.webservices.gmlline.Extent"/>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="XMax" type="xsd:double"/>
      <xsd:element name="XMin" type="xsd:double"/>
      <xsd:element name="YMax" type="xsd:double"/>
      <xsd:element name="YMin" type="xsd:double"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="GetLayerResponseType">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="example.webservices.gmlline.GetLayerResponseType"/>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="Layers" nillable="true"
         type="tns:Layers"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Layers">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="example.webservices.gmlline.Layers"/>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:complexContent>
      <xsd:extension base="tns:AbstractFeatureCollectionType">
         <xsd:sequence>
            <xsd:element name="lineFeature" nillable="true"
               type="tns:ArrayOfLineFeatureType"/>
            <xsd:element name="pointFeature" nillable="true"
               type="tns:ArrayOfPointFeatureType"/>
            <xsd:element name="polygonFeature" nillable="true"
               type="tns:ArrayOfPolygonFeatureType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="AbstractFeatureCollectionType">
```

```
            <xsd:annotation>
              <xsd:appinfo>
                <map:java-type
                name="example.webservices.gmlline.AbstractFeatureCollectionType"/>
              </xsd:appinfo>
            </xsd:annotation>
            <xsd:complexContent>
              <xsd:extension base="tns:AbstractFeatureCollectionBaseType">
                <xsd:sequence>
                  <xsd:element name="featureMember" nillable="true"
                    type="tns:ArrayOfFeatureAssociationType"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
          <xsd:complexType name="AbstractFeatureCollectionBaseType">
            <xsd:annotation>
              <xsd:appinfo>
                <map:java-type

name="example.webservices.gmlline.AbstractFeatureCollectionBaseType"/>
              </xsd:appinfo>
            </xsd:annotation>
            <xsd:complexContent>
              <xsd:extension base="tns:AbstractFeatureType">
                <xsd:sequence/>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
          <xsd:complexType name="AbstractFeatureType">
            <xsd:annotation>
              <xsd:appinfo>
                <map:java-type
                name="example.webservices.gmlline.AbstractFeatureType"/>
              </xsd:appinfo>
            </xsd:annotation>
            <xsd:sequence>
              <xsd:element name="boundedBy" nillable="true"
                type="tns:BoundingShapeType"/>
              <xsd:element name="description" nillable="true"
                type="xsd:string"/>
              <xsd:element name="fid" nillable="true" type="xsd:string"/>
              <xsd:element name="name" nillable="true" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="BoundingShapeType">
            <xsd:annotation>
```

```
      <xsd:appinfo>
        <map:java-type
        name="example.webservices.gmlline.BoundingShapeType"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name="box" type="tns:BoxType"/>
      <xsd:element name="null" type="tns:NullType"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="BoxType">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="example.webservices.gmlline.BoxType"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name="coord" type="tns:ArrayOfCoordType"/>
      <xsd:element name="coordinates" type="tns:CoordinatesType"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="CoordType">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="example.webservices.gmlline.CoordType"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="X" nillable="true" type="xsd:decimal"/>
      <xsd:element name="Y" nillable="true" type="xsd:decimal"/>
      <xsd:element name="Z" nillable="true" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ArrayOfCoordType">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="[Lexample.webservices.gmlline.CoordType;"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
        name="CoordType" nillable="true" type="tns:CoordType"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:complexType name="LineFeatureType">
  <xsd:annotation>
    <xsd:appinfo>
      <map:java-type
      name="example.webservices.gmlline.LineFeatureType"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="tns:AbstractFeatureType">
      <xsd:sequence>
      <xsd:element name="address" nillable="true"  type="xsd:string"/>
      <xsd:element name="diameter" type="xsd:float"/>
      <xsd:element name="fnode_" nillable="true"    type="xsd:integer"/>
      <xsd:element name="label" nillable="true"    type="xsd:string"/>
      <xsd:element name="length" type="xsd:float"/>
      <xsd:element name="lpoly_" nillable="true"   type="xsd:integer"/>
      <xsd:element name="map_" nillable="true"    type="xsd:string"/>
      <xsd:element name="material" nillable="true"     type="xsd:string"/>
      <xsd:element name="multiCenterLineOf"   nillable="true"
            type="tns:MultiLineStringPropertyType"/>
      <xsd:element name="rpoly_" nillable="true"  type="xsd:integer"/>
      <xsd:element name="segid" nillable="true"       type="xsd:integer"/>
      <xsd:element name="sewmain_" nillable="true"  type="xsd:integer"/>
      <xsd:element name="sewmain_id" nillable="true"   type="xsd:integer"/>
      <xsd:element name="tnode_" nillable="true"    type="xsd:integer"/>
      <xsd:element name="type" nillable="true"     type="xsd:string"/>
      <xsd:element name="year" nillable="true"     type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MultiLineStringPropertyType">
  <xsd:annotation>
    <xsd:appinfo>
      <map:java-type
      name="example.webservices.gmlline.MultiLineStringPropertyType"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="tns:GeometryPropertyType">
      <xsd:sequence>
        <xsd:element name="MultiLineString" nillable="true"
          type="tns:MultiLineStringType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:complexType name="GeometryPropertyType">
  <xsd:annotation>
    <xsd:appinfo>
      <map:java-type
      name="example.webservices.gmlline.GeometryPropertyType"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="_Geometry" nillable="true"
      type="tns:AbstractGeometryType"/>
    <xsd:element name="actuate" nillable="true"   type="tns:Actuate"/>
    <xsd:element name="arcrole" nillable="true"    type="xsd:string"/>
    <xsd:element name="href" nillable="true" type="xsd:string"/>
    <xsd:element name="remoteSchema" nillable="true"  type="xsd:string"/>
    <xsd:element name="role" nillable="true" type="xsd:string"/>
    <xsd:element name="show" nillable="true" type="tns:Show"/>
    <xsd:element name="title" nillable="true"    type="xsd:string"/>
    <xsd:element name="type" nillable="true" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AbstractGeometryType">
  <xsd:annotation>
    <xsd:appinfo>
      <map:java-type
      name="example.webservices.gmlline.AbstractGeometryType"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="gid" nillable="true" type="xsd:string"/>
    <xsd:element name="srsName" nillable="true"
      type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="MultiLineStringType">
  <xsd:annotation>
    <xsd:appinfo>
      <map:java-type
      name="example.webservices.gmlline.MultiLineStringType"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="tns:GeometryCollectionType">
      <xsd:sequence>
        <xsd:element name="lineStringMember" nillable="true"
          type="tns:ArrayOfLineStringMember"/>
      </xsd:sequence>
    </xsd:extension>
```

```
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="GeometryCollectionType">
      <xsd:annotation>
        <xsd:appinfo>
          <map:java-type
          name="example.webservices.gmlline.GeometryCollectionType"/>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:complexContent>
        <xsd:extension base="tns:AbstractGeometryCollectionBaseType">
          <xsd:sequence>
            <xsd:element name="geometryMember" nillable="true"
              type="tns:ArrayOfGeometryAssociationType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AbstractGeometryCollectionBaseType">
      <xsd:annotation>
        <xsd:appinfo>
          <map:java-type

name="example.webservices.gmlline.AbstractGeometryCollectionBaseType"/>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:complexContent>
        <xsd:extension base="tns:AbstractGeometryType">
          <xsd:sequence/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="LineStringMember">
      <xsd:annotation>
        <xsd:appinfo>
          <map:java-type
          name="example.webservices.gmlline.LineStringMember"/>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="LineString" nillable="true"
          type="tns:LineStringType"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="LineStringType">
      <xsd:annotation>
```

```
      <xsd:appinfo>
        <map:java-type
        name="example.webservices.gmlline.LineStringType"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name="coord" type="tns:ArrayOfCoordType"/>
      <xsd:element name="coordinates" type="tns:CoordinatesType"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="ArrayOfLineStringMember">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="[Lexample.webservices.gmlline.LineStringMember;"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
        name="LineStringMember" nillable="true"
        type="tns:LineStringMember"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ArrayOfLineFeatureType">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="[Lexample.webservices.gmlline.LineFeatureType;"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0"
        name="LineFeatureType" nillable="true"
        type="tns:LineFeatureType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="PointFeatureType">
    <xsd:annotation>
      <xsd:appinfo>
        <map:java-type
        name="example.webservices.gmlline.PointFeatureType"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="tns:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="elev" type="xsd:float"/>
```

```
            <xsd:element name="fitt_type" nillable="true"
               type="xsd:string"/>
            <xsd:element name="location" nillable="true"
               type="tns:PointPropertyType"/>
            <xsd:element name="map_" nillable="true"
               type="xsd:string"/>
            <xsd:element name="mh_code" nillable="true"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PointPropertyType">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="example.webservices.gmlline.PointPropertyType"/>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:complexContent>
      <xsd:extension base="tns:GeometryPropertyType">
         <xsd:sequence>
            <xsd:element name="Point" nillable="true"
               type="tns:PointType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="PointType">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="example.webservices.gmlline.PointType"/>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:choice>
      <xsd:element name="coord" type="tns:CoordType"/>
      <xsd:element name="coordinates" type="tns:CoordinatesType"/>
   </xsd:choice>
</xsd:complexType>
<xsd:complexType name="ArrayOfPointFeatureType">
   <xsd:annotation>
      <xsd:appinfo>
         <map:java-type
         name="[Lexample.webservices.gmlline.PointFeatureType;"/>
      </xsd:appinfo>
   </xsd:annotation>
```

```
            <xsd:sequence>
               <xsd:element maxOccurs="unbounded" minOccurs="0"
                  name="PointFeatureType" nillable="true"
                  type="tns:PointFeatureType"/>
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="PolygonFeatureType">
            <xsd:annotation>
               <xsd:appinfo>
                  <map:java-type
                  name="example.webservices.gmlline.PolygonFeatureType"/>
               </xsd:appinfo>
            </xsd:annotation>
            <xsd:complexContent>
               <xsd:extension base="tns:AbstractFeatureType">
                  <xsd:sequence>
                     <xsd:element name="multiExtentOf" nillable="true"
                        type="tns:MultiPolygonPropertyType"/>
                  </xsd:sequence>
               </xsd:extension>
            </xsd:complexContent>
         </xsd:complexType>
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name='GetLayersPortImpl_getLayer_1_Request'>
      <wsdl:part name='p0' element='ns0:p0'/>
   </wsdl:message>
   <wsdl:message name='runtimeHeaders_0'>
      <wsdl:part name='header_0' element='ns1:instance'/>
   </wsdl:message>
   <wsdl:message name='GetLayersPortImpl_getLayer_Response'>
      <wsdl:part name='response' element='ns0:GetLayerResponseType_Response'/>
   </wsdl:message>
   <wsdl:portType name='GetLayersPortImpl'>
      <wsdl:operation name='getLayer' parameterOrder='p0'>
         <wsdl:input message='tns:GetLayersPortImpl_getLayer_1_Request'/>
         <wsdl:output message='tns:GetLayersPortImpl_getLayer_Response'/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name='GetLayersPortImpl' type='tns:GetLayersPortImpl'>
      <soap:binding transport='http://schemas.xmlsoap.org/soap/http' style='document'/>
      <wsdl:operation name='getLayer'>
         <map:java-operation name='getLayer'/>
         <soap:operation

soapAction='http://systinet.com/wsdl/example/webservices/gmlline/GetLayersPortImpl#
getLayer'
```

```
        style='document'/>
        <wsdl:input>
          <soap:body use='literal'/>
          <soap:header message='tns:runtimeHeaders_0' part='header_0' use='literal'/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use='literal'/>
          <soap:header message='tns:runtimeHeaders_0' part='header_0' use='literal'/>
        </wsdl:output>
      </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name='GetLayersPortImpl'>
    <wsdl:port name='GetLayersPortImpl' binding='tns:GetLayersPortImpl'>
      <soap:address location='http://130.18.90.209:6060/demo/basic/gmllineService1'/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.2 Mobile Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<definitions  xmlns:s1="http://www.opengis.net/gml"
                 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                 xmlns:s="http://www.w3.org/2001/XMLSchema"
                 xmlns:s0="http://www.erc.msstate.edu/MobileService"
                 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
               xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
              targetNamespace="http://www.erc.msstate.edu/MobileService"
              xmlns="http://schemas.xmlsoap.org/wsdl/">
 <types>
s:schema elementFormDefault="qualified"
          targetNamespace="http://www.erc.msstate.edu/MobileService">
<s:import namespace="http://www.opengis.net/gml" />
<s:element name="WorkOrderRequest" type="s0:Inspector" />
 <s:complexType name="Inspector">
  <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="InspectorName" type="s:string" />
 </s:sequence>
 </s:complexType>
<s:element name="WorkOrder" type="s0:WorkOrder" />
    <s:complexType name="WorkOrder">
     <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="workorderid" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="createdOn" type="s:string" />
```

```
<s:element minOccurs="0" maxOccurs="1" name="description" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="priority" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="owner" type="s0:AddressType" />
<s:element minOccurs="0" maxOccurs="1" name="workOrderStatus" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="InspectionDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Inspector" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="actionTaken" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="comments" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="map" type="s0:MapType" />
<s:element minOccurs="0" maxOccurs="1" name="location" type="s:string" />
    </s:sequence>
   </s:complexType>
   <s:complexType name="AddressType">
    <s:sequence>
     <s:element minOccurs="0" maxOccurs="1" name="firstName" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="lastName" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="street" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="state" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="country" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="zip" type="s:string" />
     <s:element minOccurs="0" maxOccurs="1" name="phone" type="s:string" />
    </s:sequence>
   </s:complexType>
   <s:complexType name="MapType">
    <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Projection"
                                              type="s0:projectionType" />
     <s:element minOccurs="0" maxOccurs="1" name="redlineFeature"
                                             type="s0:lineFeatureType" />
     <s:element minOccurs="0" maxOccurs="1" name="manholeFeature"
                                             type="s0:manholeType" />
     <s:element minOccurs="0" maxOccurs="1" name="pointFeature"
                                             type="s0:pointFeatureType" />
     <s:element minOccurs="0" maxOccurs="1" name="lineFeature"
                                              type="s0:lineFeatureType" />
    </s:sequence>
   </s:complexType>
   <s:complexType name="projectionType">
    <s:sequence>
     <s:element minOccurs="0" maxOccurs="1" name="proj" type="s:string" />
    </s:sequence>
   </s:complexType>
   <s:complexType name="lineFeatureType">
    <s:complexContent mixed="false">
     <s:extension base="s1:AbstractFeatureType">
      <s:sequence>
```

```
            <s:element minOccurs="0" maxOccurs="1" ref="s1:multiCenterLineOf" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:complexType name="manholeFeatureType">
      <s:complexContent mixed="false">
        <s:extension base="s1:AbstractFeatureType">
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" ref="s1:Location" />
            <s:element minOccurs="0" maxOccurs="1" name="mh_id" type="s:string" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:complexType name="pointFeatureType">
      <s:complexContent mixed="false">
        <s:extension base="s1:AbstractFeatureType">
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" ref="s1:multiLocation" />
            <s:element minOccurs="0" maxOccurs="1" name="mh_id" type="s:string" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:complexType name="manholeType">
      <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="manhole"
                                                    type="s0:manholeFeatureType" />
      </s:sequence>
    </s:complexType>
    <s:element name="WorkOrderUpdateRequest" type="s0:UpdateWorkOrder" />
    <s:complexType name="UpdateWorkOrder">
      <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="workorderid" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="workOrderStatus" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="InspectionDate" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="actionTaken" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="comments" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="InspectorName" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="map"
                                          type="s0:featureUpdateType" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="featureUpdateType">
      <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="sewerlineFeature"
```

```
                                                    type="s0:sewerlineFeatureType" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="sewerlineFeatureType">
      <s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="sewerline"
                                                          type="s0:sewerlineType" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="sewerlineType">
      <s:complexContent mixed="false">
       <s:extension base="s1:AbstractFeatureType">
        <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="sew_id" type="s:string" />
         <s:element minOccurs="0" maxOccurs="1" name="diameter" type="s:string" />
         <s:element minOccurs="0" maxOccurs="1" name="material" type="s:string" />
         <s:element minOccurs="0" maxOccurs="1" name="label" type="s:string" />
        </s:sequence>
       </s:extension>
      </s:complexContent>
     </s:complexType>
     <s:complexType name="WorkUpdateStatus">
      <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="updateStatus"  type="s:boolean" />
      </s:sequence>
     </s:complexType>
     <s:element name="WorkOrderUpdateStatus" type="s0:WorkUpdateStatus" />
     <s:element name="WorkOrderCreateRequest" type="s0:CreateWorkOrder" />
     <s:complexType name="CreateWorkOrder">
      <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="workorder_no" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="owner_no" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="inspetor_name" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="dateCreated" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="tasks" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="workStatus" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="dateCompleted" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="actionTaken" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="comments" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="location" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="XMIN" type="s:double" />
    <s:element minOccurs="1" maxOccurs="1" name="YMIN" type="s:double" />
    <s:element minOccurs="1" maxOccurs="1" name="XMAX" type="s:double" />
    <s:element minOccurs="1" maxOccurs="1" name="YMAX" type="s:double" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="WorkCreateStatus">
```

```
      <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="createStatus" type="s:boolean" />
      </s:sequence>
     </s:complexType>
     <s:element name="WorkOrderCreateStatus" type="s0:WorkCreateStatus" />
     <s:element name="QueryDataSource" type="s0:DataSource" />
     <s:complexType name="DataSource">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="dsName" type="s:string" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="DataSourceLayers">
      <s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="layer"
                                                    type="s0:LayerType" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="LayerType">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="schema" type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="meta" type="s:string" />
      </s:sequence>
     </s:complexType>
     <s:element name="DataSourceLayers" type="s0:DataSourceLayers" />
    </s:schema>
    <s:schema elementFormDefault="qualified"
                targetNamespace="http://www.opengis.net/gml">
     <s:import namespace="http://www.erc.msstate.edu/MobileService" />
     <s:complexType name="AbstractFeatureType" abstract="true">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="description" type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="boundedBy"
                                                  type="s1:BoundingShapeType" />
      </s:sequence>
      <s:attribute name="fid" type="s:ID" />
     </s:complexType>
     <s:complexType name="BoundingShapeType">
      <s:sequence>
       <s:choice minOccurs="1" maxOccurs="1">
        <s:element minOccurs="0" maxOccurs="1" name="Box" type="s1:BoxType" />
        <s:element minOccurs="1" maxOccurs="1" name="null" type="s1:NullType" />
       </s:choice>
      </s:sequence>
     </s:complexType>
     <s:complexType name="BoxType">
```

```
   <s:complexContent mixed="false">
     <s:extension base="s1:AbstractGeometryType">
       <s:sequence>
         <s:choice minOccurs="0" maxOccurs="unbounded">
   <s:element minOccurs="0" maxOccurs="1" name="coord" type="s1:CoordType" />
   <s:element minOccurs="0" maxOccurs="1" name="coordinates"
                                                  type="s1:CoordinatesType" />
         </s:choice>
       </s:sequence>
     </s:extension>
   </s:complexContent>
  </s:complexType>
  <s:complexType name="AbstractGeometryType" abstract="true">
   <s:attribute name="gid" type="s:ID" />
   <s:attribute name="srsName" type="s:anyURI" />
  </s:complexType>
  <s:complexType name="PointType">
   <s:complexContent mixed="false">
     <s:extension base="s1:AbstractGeometryType">
       <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="coord"
                                               type="s1:CoordinatesType" />
<s:element minOccurs="1" maxOccurs="1" name="fid" type="s:int" />
<s:element minOccurs="0" maxOccurs="1" name="mh_id" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="fitt_type" type="s:string" />
<s:element minOccurs="1" maxOccurs="1" name="elev" type="s:double" />
<s:element minOccurs="0" maxOccurs="1" name="map_" type="s:string" />
       </s:sequence>
     </s:extension>
   </s:complexContent>
  </s:complexType>
  <s:complexType name="CoordinatesType">
   <s:simpleContent>
     <s:extension base="s:string">
       <s:attribute default="." name="decimal" type="s:string" />
       <s:attribute default="," name="cs" type="s:string" />
       <s:attribute default=" " name="ts" type="s:string" />
     </s:extension>
   </s:simpleContent>
  </s:complexType>
  <s:complexType name="LineStringType">
   <s:complexContent mixed="false">
     <s:extension base="s1:AbstractGeometryType">
       <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="coord"
                                                    type="s1:CoordinatesType" />
        <s:element minOccurs="1" maxOccurs="1" name="fid" type="s:int" />
```

```
      <s:element minOccurs="1" maxOccurs="1" name="fnode_" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="tnode_" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="lpoly_" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="rpoly_" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="length" type="s:double" />
      <s:element minOccurs="1" maxOccurs="1" name="sewmain_" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="sewmain_id" type="s:int" />
      <s:element minOccurs="0" maxOccurs="1" name="segid" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="type" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="diameter" type="s:double" />
      <s:element minOccurs="0" maxOccurs="1" name="material" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="address" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="map_" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="year" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="label" type="s:string" />
    </s:sequence>
   </s:extension>
  </s:complexContent>
</s:complexType>
<s:complexType name="AbstractGeometryCollectionBaseType" abstract="true">
  <s:complexContent mixed="false">
    <s:extension base="s1:AbstractGeometryType" />
  </s:complexContent>
</s:complexType>
<s:complexType name="GeometryCollectionType">
  <s:complexContent mixed="false">
    <s:extension base="s1:AbstractGeometryCollectionBaseType">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
              name="geometryMember" type="s1:GeometryAssociationType"/ >
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>
<s:complexType name="GeometryAssociationType">
  <s:sequence>
    <s:choice minOccurs="1" maxOccurs="1">
 <s:element minOccurs="0" maxOccurs="1" name="Point" type="s1:PointType" />
 <s:element minOccurs="0" maxOccurs="1" name="LineString"
                                              type="s1:LineStringType" />
 <s:element minOccurs="0" maxOccurs="1" name="MultiPoint"
                                               type="s1:MultiPointType" />
    </s:choice>
  </s:sequence>
  <s:attribute name="remoteSchema" type="s:anyURI" />
</s:complexType>
<s:complexType name="MultiPointType">
```

```
  <s:complexContent mixed="false">
    <s:extension base="s1:GeometryCollectionType">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded" name="pointMember"
                                                              type="s1:PointType" />
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>
<s:complexType name="CoordType">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="X" type="s:decimal" />
    <s:element minOccurs="0" maxOccurs="1" name="Y" type="s:decimal" />
    <s:element minOccurs="0" maxOccurs="1" name="Z" type="s:decimal" />
  </s:sequence>
</s:complexType>
<s:simpleType name="NullType">
  <s:restriction base="s:string">
    <s:enumeration value="inapplicable" />
    <s:enumeration value="unknown" />
    <s:enumeration value="unavailable" />
    <s:enumeration value="missing" />
  </s:restriction>
</s:simpleType>
<s:element name="Location" type="s1:PointPropertyType" />
<s:complexType name="PointPropertyType">
  <s:complexContent mixed="false">
    <s:extension base="s1:GeometryPropertyType">
      <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Point" type="s1:PointType" />
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>
<s:complexType name="GeometryPropertyType">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Item" />
  </s:sequence>
  <s:attribute name="remoteSchema" type="s:anyURI" />
</s:complexType>
<s:complexType name="MultiPointPropertyType">
  <s:complexContent mixed="false">
    <s:extension base="s1:GeometryPropertyType">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="MultiPoint"
                                                    type="s1:MultiPointType" />
      </s:sequence>
```

```
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:complexType name="MultiGeometryPropertyType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType" />
      </s:complexContent>
    </s:complexType>
    <s:complexType name="MultiPolygonPropertyType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType" />
      </s:complexContent>
    </s:complexType>
    <s:complexType name="LineStringPropertyType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType" />
      </s:complexContent>
    </s:complexType>
    <s:complexType name="PolygonPropertyType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType" />
      </s:complexContent>
    </s:complexType>
    <s:complexType name="MultiLineStringType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType">
          <s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="lineStringMember"
                                                    type="s1:LineStringType" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:complexType name="MultiLineStringPropertyType">
      <s:complexContent mixed="false">
        <s:extension base="s1:GeometryPropertyType">
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="MultiLineString"
                                                    type="s1:MultiLineStringType" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
    <s:element name="multiLocation" type="s1:MultiPointPropertyType" />
    <s:complexType name="AbstractFeatureCollectionBaseType" abstract="true">
      <s:complexContent mixed="false">
        <s:extension base="s1:AbstractFeatureType" />
```

```
    </s:complexContent>
   </s:complexType>
   <s:complexType name="AbstractFeatureCollectionType" abstract="true">
    <s:complexContent mixed="false">
     <s:extension base="s1:AbstractFeatureCollectionBaseType">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="unbounded" name="featureMember"
                                                    type="s1:FeatureAssociationType" />
      </s:sequence>
     </s:extension>
    </s:complexContent>
   </s:complexType>
   <s:complexType name="FeatureAssociationType">
    <s:sequence>
     <s:element minOccurs="0" maxOccurs="1" name="Item" />
    </s:sequence>
    <s:attribute name="remoteSchema" type="s:anyURI" />
   </s:complexType>
   <s:element name="multiCenterLineOf" type="s1:MultiLineStringPropertyType" />
  </s:schema>
</types>
<message name="QueryWorkOrderSoapIn">
  <part name="InspectorName" element="s0:WorkOrderRequest" />
</message>
<message name="QueryWorkOrderSoapOut">
  <part name="QueryWorkOrderResult" element="s0:WorkOrder" />
</message>
<message name="UpdateWorkOrderSoapIn">
  <part name="workorder" element="s0:WorkOrderUpdateRequest" />
</message>
<message name="UpdateWorkOrderSoapOut">
  <part name="UpdateWorkOrderResult" element="s0:WorkOrderUpdateStatus" />
</message>
<message name="CreateWorkOrderSoapIn">
  <part name="workorder" element="s0:WorkOrderCreateRequest" />
</message>
<message name="CreateWorkOrderSoapOut">
  <part name="CreateWorkOrderResult" element="s0:WorkOrderCreateStatus" />
</message>
<message name="QueryDataSourceSoapIn">
  <part name="datasource" element="s0:QueryDataSource" />
</message>
<message name="QueryDataSourceSoapOut">
  <part name="QueryDataSourceResult" element="s0:DataSourceLayers" />
</message>
<portType name="MobileInterface">
  <operation name="QueryWorkOrder">
```

```
    <input message="s0:QueryWorkOrderSoapIn" />
    <output message="s0:QueryWorkOrderSoapOut" />
   </operation>
   <operation name="UpdateWorkOrder">
    <input message="s0:UpdateWorkOrderSoapIn" />
    <output message="s0:UpdateWorkOrderSoapOut" />
   </operation>
   <operation name="CreateWorkOrder">
    <input message="s0:CreateWorkOrderSoapIn" />
    <output message="s0:CreateWorkOrderSoapOut" />
   </operation>
   <operation name="QueryDataSource">
    <input message="s0:QueryDataSourceSoapIn" />
    <output message="s0:QueryDataSourceSoapOut" />
   </operation>
  </portType>
  <binding name="MobileInterface" type="s0:MobileInterface">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
   <operation name="QueryWorkOrder">
    <soap:operation soapAction="" style="document" />
    <input>
     <soap:body use="literal" />
    </input>
    <output>
     <soap:body use="literal" />
    </output>
   </operation>
   <operation name="UpdateWorkOrder">
    <soap:operation soapAction="" style="document" />
    <input>
     <soap:body use="literal" />
    </input>
    <output>
     <soap:body use="literal" />
    </output>
   </operation>
   <operation name="CreateWorkOrder">
    <soap:operation soapAction="" style="document" />
    <input>
     <soap:body use="literal" />
    </input>
    <output>
     <soap:body use="literal" />
    </output>
   </operation>
   <operation name="QueryDataSource">
    <soap:operation soapAction="" style="document" />
```

```
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <service name="MobileService">
    <port name="MobileInterface" binding="s0:MobileInterface">
      <soap:address location="http://130.18.13.126/MobileService/MobileService.asmx" />
    </port>
  </service>
</definitions>
```

B.1 Manhole request SOAP Message

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:m0="http://systinet.com/wsdl/example/webservices/gmlline/">
    <SOAP-ENV:Header>
        <m:instance xmlns:m="http://idoox.com/interface">
            <m:id>String</m:id>
            <m:setId>String</m:setId>
            <m:notFound>String</m:notFound>
        </m:instance>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <m:p0 xmlns:m="http://systinet.com/xsd/SchemaTypes/">
            <m0:extent>
                <m0:XMax>1603937.36460391</m0:XMax>
                <m0:XMin>1602657.61547023</m0:XMin>
                <m0:YMax>647191.61784469</m0:YMax>
                <m0:YMin>645803.54348906</m0:YMin>
            </m0:extent>
            <m0:name>manhole</m0:name>
            <m0:type>POINT</m0:type>
        </m:p0>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2 Manhole response SOAP Message


```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:e="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:d="http://www.w3.org/2001/XMLSchema"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wn0="http://systinet.com/xsd/SchemaTypes/"
xmlns:wn1="http://idoox.com/interface"
xmlns:wn2="http://systinet.com/wsdl/example/webservices/gmlline/">
<e:Body>
<wn0:GetLayerResponseType_Response  i:type="wn2:GetLayerResponseType">
          <wn2:Layers i:type="wn2:Layers">
             <wn2:boundedBy i:nil="true"/>
             <wn2:description i:nil="true"/>
             <wn2:fid i:nil="true"/>
             <wn2:name i:nil="true"/>
             <wn2:featureMember i:nil="true"/>
             <wn2:lineFeature i:nil="true"/>
             <wn2:pointFeature i:type="wn2:ArrayOfPointFeatureType">
                <wn2:PointFeatureType i:type="wn2:PointFeatureType">
                   <wn2:boundedBy i:nil="true"/>
                   <wn2:description i:nil="true"/>
                   <wn2:fid i:type="d:string">600</wn2:fid>
                   <wn2:name i:nil="true"/>
                   <wn2:elev i:type="d:float">0.0</wn2:elev>
                   <wn2:fitt_type i:type="d:string">S</wn2:fitt_type>
                   <wn2:location i:type="wn2:PointPropertyType">
                      <wn2:Point i:type="wn2:PointType">
                         <wn2:coord i:type="wn2:CoordType">
                            <wn2:X i:type="d:decimal">1602718.057004</wn2:X>
                            <wn2:Y i:type="d:decimal">647121.539068</wn2:Y>
                            <wn2:Z i:nil="true"/>
                         </wn2:coord>
                      </wn2:Point>
                   </wn2:location>
                   <wn2:map_ i:type="d:string">8319SW</wn2:map_>
                   <wn2:mh_code i:type="d:string">S05814</wn2:mh_code>
                </wn2:PointFeatureType>
                <wn2:PointFeatureType i:type="wn2:PointFeatureType">
                   <wn2:boundedBy i:nil="true"/>
                   <wn2:description i:nil="true"/>
                   <wn2:fid i:type="d:string">602</wn2:fid>
                   <wn2:name i:nil="true"/>
                   <wn2:elev i:type="d:float">0.0</wn2:elev>
                   <wn2:fitt_type i:type="d:string">S</wn2:fitt_type>
                   <wn2:location i:type="wn2:PointPropertyType">
```

```
        <wn2:Point i:type="wn2:PointType">
            <wn2:coord i:type="wn2:CoordType">
                <wn2:X i:type="d:decimal">1603785.469967</wn2:X>
                <wn2:Y i:type="d:decimal">647071.154604</wn2:Y>
                <wn2:Z i:nil="true"/>
            </wn2:coord>
        </wn2:Point>
    </wn2:location>
    <wn2:map_ i:type="d:string">8319SW</wn2:map_>
    <wn2:mh_code i:type="d:string">S05915</wn2:mh_code>
</wn2:PointFeatureType>
<wn2:PointFeatureType i:type="wn2:PointFeatureType">
    <wn2:boundedBy i:nil="true"/>
    <wn2:description i:nil="true"/>
    <wn2:fid i:type="d:string">607</wn2:fid>
    <wn2:name i:nil="true"/>
    <wn2:elev i:type="d:float">102.0</wn2:elev>
    <wn2:fitt_type i:type="d:string">S</wn2:fitt_type>
    <wn2:location i:type="wn2:PointPropertyType">
        <wn2:Point i:type="wn2:PointType">
            <wn2:coord i:type="wn2:CoordType">
                <wn2:X i:type="d:decimal">1603355.875</wn2:X>
                <wn2:Y i:type="d:decimal">647000</wn2:Y>
                <wn2:Z i:nil="true"/>
            </wn2:coord>
        </wn2:Point>
    </wn2:location>
    <wn2:map_ i:type="d:string">8319SW</wn2:map_>
    <wn2:mh_code i:type="d:string">S05908</wn2:mh_code>
</wn2:PointFeatureType>
<wn2:PointFeatureType i:type="wn2:PointFeatureType">
    <wn2:boundedBy i:nil="true"/>
    <wn2:description i:nil="true"/>
    <wn2:fid i:type="d:string">610</wn2:fid>
    <wn2:name i:nil="true"/>
    <wn2:elev i:type="d:float">0.0</wn2:elev>
    <wn2:fitt_type i:type="d:string">S</wn2:fitt_type>
    <wn2:location i:type="wn2:PointPropertyType">
        <wn2:Point i:type="wn2:PointType">
            <wn2:coord i:type="wn2:CoordType">
                <wn2:X i:type="d:decimal">1602905.989883</wn2:X>
                <wn2:Y i:type="d:decimal">646941.229118</wn2:Y>
                <wn2:Z i:nil="true"/>
            </wn2:coord>
        </wn2:Point>
    </wn2:location>
    <wn2:map_ i:type="d:string">8319SW</wn2:map_>
```

```
                    <wn2:mh_code i:type="d:string">S05818</wn2:mh_code>
                </wn2:PointFeatureType>
        </wn2:Layers>
        </wn0:GetLayerResponseType_Response>
    </e:Body>
</e:Envelope>
```

B.3 Sewmain request SOAP Message

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:m0="http://systinet.com/wsdl/example/webservices/gmlline/">
    <SOAP-ENV:Header>
        <m:instance xmlns:m="http://idoox.com/interface">
            <m:id>String</m:id>
            <m:setId>String</m:setId>
            <m:notFound>String</m:notFound>
        </m:instance>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <m:p0 xmlns:m="http://systinet.com/xsd/SchemaTypes/">
            <m0:extent>
                <m0:XMax>1603937.36460391</m0:XMax>
                <m0:XMin>1602657.61547023</m0:XMin>
                <m0:YMax>647191.61784469</m0:YMax>
                <m0:YMin>645803.54348906</m0:YMin>
            </m0:extent>
            <m0:name>sewmain</m0:name>
            <m0:type>POLYLINE</m0:type>
        </m:p0>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4 Sewmain response SOAP Message

```
<e:Envelope xmlns:e="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:d="http://www.w3.org/2001/XMLSchema"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wn0="http://systinet.com/xsd/SchemaTypes/"
xmlns:wn1="http://idoox.com/interface"
xmlns:wn2="http://systinet.com/wsdl/example/webservices/gmlline/">
```

```
<e:Body> <wn0:GetLayerResponseType_Response
                                    i:type="wn2:GetLayerResponseType">
          <wn2:Layers i:type="wn2:Layers">
             <wn2:boundedBy i:nil="true"/>
             <wn2:description i:nil="true"/>
             <wn2:fid i:nil="true"/>
             <wn2:name i:nil="true"/>
             <wn2:featureMember i:nil="true"/>
             <wn2:lineFeature i:type="wn2:ArrayOfLineFeatureType">
                <wn2:LineFeatureType i:type="wn2:LineFeatureType">
                   <wn2:boundedBy i:nil="true"/>
                   <wn2:description i:nil="true"/>
                   <wn2:fid i:type="d:string">711</wn2:fid>
   <wn2:address i:type="d:string">B ST - JENKINS TO IRVING</wn2:address>
   <wn2:diameter i:type="d:float">8.0</wn2:diameter>
   <wn2:fnode_ i:nil="true"/>
   <wn2:label i:type="d:string">8&amp;quot; INSF 1997</wn2:label>
   <wn2:length i:type="d:float">360.22287</wn2:length>
   <wn2:lpoly_ i:nil="true"/>
   <wn2:map_ i:type="d:string">8330NW</wn2:map_>
   <wn2:material i:type="d:string">INSF</wn2:material>
<wn2:multiCenterLineOf i:type="wn2:MultiLineStringPropertyType">
   <wn2:MultiLineString i:type="wn2:MultiLineStringType">
      <wn2:lineStringMember i:type="wn2:ArrayOfLineStringMember">
         <wn2:LineStringMember i:type="wn2:LineStringMember">
               <wn2:LineString i:type="wn2:LineStringType">
                     <wn2:coord i:type="wn2:ArrayOfCoordType">
                     <wn2:CoordType i:type="wn2:CoordType">
                        <wn2:X i:type="d:decimal">1603985.00002358</wn2:X>
                        <wn2:Y i:type="d:decimal">645837.249931257</wn2:Y>
                        <wn2:Z i:nil="true"/>
                     </wn2:CoordType>
                    <wn2:CoordType i:type="wn2:CoordType">
                        <wn2:X i:type="d:decimal">1603726.49999134</wn2:X>
                        <wn2:Y i:type="d:decimal">645586.374923023</wn2:Y>
                        <wn2:Z i:nil="true"/>
                    </wn2:CoordType>
                     </wn2:coord>
              </wn2:LineString>
                </wn2:LineStringMember>
                </wn2:lineStringMember>
                </wn2:MultiLineString>
                   </wn2:multiCenterLineOf>
                   <wn2:rpoly_ i:nil="true"/>
                   <wn2:segid i:nil="true"/>
                   <wn2:sewmain_ i:nil="true"/>
```

```
                    <wn2:sewmain_id i:nil="true"/>
                    <wn2:tnode_ i:nil="true"/>
                    <wn2:type i:type="d:string">GRAVITY</wn2:type>
                    <wn2:year i:type="d:string">1997</wn2:year>
                </wn2:LineFeatureType>
            <wn2:LineFeatureType i:type="wn2:LineFeatureType">
<wn2:fid i:type="d:string">727</wn2:fid>
<wn2:address i:type="d:string">D ST &amp;amp; IRVING</wn2:address>
<wn2:diameter i:type="d:float">8.0</wn2:diameter>
<wn2:fnode_ i:nil="true"/>
<wn2:label i:type="d:string">8&amp;quot; PVC 1989</wn2:label>
<wn2:length i:type="d:float">362.92834</wn2:length>
<wn2:lpoly_ i:nil="true"/>
<wn2:map_ i:type="d:string">8330NW</wn2:map_>
<wn2:material i:type="d:string">PVC</wn2:material>
<wn2:multiCenterLineOf i:type="wn2:MultiLineStringPropertyType">
<wn2:MultiLineString i:type="wn2:MultiLineStringType">
    <wn2:lineStringMember i:type="wn2:ArrayOfLineStringMember">
        <wn2:LineStringMember i:type="wn2:LineStringMember">
            <wn2:LineString i:type="wn2:LineStringType">
                <wn2:coord i:type="wn2:ArrayOfCoordType">
                    <wn2:CoordType i:type="wn2:CoordType">
                        <wn2:X i:type="d:decimal">1603368.624983</wn2:X>
                        <wn2:Y i:type="d:decimal">645953.875019365</wn2:Y>
                        <wn2:Z i:nil="true"/>
                    </wn2:CoordType>
                    <wn2:CoordType i:type="wn2:CoordType">
                        <wn2:X i:type="d:decimal">1603355.05659899</wn2:X>
                        <wn2:Y i:type="d:decimal">645941.732747354</wn2:Y>
                        <wn2:Z i:nil="true"/>
                    </wn2:CoordType>
                    <wn2:CoordType i:type="wn2:CoordType">
                        <wn2:X i:type="d:decimal">1603109.38299876</wn2:X>
                        <wn2:Y i:type="d:decimal">645699.913675129</wn2:Y>
                        <wn2:Z i:nil="true"/>
                    </wn2:CoordType>
                </wn2:coord>
            </wn2:LineString>
        </wn2:LineStringMember>
    </wn2:lineStringMember>
  </wn2:MultiLineString>
</wn2:multiCenterLineOf>
<wn2:rpoly_ i:nil="true"/>
<wn2:segid i:nil="true"/>
<wn2:sewmain_ i:nil="true"/>
<wn2:sewmain_id i:nil="true"/>
<wn2:tnode_ i:nil="true"/>
```

```
    <wn2:type i:type="d:string">GRAVITY</wn2:type>
    <wn2:year i:type="d:string">1989</wn2:year>
 </wn2:LineFeatureType>
            </wn2:Layers>
         </wn0:GetLayerResponseType_Response>
    </e:Body>
</e:Envelope>
```

C.1 QueryWorkorder SOAP Request

```
<S:Envelope S:encodingStyle=""
      xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:a="http://www.erc.msstate.edu/MobileService"
      xmlns:XS="http://www.w3.org/2001/XMLSchema"
      xmlns:XI="http://www.w3.org/2001/XMLSchema-instance">
   <S:Body>
      <a:WorkOrderRequest>
          <a:InspectorName
              XI:type="XS:string">VenuK</a:InspectorName>
        </a:WorkOrderRequest>
     </S:Body>
  </S:Envelope>
```

C.2 QueryWorkorder SOAP Response

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<WorkOrder xmlns="http://www.erc.msstate.edu/MobileService">
 <workorderid>380319170105</workorderid>
<createdOn>3/23/2004 1:29:41 PM</createdOn>
<description>Inspect redline feature and perform testing</description>
<priority>EMERGENCYCALL</priority>
<workOrderStatus>NOTSTARTED</workOrderStatus>
<InspectionDate>NA</InspectionDate>
<actionTaken>NA</actionTaken>
<comments>NA</comments>
<map>
<Projection>
<proj>PROJCS["NAD_1927_StatePlane_Washington_North_FIPS_4601",GEOGCS["G
CS_North_American_1927",DATUM["D_North_American_1927",SPHEROID["Clarke
_1866",6378206.4,294.9786982]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532
92519943295]],PROJECTION["Lambert_Conformal_Conic"],PARAMETER["False_Ea
```

sting",2000000],PARAMETER["False_Northing",0],PARAMETER["Central_Meridian"
,-
120.8333333333333],PARAMETER["Standard_Parallel_1",47.5],PARAMETER["Stand
ard_Parallel_2",48.73333333333333],PARAMETER["Latitude_Of_Origin",47],UNIT["
Foot_US",0.30480060960121924]]</proj>
</Projection>
<redlineFeature>
<multiCenterLineOf xmlns="**http://www.opengis.net/gml**">
<MultiLineString>
<lineStringMember>
<coord>1603420.50261505 647441.394636751,1603278.01608692
647302.459852621,1603167.02920682 647194.287436521</coord>
<fid>878</fid>
<fnode_>2976</fnode_>
<tnode_>3032</tnode_>
<lpoly_>2</lpoly_>
<rpoly_>2</rpoly_>
<length>353.99259</length>
<sewmain_>1529</sewmain_>
<sewmain_id>9392</sewmain_id>
<segid>05909 05905</segid>
<type>GRAVITY</type>
<diameter>8</diameter>
<material>VIT</material>
<address>LOGAN & H ST</address>
<map_>8319SW</map_>
<year>1892</year>
<label>8" VIT 1892</label>
</lineStringMember>
<lineStringMember>
<coord>1603234.32053488 647623.59662092,1602980.40059864
647376.56922869</coord>
<fid>892</fid>
<fnode_>2924</fnode_>
<tnode_>2989</tnode_>
<lpoly_>2</lpoly_>
<rpoly_>2</rpoly_>
<length>354.25675</length>
<sewmain_>1498</sewmain>
<sewmain_id>6597</sewmain_id>
<segid>05904 05903</segid>
<type>GRAVITY</type>
<diameter>8</diameter>
<material>VIT</material>
<address>I ST & LOGAN ST TO I ST & KEARNEY ST</address>
<map_>8319SW</map_>
<year>1892</year>

```
            <label>8" VIT 1892</label>
          </lineStringMember>
        </MultiLineString>
      </multiCenterLineOf>
    </redlineFeature>
    <pointFeature>
        <multiLocation xmlns="http://www.opengis.net/gml">
          <MultiPoint>
            <pointMember>
              <coord>1602858.25,648012.5625</coord>
              <fid>503</fid>
              <mh_id>U0033-S11</mh_id>
              <fitt_type>S</fitt_type>
              <elev>97.53</elev>
              <map_>8319SW</map_>
            </pointMember>
            <pointMember>
            <coord>1603171.828418,647936.206351</coord>
              <fid>509</fid>
              <mh_id>S08402</mh_id>
              <fitt_type>S</fitt_type>
              <elev>0</elev>
              <map_>8319SW</map_>
            </pointMember>
            <pointMember>
            <coord>1603675.864721,647920.721049</coord>
              <fid>512</fid>
              <mh_id>S08416</mh_id>
              <fitt_type>S</fitt_type>
              <elev>0</elev>
              <map_>8319SW</map_>
            </pointMember>
          </MultiPoint>
        </multiLocation>
      </pointFeature>
      <lineFeature>
      <multiCenterLineOf xmlns="http://www.opengis.net/gml">
        <MultiLineString>
          <lineStringMember>
            <coord>1603948.61762354 646866.184332215,1603713.48341532
            646637.503628002</coord>
              <fid>825</fid>
              <fnode_>3122</fnode_>
              <tnode_>3178</tnode_>
              <lpoly_>2</lpoly_>
              <rpoly_>2</rpoly_>
              <length>327.99847</length>
```

```
      <sewmain_>1646</sewmain_>
      <sewmain_id>9379</sewmain_id>
      <segid>06006 05914</segid>
      <type>GRAVITY</type>
      <diameter>8</diameter>
      <material>VIT</material>
      <address>E ST - LOGAN TO KEARNEY</address>
      <map_>8319SW</map_>
      <year>1905</year>
      <label>8" VIT 1905</label>
    </lineStringMember>
  <lineStringMember>
    <coord>1603970.39778356 646886.336268234,1604141.25000772
        646707.187468067</coord>
      <fid>832</fid>
      <fnode_>3114</fnode_>
      <tnode_>3168</tnode_>
      <lpoly_>0</lpoly_>
      <rpoly_>0</rpoly_>
      <length>247.55754</length>
      <sewmain_>6360</sewmain_>
      <sewmain_id>9368</sewmain_id>
      <segid>06005 06010</segid>
      <type>GRAVITY</type>
      <diameter>8</diameter>
      <material>VIT</material>
      <address>LOGAN & D ST TO LOGAN & E ST</address>
      <map_>8319SW</map_>
      <year>1892</year>
      <label>8" VIT 1892</label>
    </lineStringMember>
  <lineStringMember>
    <coord>1602905.98991057 646941.229132285,1602649.72123833
        646691.843532053</coord>
      <fid>835</fid>
      <fnode_>3106</fnode_>
      <tnode_>3171</tnode_>
      <lpoly_>2</lpoly_>
      <rpoly_>2</rpoly_>
      <length>357.58469</length>
      <sewmain_>1643</sewmain_>
      <sewmain_id>10986</sewmain_id>
      <segid>05818 05815</segid>
      <type>GRAVITY</type>
      <diameter>8</diameter>
      <material>VIT</material>
      <address>JENKINS & H ST.</address>
```

```
                        <map_>8319SW</map_>
                        <year>1892</year>
                        <label>8" VIT 1892</label>
                    </lineStringMember>
                </MultiLineString>
            </multiCenterLineOf>
        </lineFeature>
    </map>
</WorkOrder>
</soap:Body>
</soap:Envelope>
```