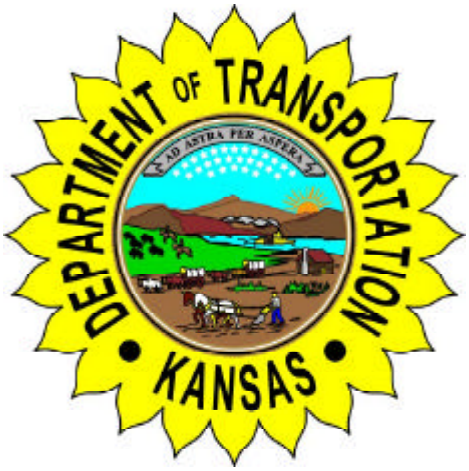
Report No. K-TRAN: KSU-00-5
Final Report

KNOWLEDGE MANAGEMENT TECHNOLOGIES

Virgil Wallentine
William Hankley
Maria Zamfir-Bleyberg

Kansas State University
Manhattan, Kansas



August 2000

K-TRAN

A COOPERATIVE TRANSPORTATION RESEARCH PROGRAM BETWEEN:
THE KANSAS DEPARTMENT OF TRANSPORTATION
KANSAS STATE UNIVERSITY
THE UNIVERSITY OF KANSAS

Knowledge Management Technologies

K-TRAN Report

submitted to

Kansas Department of Transportation

by

Virgil Wallentine (virg@cis.ksu.edu**)**
William Hankley (hankley@cis.ksu.edu**)**
Maria Zamfir-Blyberg (maria@cis.ksu.edu**)**

Department of Computing and Information Sciences
Kansas State University
Manhattan, KS

August 2000

1. Report No. K-TRAN: KSU-00-5	**2. Government Accession No.**	**3. Recipient Catalog No.**
4 **Title and Subtitle** KNOWLEDGE MANAGEMENT TECHNOLOGIES		**5 Report Date** August 2000 **6 Performing Organization Code**
7. Author(s) Virgil Wallentine, William Hankley and Maria Zamfir-Bleyberg		**8 Performing Organization Report No.**
9 Performing Organization Name and Address Kansas State University Department of Computing and Informational Sciences Manhattan, Kansas 66506		**10 Work Unit No. (TRAIS)** **11 Contract or Grant No.** C-1147
12 Sponsoring Agency Name and Address Kansas Department of Transportation Docking State Office Bldg. Topeka, Kansas 66612		**13 Type of Report and Period Covered** Final Report June 1999 to August 2000 **14 Sponsoring Agency Code** 106-RE-0194-01

15 Supplementary Notes

16 Abstract

This report presents recommendations for technologies that have potential to contribute to future development and management of Kansas Department of Transportation (KDOT) information systems. The recommendations are accomplished by examples of designs, software demonstrations, and discussion of experiences using these technologies. The recommendations cover the following topics:

(1) use of object models to capture and to present the structure and processing of KDOT information; use of the Unified Modeling Language (UML) and tools to support UML.

(2) a structure for flexible, network-based access for KDOT data; use of Java servlet technology and eXtensible Markup Language representation of shared data structures.

(3) use of Visual Basic scripts to integrate Office 2000 tools with stored KDOT data; use of Visual Basic scripts and define and enforce KDOT procedures for workflow processing within a network-based environment.

(4) structure of a virtual data warehouse for KDOT project and management information for use in studying deeper relationships in existing data.

This report, together with all of the supporting examples and references are available on-line at the KSU/KDOT website: http: www.cis.ksu.edu/kdot.

17 Key Words Data warehouse, EXtensible Markup Language (XML), Unified Modeling Language, Visual Basic scripts		**18 Distribution Statement** No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161	
19 Security Classification (of this report) Unclassified	**20 Security Classification (of this page)** Unclassified	**21 No. of pages** 42	**22 Price**

PREFACE

This research project was funded by the Kansas Department of Transportation K-TRAN research program. The Kansas Transportation Research and New-Developments (K-TRAN) Research Program is an ongoing, cooperative and comprehensive research program addressing transportation needs of the State of Kansas utilizing academic and research resources from the Kansas Department of Transportation, Kansas State University and the University of Kansas. The projects included in the research program are jointly developed by transportation professionals in KDOT and the universities.

NOTICE

The authors and the State of Kansas do not endorse products or manufacturers. Trade and manufacturers names appear herein solely because they are considered essential to the object of this report.

This information is available in alternative accessible formats. To obtain an alternative format, contact the Kansas Department of Transportation, Office of Public Information, 7th Floor, Docking State Office Building, Topeka, Kansas, 66612-1568 or phone (785)296-3585 (Voice) (TDD).

DISCLAIMER

The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the views or the policies of the State of Kansas. This report does not constitute a standard, specification or regulation.

Summary

This report presents recommendations for technologies that have potential to contribute to future development and management of Kansas Department of Transportation (KDOT) information systems. The recommendations are accompanied by examples of designs, software demonstrations, and discussion of experiences using these technologies. The recommendations cover the following topics:

(1) use of object models to capture and to present the structure and processing of KDOT information; use of the Unified Modeling Language (UML) and tools to support UML.

(2) a structure for flexible, network-based access for KDOT data; use of Java servlet technology and eXtensible Markup Language (XML) representation of shared data structures.

(3) use of Visual Basic scripts to integrate Office 2000 tools with stored KDOT data; use of Visual Basic scripts and define and enforce KDOT procedures for workflow processing within a network-based environment.

(4) structure of a virtual data warehouse for KDOT project and management information for use in studying deeper relationships in existing data.

This report together with all of the supporting examples and references are available on-line at the KSU/KDOT web site http://www.cis.ksu.edu/kdot.

Table of Contents

List of Figures

List of Mnemonics

ADO	Active Data Objects, a Microsoft control for database access
CIS	Computing & Information Sciences, Department at KSU
CMS	KDOT Construction Management System
CPMS	KDOT Comprehensive Program Management Systems
DOC	Word document format
EJB	Enterprise Java Beans; plugable components at the server side
FIMS	KDOT Financial Information Management System
HTML	Hypertext Markup Language, for web pages
HTTP	Hypertext Transfer Protocol, for web data transfer
IBM	International Business Machines, a computer company
JDBC	Java DataBase Connectivity
KDOT	Kansas Department of Transportation
KSU	Kansas State University
LAN	a local area network
MS	CIS Master of Science degree
MSE	CIS Master of Software Engineering degree
OLE	a Microsoft mnemonic
OO4O	an OLE driver for Oracle databases
PDA	personal digital assistant device, such as Palm Pilot
QWERTY	standard layout for keys on a keyboard
RTF	Rich Text Format, generic Microsoft document format
SQL	Sequential Query Language, for operations on databases
TCP/IP	common low-level protocols for internet data transfer
UML	Unified Modeling Language, for documenting software design
VB	Visual Basic, a Microsoft programming system
WAP	Wireless Application Protocol, for wireless devices
WML	Wireless Markup Language, for documents for wireless devices
XML	eXtensible Markup Language, for data description
XSL	eXtensible Style sheet Language, for style sheets for XML

Overview and Recommendations

Introduction

In any large enterprise such as KDOT, a certain part of corporate knowledge resides in computer based information and in the human handling of that information – handling such as the organization and documentation of the information, communication and presentation, sharing of information by collaboration and the ordered flow of processing, and many other activities. The use of technologies for management of and access to such information is essential for effective operation of the enterprise. Such technologies change rapidly, so that investigation of new directions must be conducted regularly. In particular, the internet provides many new forms for sharing and for visualizing corporate information. Internet technologies can make business work more efficient and effective, but it can also be a burden just to track new technologies and tools.

Under support of the K-TRAN program, the Computer and Information Sciences Department (CIS) at Kansas State University has examined some aspects of knowledge and information management technologies that have potential either to be incorporated in future systems within KDOT or to be used in the management of such systems. During the summer of 1999, we visited with managers and users of information systems at KDOT. We examined the structure and function of the Comprehensive Program Management System (CPMS), Construction Management System (CMS), and Financial Information Management System (FIMS). We viewed examples of KDOT data marts. We listened to explanations of need for new features for executive level decision-making. Based upon these interviews, we identified several technologies that have potential to contribute to enhancement of either the management or usefulness of KDOT information systems. During the academic year 1999-2000, we worked with several graduate students to develop small tutorial examples that illustrate the technologies that we considered. Based upon that work, we recommend development of these newer technologies within KDOT.

Recommendations

The recommendations and examples are listed below and then explained within the chapters of the report. In all of the cases, the examples are quite small; they do not reflect actual KDOT data. As a practical matter, a first step towards adopting new technologies should be to develop a larger prototype specifically tailored to KDOT systems and using existing KDOT.

Recommendation 1:

Develop a Conceptual Model of KDOT Information Structure and Processing

A foundation discipline for maintaining oversight of a corporate knowledge structure is to maintain a conceptual model of overall structure of and relationships of information and enterprise operations. The Unified Modeling Language is the defacto common notation for writing system models. We recommend development of a conceptual model for

KDOT information and operations, particularly to cover information within existing databases and information that is now maintained in various ad hoc ways outside of the central databases. We illustrate the UML notation with a small conceptual object model of some KDOT information for use for long-term tracking and management of road and funds information.

Recommendation 2:

Provide Network-Based Access for KDOT Data using Java with XML

One area of rapid growth of the internet is for electronic commerce, broadly including internal corporate sharing of information, business-to-business communication, and business-to-client communication. There are several technologies for providing network access to database information and for providing web information. We recommend using Java servlet architecture for providing access to information in KDOT databases and using XML as the common format for exchange of information between diverse applications. Such technologies are likely to be the easiest to maintain for use by different applications within a networked system. We illustrate the structure of a web-based access using Java servlet architecture together with XML.

Recommendation 3:

Develop Scripts for Office 2000 to Automate and Standardize Routine Operations

KDOT staff indicated that Office 2000 tools likely will be used (or may already be used) for corporate work processing. Office 2000 tools can be used to access corporate data stored in shared databases. However, such use requires development of supporting scripts (which are Visual Basic macros). The development of such scripts is not as straightforward as just using the Office 2000 tools. We recommend that before fully using Office 2000 tools to access information in KDOT databases there should be a development to identify common queries and to build and evaluate corresponding Visual Basic database access scripts. We demonstrate scripting features for Office 2000 tools for user access to data items in a remote database system.

A second area of use of scripts for Office 2000 tools is to manage the orderly handling of data items and transactions, often referred to as managing workflow. Normal procedures for handling work items can be represented visually as one component of a process model (which is included in recommendation 1). The ordering of work procedures can also be represented in textual form by what are called production rules. Compliance with written rules of workflow can be checked and enforced by scripts associated with either documents or forms, which are handled by Office 2000 tools. Before allowing input of transactions via the internet, we recommend a period of development for documentation of workflow rules for KDOT operations and coding of corresponding scripts for workflow rules. We demonstrate an example of Visual Basic scripts for enforcement of workflow rules for entering transactions for a simple warehouse.

Recommendations 4:

Develop a Data Warehouse for Support of Research About KDOT Data

At present, information for regular KDOT operations is maintained in several separate databases and the databases contain a large amount of detail information. Consequently, stored data is not easily usable for executive research and decision-making. Two steps towards making such information more usable are (i) to provide convenient interactive access via the KDOT intranet and (ii) to develop a virtual data warehouse for study of extended data relationships. (Recommendation 2 dealt with network access.) We also recommend development of a data warehouse to be connected to archival copies of CPMS and CMS information. The warehouse would be used for investigating past trends and relationships within historical data. It would allow interactive analysis of the data and use of data mining tools. We illustrate a conceptual structure for a data warehouse for information in existing CPMS, CMS, and FIMS systems.

Results

This project provided support in part for five graduate students. The reports and source code from their projects are available for review.

Dongsheng Zhu (MSE 1999) developed the initial recommended structure for a KDOT data warehouse (Recommendation 3). His Master of Software Engineering (MSE) project implemented a similar data warehouse for the KSU Library.

Nelson Terrazas's (MSE 1999) work was influenced and directed by the KSU/KDOT team. His MSE project implemented a web-based data mart using Enterprise Java Beans executing within an Apache web server. He also demonstrated rapid application development (RAD) of Java clients using data aware components within the Jbuilder tool.

Prapoorna Garimella (MSE 2000) developed the web based KDOT model using XML as described in Recommendation 2. That model is further described in her MSE report.

Ahmed Yasien (MSE 2000) implemented the Visual Basic front end for the product warehouse Access database (Recommendation 3). The key feature of his project is that the user is constrained to follow the correct workflow ordering of transaction operations.

Deep Kapadia (Master of Science (MS) expected 2001) developed the demonstration of wireless access to web data. His MS work will focus on tools and protocols for wireless access to web data.

Finally, there is a secondary effect from these projects. The examples from master's projects will soon be shown as examples and prototypes for assignments in undergraduate classes. Many of the undergraduate students then elect to work in and around Kansas.

Future Work

As noted above in the recommendations, the examples developed as part of this study are small tutorial examples which do not reflect actual KDOT data. If any of the recommended technologies are to be developed at KDOT, it is possible to use available commercial tools or to develop systems "in house." In either case, but particularly for "in house" development, we recommend, as a first step, the development of a prototype specifically tailored to KDOT systems, using existing KDOT data, forms, procedures, and staff. Towards that end, we request additional work to be performed jointly with KDOT staff to develop the following:

(i) a KDOT object model for overall road maintenance, budgeting, and contracting,
(ii) a specific workflow model for some aspects of common transaction processing,
(iii) a separate KDOT data warehouse and initial explorations of that data,
(iv) an XML enabled database with access by both Java clients and Office 2000 clients.

We suggest the development of a laboratory of software models and prototypes that could be studied for usability and performance; these issues were not addressed within this report.

Recommendation 1:

Develop a conceptual model of KDOT information and processing

In managing shared knowledge within an enterprise, it is necessary to have both information technologies (such as database systems, networking systems, and desktop applications software) and mastery of a higher view of the overall information and processing that comprise the enterprise functions. Such a higher view may be called a conceptual model. A conceptual model identifies general functions and information used by the enterprise, but without the burden of details of how software and data are actually implemented. The key aspect of a conceptual model is that it is an abstraction at the level of an executive summary. It allows managers to maintain awareness of essential capabilities and limitations of the enterprise. It also allows technical staff to see the exact structures that managers expect. A conceptual model is an essential vehicle for effective communication about software and business systems. A second important aspect of a conceptual model is that it can encompass information and processing that may be handled manually as part of the decision-making activities of the enterprise.

The remainder of this chapter presents a tutorial summary of the Unified Modeling Language (UML) notation, which is the most commonly used to document models of information systems.

UML Views

Conceptual models are expressed as visual models. They are comprised of various kinds of diagrams and supporting explanations that represent different aspects of a system. Currently, the most commonly accepted notation for such models is the Unified Modeling Language (UML [1,2]), which presents an object-oriented structure. UML defines at least 10 different kinds of diagrams. Each kind of diagram represents a different aspect of view of the overall model. Some of the diagrams are more related to implementation and deployment of software. Five aspects of conceptual level models are listed and illustrated below. Example diagrams referenced below are taken from the UML 1.3 Documentation [1]. These diagrams do not represent KDOT systems; they are presented just to show the concept of the modeling language. (A UML model for KDOT is presented later in this chapter.)

> Use-case diagrams: These show different users (or clients for a system) and modes of usage, e.g. Figure 1a. The users are called actors. Specific relations between actors may be shown as an Actor-Relationship diagram, Figure 1b. There may be many use-case diagrams corresponding to the many different uses of the total information system.
>
> Class diagrams: These are somewhat similar to entity-relation models; they show the major static structures of information storage. In object-oriented vocabulary, classes are the defining entities that represent information structure and related operations. A class can represent a software component or entity within the enterprise. A tiny class diagram

is shown in Figure 2a. The rectangles represent classes. Generally, each class will have three sections, its name, data items (called fields), and operations (called methods). Figure 2a represents an information structure, but no specific operations. More realistically, a class diagram for even one aspect of an enterprise will represent many component classes. A larger diagram from a recent student project is shown in Figure 2b; it represents the concepts involved in class diagram, so it is a kind of self-reflective model. Class diagrams are typically the largest and most encompassing of the components of a software model because they contain information that is used in all of the other views of the model. There may be many class diagrams corresponding to the many different focuses of details of information within a total enterprise. Class diagrams are called static diagrams, because they show the fixed structure of the model. (By contrast, the other constituent diagrams represent the dynamic behavior associated with various uses or activities represented in the composite model.)

Scenario diagrams: These show common sequences of substeps of processing operations. For example, Figure 3 shows the steps of processing for the operation of a customer returning a video to a video rental store. The steps of the operation occur in time order as shown. The scenario corresponds to one case of the use-case model for the video store. The objects shown along the top row are instances of the classes in the class model. Each operation shown on each horizontal arrow corresponds to an operation of the class at the tip end of the arrow. There should be many scenario diagrams corresponding to many use-cases and suboperations within use-cases. Scenario diagrams present a visual "walkthru" of normal processing for each "transaction" within an enterprise.

Collaboration diagrams: These show how different operations and data interact together. A collaboration can be shown as multiple actors in a use-case diagram, as in Figure 1. Alternately, the interaction of two operations can be shown in the form of a sequence diagram with starting operations from both the left and the right sides of the diagram.

Activity diagrams: These show temporal coordination of operations, e.g. Figure 4a. These typically represent a workflow structure. In cases where work activities are split between different departments, activities are shown in what are called swim lanes, as in Figure 4b.

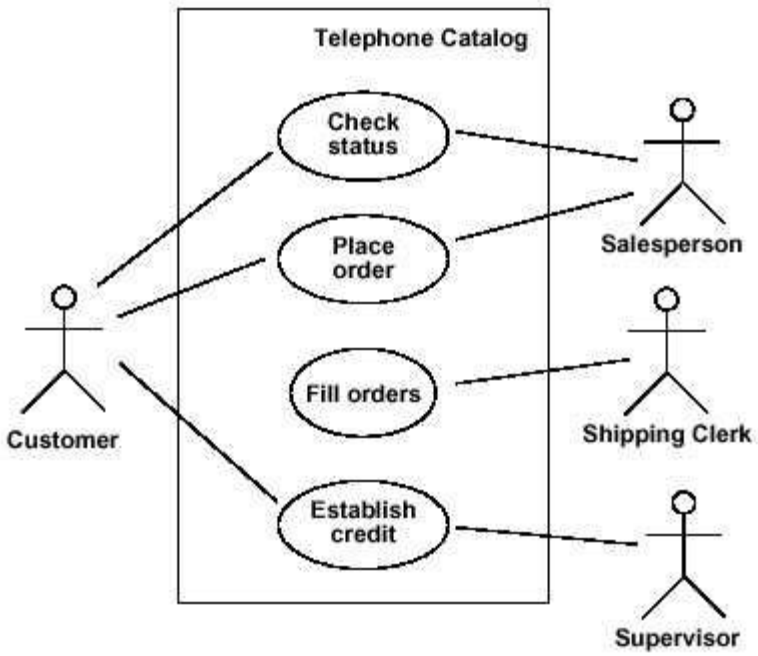


Figure 1a. Use Case Diagram

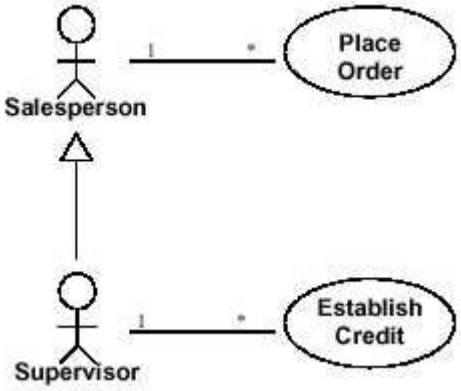


Figure 1b. Actor Relationship

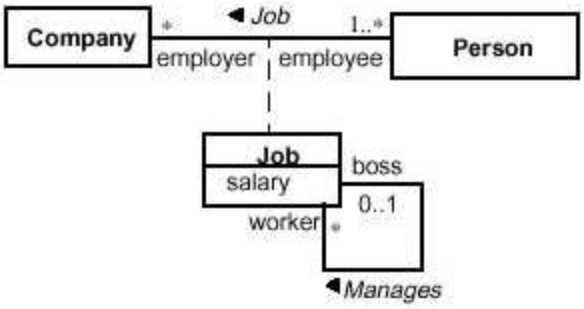


Figure 2a. A Small Class Diagram

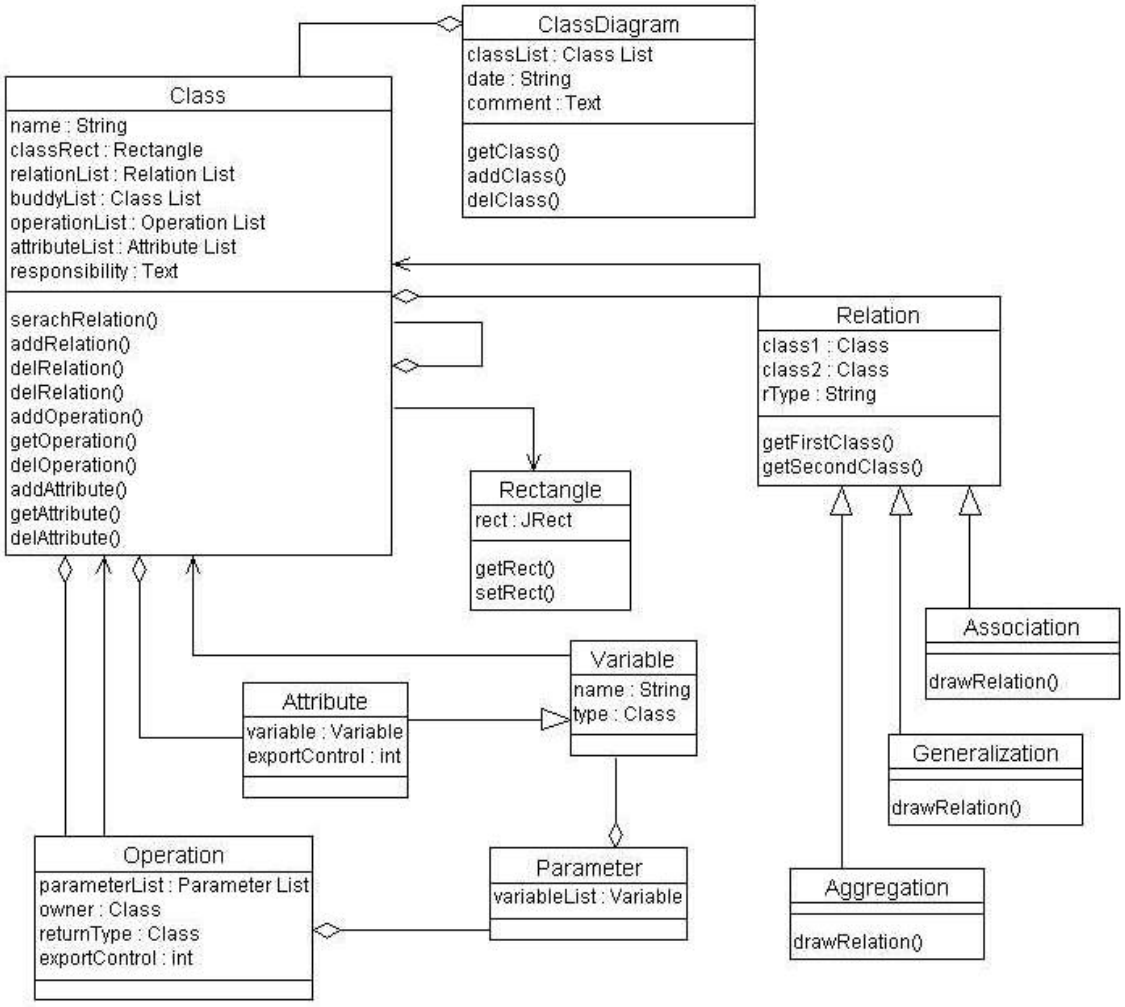


Figure 2b. A larger class diagram

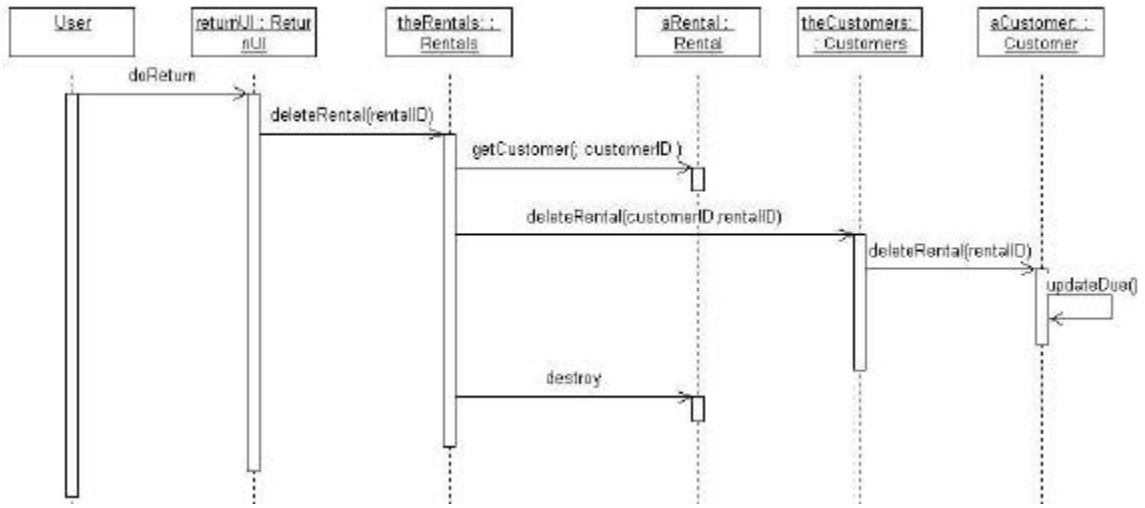


Figure 3. A scenario for a video rental store.

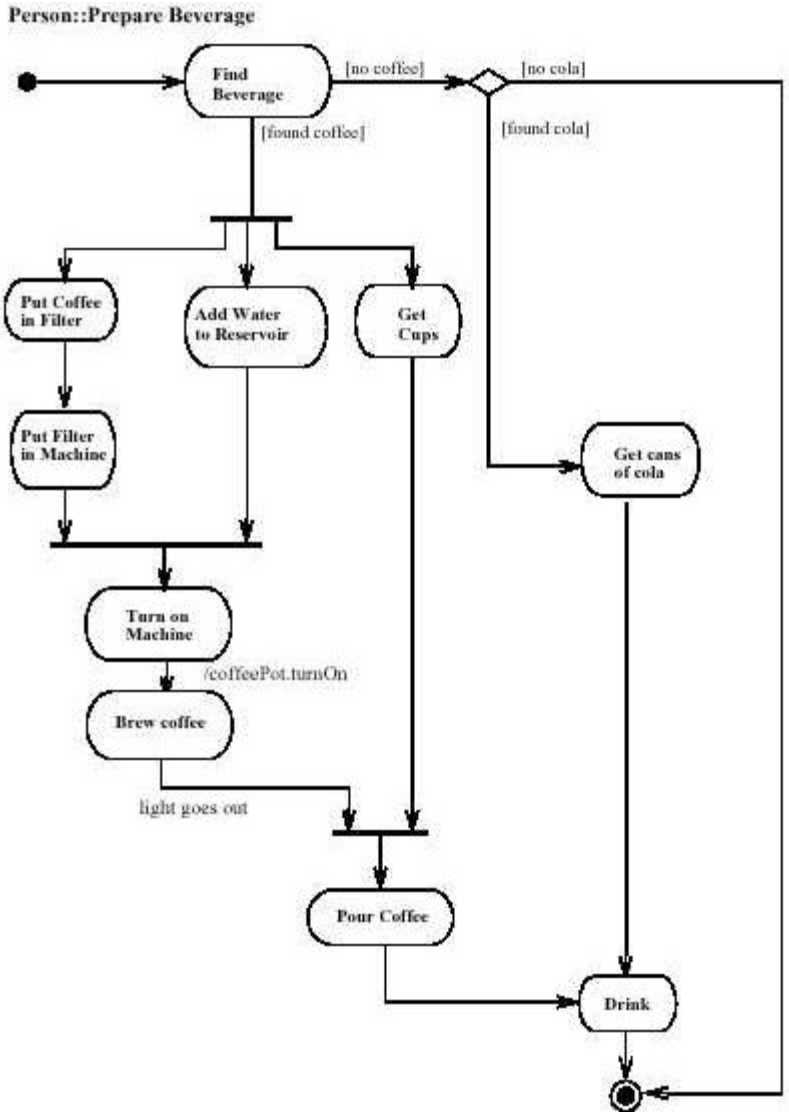


Figure 4a. Activity diagram.

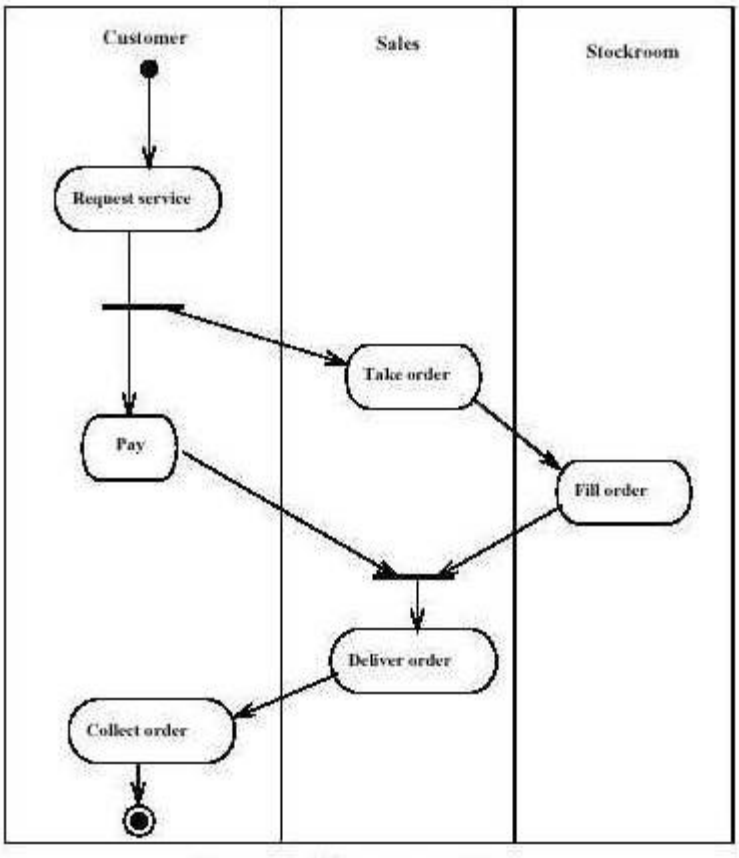


Figure 4b. Swim lane diagram.

Tools for building models

There are several software tools for drawing and viewing UML diagrams. The most common tool is Rose (from Rational Corporation [3]). Another tool, which seems easier to use, but which is less well known, is the Together products (TogetherJ and TogetherC, from TogetherSoft [5]). All of those tools provide capabilities to generate a program framework code from the design diagrams, but that capability is not required for current KDOT work. There are also free tools such as Argo/UML [6]. Finally, if models are at the conceptual level and are not directly related to program code, then simple drawing tools can be used (such as Visio, Word, etc.).

Conceptual Models

UML diagrams can be used to represent models at different levels of abstraction. At a concrete level, object models are intended to represent the structure of an existing or proposed software system. At this level, classes and methods of the model correspond directly to classes and methods of the software system. There are many texts about building software models. At an abstract level, object models are intended to represent the structure of an existing or proposed business system, which would be called a

conceptual model. At this level, classes and methods of the model need not correspond directly to specific software structure, but they correspond to the functional structure of the enterprise. This is the kind of model that we recommend be developed within KDOT. There are several texts about building business models, for example Business Modeling with UML [5].

An Example Model

Now, consider what a conceptual model for KDOT information processing might entail. KDOT deals with a number of collections of conceptual entities such as roads, road sections, geographical regions, governmental units, budgets, funds, contracts, plans, vouchers, payments, reference documents, various kinds of reports, maps, graphs, and more. To have full oversight of operations it is necessary to know the meaning of all these concepts, the constraints that apply to them, the relationships between them, and the operations for collecting data and the resulting reports and actions. A conceptual model would capture in visual form an overview of all such information.

In visiting with KDOT, we found that there was not a current model of the overall information and processing. Current KDOT database systems record some information about these entities, but not all information that is used is stored in such databases. The bulk of information relating to projects and contracts is stored within CMS and CPMS. There were some schema models (software models) for the database systems, but even for those models some aspects were not clearly defined or not consistent with the current implementation. Certain executive level processing of information is done by individuals using local spreadsheets and computations recorded on paper. For example, some planning for funds investment and early planning of repairs are now done outside of the CMS and CPMS systems. Now, an organization can function effectively in its normal operations even without an explicit overall model. However, planning for new features and processing will be hampered.

To illustrate UML, we present as an example, a partial class model for roads and funds planning, shown in Figure 5. It is a hypothetical model; it does not represent the actual structure of KDOT operations or even KDOT vocabulary. It is intended to represent information structures necessary for projecting costs for future roads maintenance and for managing "idle funds," which are two items not handled with the current KDOT databases. Indeed, even though appropriate vocabulary is well known within KDOT, there was not a model nor documentation available as reference. The model is presented to illustrate the structure of the kind of conceptual model that could be developed. The model was constructed using Rational Rose.

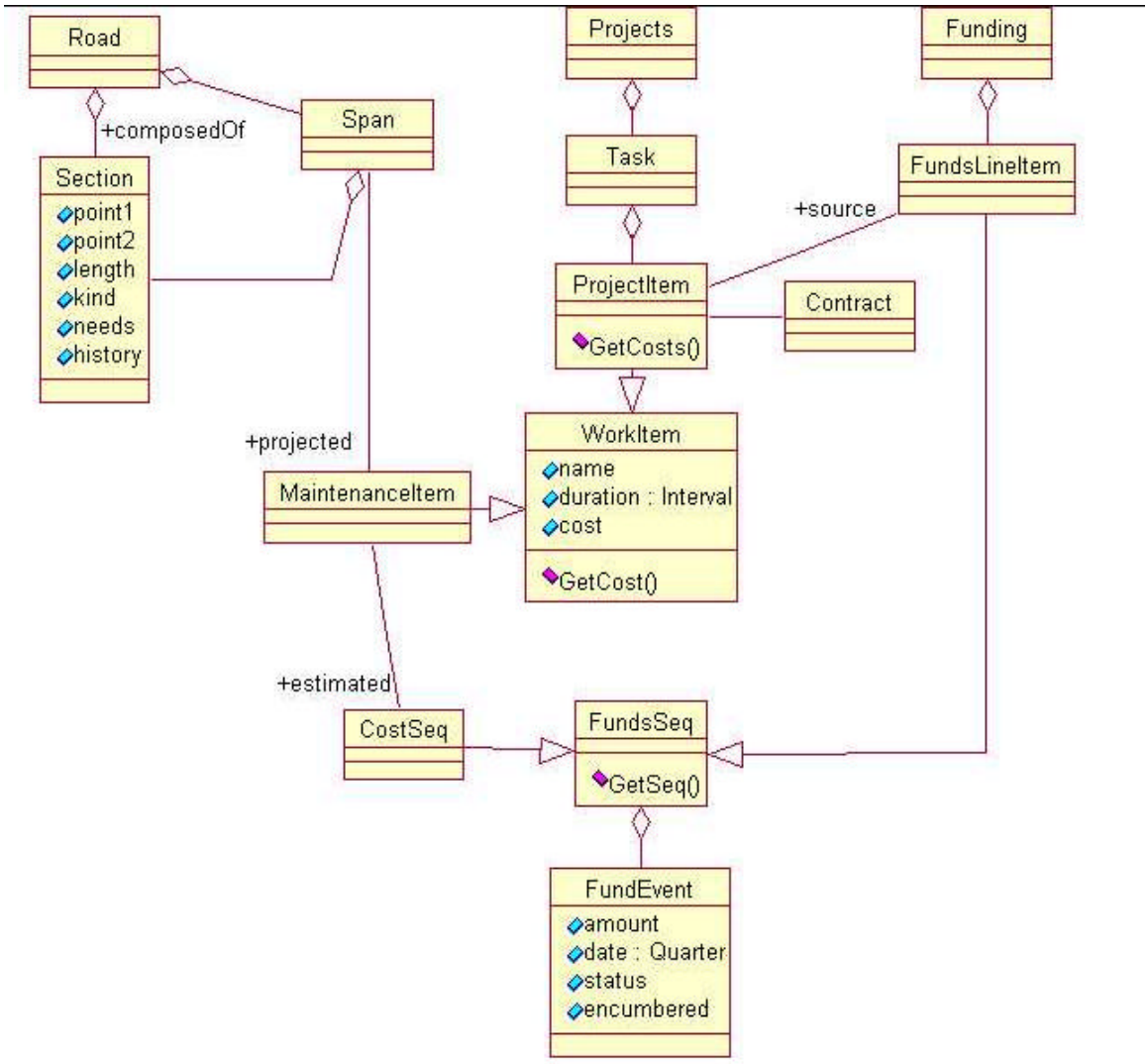


Figure 5. Maintenance and Funds Class Model

The model defines classes and relations that may be involved with funds management or projection of road maintenance. Classes encompass both data structures, which are listed in the center band of each class and a few methods (or operations), which are the "Get-info" lines in the bottom band of some of the classes. In discussing the model, a class has a single name, such as "Road." Multiple instances of class Road are referred to as "Road's," since "Roads" may be a different class name of a collection of Road's. The class model shows:

Road's are composed of contiguous Section's .
A Span is a sequence of contiguous Section's with projected maintenance.
MaintenanceItem's are a kind of WorkItem.

A FundsSeq is a sequence of FundEvent items,
each FundEvent has an amount, date, and status.
WorkItem's have an associated estimated total cost,
which is the sum of the associated CostSeq.

Project's are composed of Task's,
which are comprised of several ProjectItem's,
which are a kind of WorkItem.
ProjectItem's have an associated funding source and contract.
The funding source is itself a FundsSeq.

Now, the meaningfulness of the class diagram can be established only by defining use-cases and by building scenarios for common operations. This has not been done. It requires joint work by persons familiar with KDOT vocabulary, functions, and persons familiar with modeling structure and notation. Possible actors and operations may be:

Planning Manager to define Road maintenance items.
Project Manager to assign Project components.
Funding Manager to define fund sources.
Investment Manager to assign investments of funds.

We recommend that this kind of modeling be developed for KDOT operations, both for current database systems and for overall operations including current manual operations. Such a model will provide a documentation of current capabilities and a roadmap for new information management.

Recommendation 2:

Support web-based access for KDOT Information;
use XML representation of data;
consider Java servlets for middle tier architecture.

A most phenomenal development in computing has occurred in recent years. That has been the enhancement of business functions and services through use of internet and intranet communication for all kinds of electronic commerce, from merely providing information to potential clients, to supporting on-line transactions for corporate business processing. These later services go far beyond merely providing passive web pages; they provide interactive access to and sharing of the corporate knowledge structure and operations, in effect, allowing distributed management of the shared corporate knowledge. In this report we address two different aspects of such electronic commerce; first (in this section) discuss the architecture and protocols for providing information to network users and second (in the following section) we present features of personal computer software to support shared use of corporate information. Client KDOT workers might access shared information by using web browsers, by using Office 2000 tools (such as Word), or by using specialized software (such as Visual Basic programs). In this section, we focus on examples using a web browser. The next section considers features of other client software. We recommend two technologies that are part of "serving" shared information, Java servlets as a server architecture and XML as an information format.

KDOT has the need to provide distributed access to its knowledge base through LAN-connected desktops, wireless mobile laptops, and narrow band devices such as personal digital assistants, cell phones and pagers. Thus, the underlying knowledge management infrastructure must support all of these distributed technologies. We suggest the architecture shown in Figure 6.

Knowledge workers who need access to the legacy data from their web browser can access it through a LAN (which may be hard-wired or mobile) or they access it through Office 2000 applications such as Word, PowerPoint, and Excel. Traditional TCP/IP protocol supports this connection and the XML interpreter in the Server does the translation from XML to HTML, RTF, and/or Doc format. Mobile narrowband devices such as PDAs, pagers, and cell phones also can access the database by using the Wireless Application Protocol to talk to the server and using Wireless Markup Language (the de factor standard for mini-browsers on these portable devices).

The server (for example, implemented using the Apache server) communicates to the "thin clients" using HTTP for HTML service and WAP for WML service. It uses standard TCP/IP protocol to communicate the XML documents to and from the Enterprise Java Beans server. The EJB serves to coordinate multiple streams to/from the legacy database through a JDBC interface and the TCP/IP protocol. Using this strategy we can deploy distributed access to the knowledge base in multiple configurations from one centralized system to putting each function on a separate server.

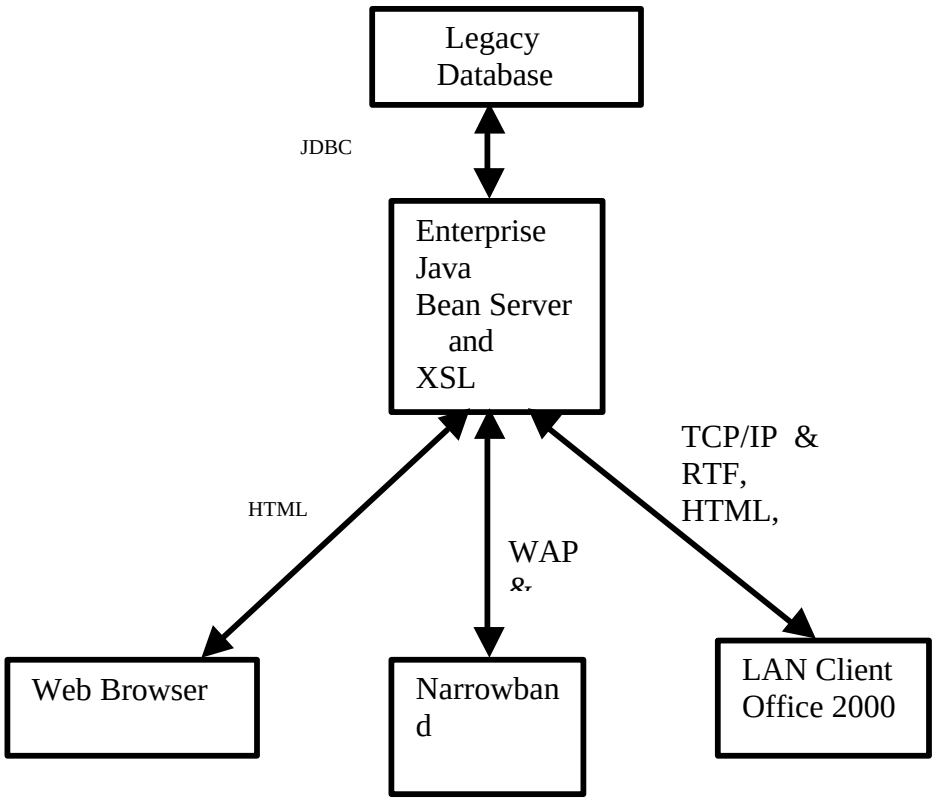


Figure 6. Distributed Access Architectures

Java Bean Architecture

The multi-tier architecture for distributed access for business information is well established. But, there are several competing technologies for the middle-ware software component, notably, Java Enterprise Beans, Microsoft Active Server Pages (using Visual Basic Scripts), and "third party" middleware products. There are both technical and non-technical arguments for each of these alternatives. From the consideration of software development, we feel that the Java language offers the best capabilities. Java is a powerful language that will be well know by many skilled programmers and it is supported by many software development tools.

XML

The Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data, which may range from web pages, to documents, to information from legacy databases. In a limited way, XML description of data is analogous to the HTML description of document data. The "extensible" aspect of XML resides in that XML provides a framework for declaring new descriptors (or tags) for data

structures. Likely XML descriptions of data objects will be adopted by most internet programs. Unlike HTML, XML describes only the structure of information, not the form of presentation to users. This separation of data from presentation enables the seamless integration of data from diverse sources as well as the shared use of data by diverse client programs. Within browsers, data can by viewed in different forms by applying different style sheets.

XML will be valuable in large corporate intranet environments, because it provides interoperability using a flexible, open, standards-based format. In particular, it will allow a common way of delivering data from legacy databases to Web clients. We expect that web based applications that use data from servers will be built more quickly and will be easier to maintain if they use XML, because there will a standard method of description of transferred data.

Typically, XML will be used in a multi-tier architecture as was shown in the diagram. Agents run on a middle tier to access multiple existing database management systems (DBMSs). The agents then map the legacy data into a standard XML form. The XML data can be transmitted directly to applications that can use XML or the XML can be translated to other forms by reusable translator modules. At present, not so many applications directly accept XML, but the number of applications that use XML as the underlying representation of internet-based data interchange is rapidly increasing. Initially, Office 2000 programs used HTML as a common exchange format (which was not a good data markup notation). It has been announced that Office 2000 tools will be able to use XML as an exchange format.

Potential Use of Wireless Technologies at KDOT

Most internet technologies have been designed for desktop systems which are connected to legacy systems through local area networks. Because some of KDOT's clients and employees may need access to legacy data while away from their desk or out in the field working on a project, we investigated the current wireless technologies which would permit mobile access to KDOT data. The goal of this investigation was to understand the wide variety of wireless technologies available and then recommend one or two specific wireless technologies, which would demonstrate the capabilities to access the KSU/KDOT webpage. The project report at http://www.cis.ksu.edu/~deep/690 explains the complete investigation. In this section we will give a brief overview of the results of the investigation.

Handheld wireless devices (PDAs, laptops, etc.) have a constrained set of resources when compared to desktop workstations on a LAN. They tend to have less memory, less powerful CPUs, different input devices, less communication bandwidth, and smaller displays; but such devices make it easy for a KDOT employee (while away from his/her desk), or a KDOT contractor (at a construction site) secure access to the KDOT legacy data which is represented in XML. This is achieved through the use of WAP (Wireless Access Protocol) and WML (Wireless Markup Language). The two protocols have become defacto standards for wireless transmission and markup of documents.

WAP is a standard which defines two essential elements: an end-to-end application protocol and an application environment based on a browser. The application protocol is a communication protocol stack that is embedded in each WAP-enabled wireless device (also known as the user agent). The server side implements the other end of the protocol, which is capable of communicating with any WAP client. The server side is known as a WAP gateway and serves to route requests from the client to an HTTP (or web) server. The WAP gateway can be located either in a telecom network or a computer network (an ISP). Thus, the client communicates with the WAP gateway in the wireless network. The WAP gateway translates WAP requests to WWW request, so the WAP client is able to submit requests to the Web server. Also, the WAP gateway translates web responses into WAP responses or a format understood by the WAP client.

The Wireless Markup Language (WML) is an XML-based markup language that was designed to describe how WAP content is presented on a wireless terminal. WML differs from HTML in the following ways:

- WML was specifically designed for wireless terminals with a target screen that is only a few lines long and about an inch wide.
- WML is case sensitive and all tags and attributes should be in lowercase.
- Unlike HTML, WML is unforgiving of incorrectly nested tags.
- WML doesn't assume that a "QWERTY" keyboard or a mouse is available for user input.

Based on these differences, WML provides a smaller, telephone-aware set of tags that make it more appropriate than HTML for handheld wireless terminals. But similar to HTML, with WML you can give the user input options and specify how the user agent should respond when, for example, the user presses a key.

To test the difficulty of converting the XML representation of KDOT data to WML, we considered five different approaches. First, the brute force method involved writing actual WML code after reviewing the XML data to render it in a WML browser. This was too inefficient. Second, use a small Java program which takes datafile.xml and converts it to converted.wml. This was relatively easy. Third, we build an XSL transformation. XSLT is the language that specifies the conversion of a document from one format to another. Fourth, we looked at using a web publishing framework like Cocoon, which is a web publishing framework available from http://xml.apache.org. But we felt Cocoon was cumbersome. Finally, we looked at transcoding proxy technology, a develop once and deliver everywhere technology. While this seemed promising, it is not used widely as yet. As a result, option 4, XSLT, seems to be the best approach to convert XML data to WML and permit mobile access to KDOT data.

In order to access WAP services, one needs a WAP-enabled phone like the Nokia 7110, Siemens S25, Ericsson R380, or Alcatel OneTouch. But since we didn't have any of these phones, we used a WAP emulator from Nokia's WAP toolkit and wapjag.com's online browser to demonstrate the delivery of the XML KDOT data example of the next section to wireless devices.

An Example System

A demonstration system is available from the KSU/KDOT web page, http://www.cis.ksu.edu/KDOT/index.html. The system allows browser access to fictitious information stored in an Oracle database in the Computing and Information Sciences Department. The demonstration system does support access by Office 2000 or by wireless devices. Also, certain features work with Internet Explorer, but not with Netscape. Visitors can access general information. Employees and contractors can access selected database information. Only queries are allowed. Update of information could be added.

To use the demonstration, the following items are required. First, the servlet process must be started by staff at KSU. (The default policy in the CIS department is that non-system processes are terminated after a period of inactivity.) Then, when running the client applet from the web page link:

the employee identifier name is "KDOTemp"
the contractor identifier name is "KDOTcont"
the password is "4kansas"

There is no issue of secrecy, since the data is not real and no updates are supported. Queries report data one page of information at a time. The user may also request a file of the query data in RTF form, but that is slower, and it does not yet work in Netscape.

The software structure of the Java servlet is shown by the skeletal class diagram in Figure 7 below. A more complete class diagram is included in the project document at the web site at www.cis.ksu.edu/KDOT/Prapoo/msreport/ <section>.

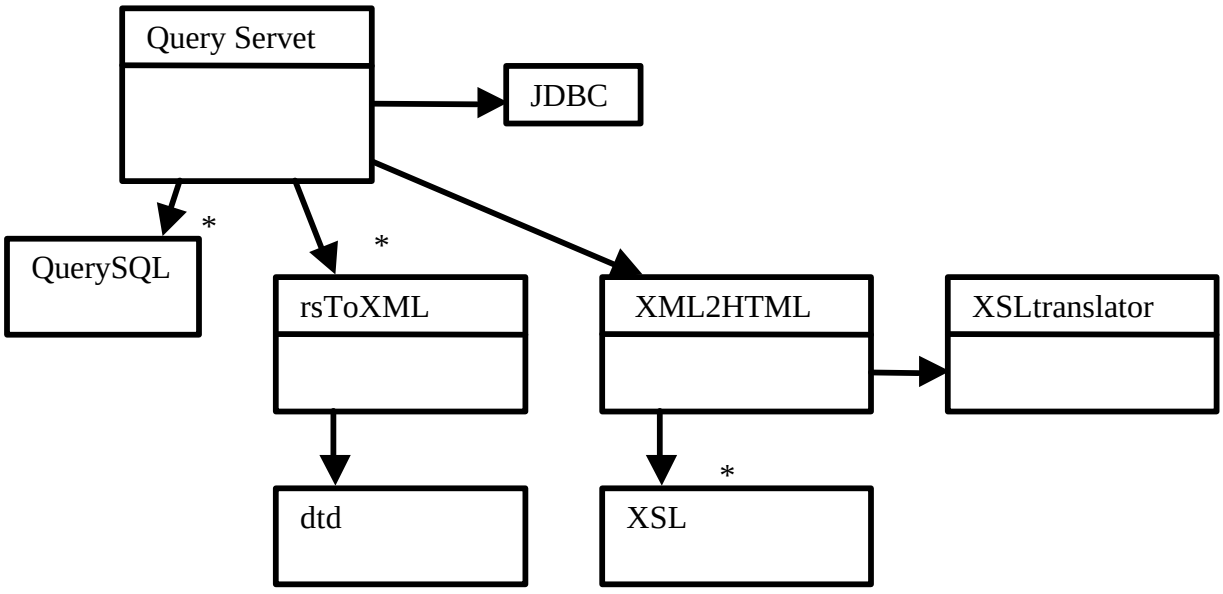


Figure 7. Model of Query Servlet

The actual software files can be accessed at the www.cis.ksu.edu/KDOT/ site in subdirectories "servlet," "projxsl," and "projdtd."

The model structure is also defined in the import statements of the queryServlet.java code:

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.xml.*;
import com.lotus.xsl.*;
import java.util.*;
```

The model structure shows that the servlet uses JDBC to access the database. The actual queries are stored as SQL text within the "query.java" class. Each query returns a result set with prefix "rs." In older client/server architectures, the result set would be returned to the client. However, since we want to provide data that multiple clients can easily use, we translate the result set to XML form. Each record set is translated by a corresponding "rs2xml" class. The definition for each result structure is defined by a document type definition ("dtd") file. For use by html clients, the XML structure is translated to HTML by applying an "xsl" style sheet to the XML document. The style sheets are quite lengthy and are not reproduced here. The style sheets are applied using an existing translator "lotusXSL" available from IBM [ref]. It in turn uses the XML parser "xml4j" also available from IBM.

While there are several parts to the servlet structure, the basic structure repeats for most queries. It is easily extended to other applications.

The application database for the demonstration project includes five tables.

- The USERLOGIN table stores information pertaining to login, password and type of user (Employee, Contractor or Visitor)
- The PROJECT_DETAILS table contains project #, description, start date, finish date, contractor involved, liaison officer and jurisdiction under which the project is being done.
- The PROJECT_BUDGET table contains budget-related data, such as budget amount, amount of change, reason for change, etc.
- The PROJECT_ACTIVITY table stores project stage, activity #, estimated cost related information.
- The CONTRACT_DETAILS table contains details such as contract type, description, workers employed, man hours, percent completed, etc.

These are organized as:

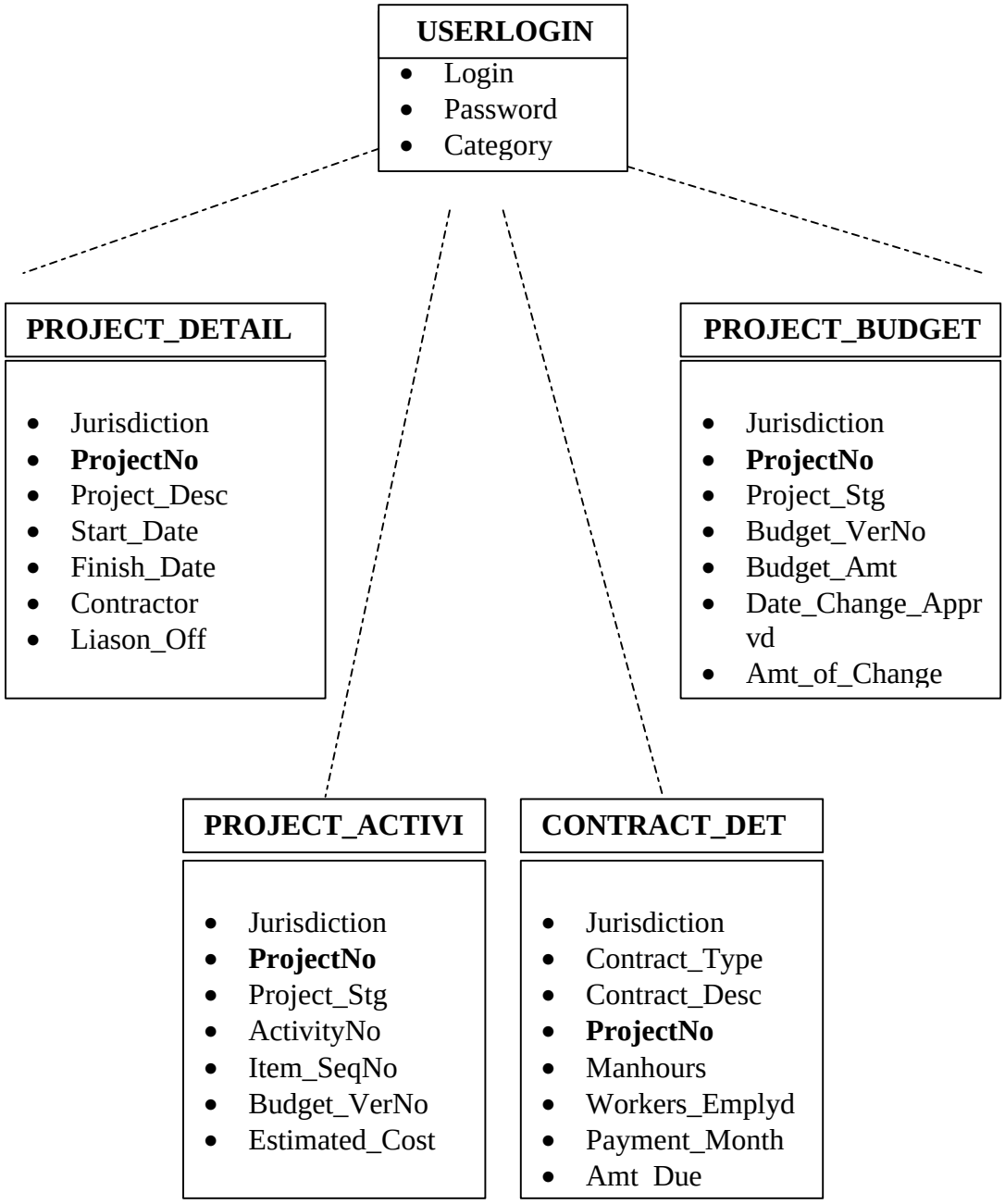


Figure 8. Example Database Structure

Some sample screen snapshots of the browsers client follow:

Figure 9a. Welcome Screen Snapshot

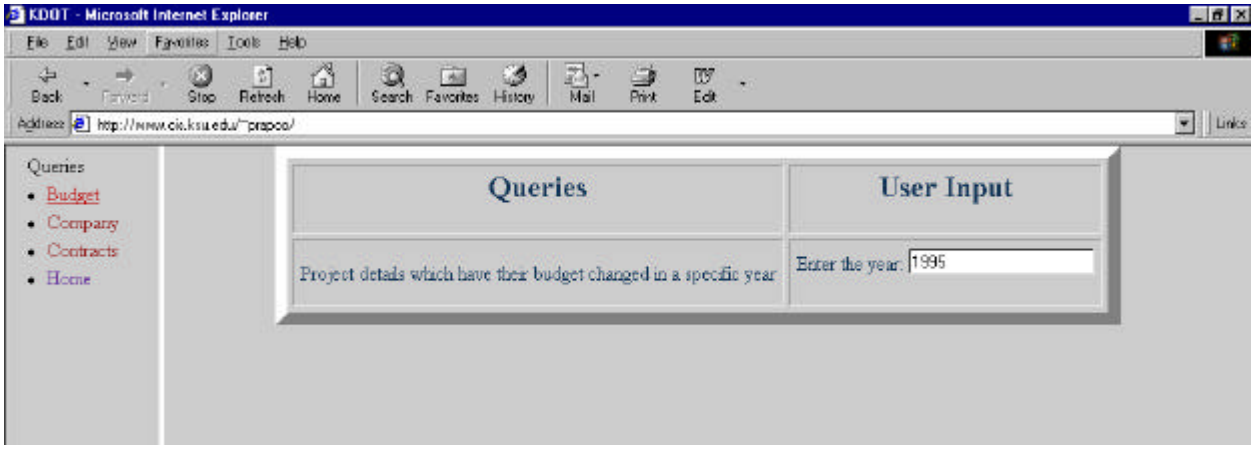


Figure 9b. Query Screen Snapshot

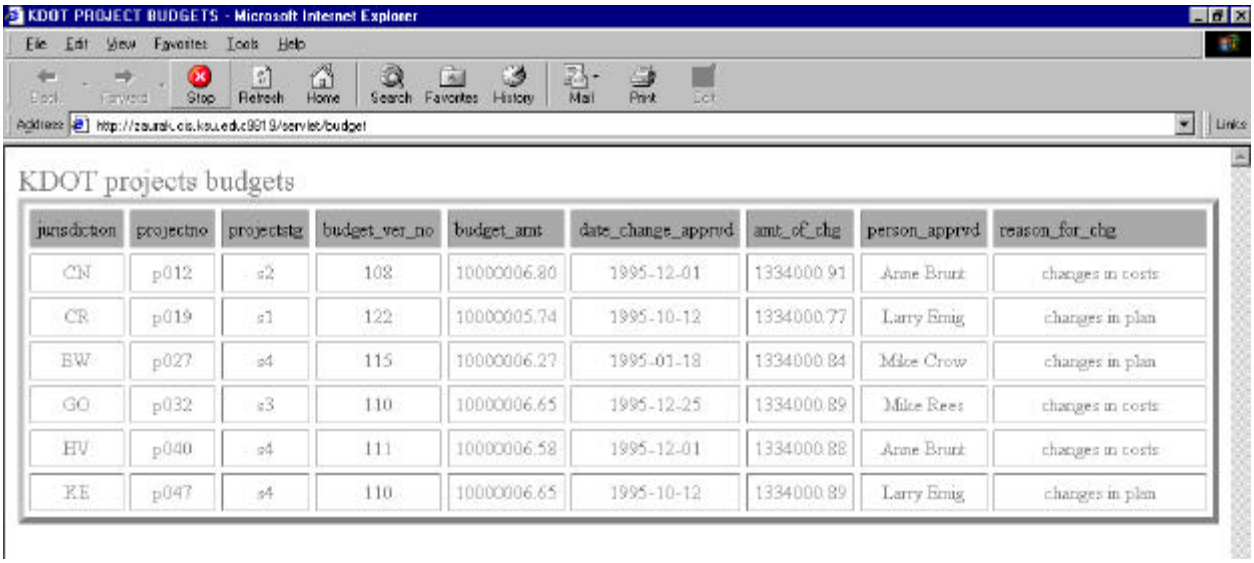


Figure 9c. Query Result Screen Snapshot

What has not been included in the example system is translation of XML to other formats such as for Office 2000. We expect translators for such applications will be forthcoming.

For example, DataChannel [1] provides an XML framework that includes extension for Microsoft Office 2000.

Recommendation 3:

Develop Scripts and Controls for Office 2000 to Automate and Standardize Routine Operations

In the past decade, personal computer tools for office workers have become well established. Now, office workers must be familiar with word processors, spreadsheets, visual presentation tools, and others. However, such tools have not been well integrated, have not been network connected, and techniques for scripting have not been widely recognized and used (except perhaps as a new mechanism for a computer virus). New generations of office tools hold the potential that workers can cooperate on development of documents, or at least can actively share such documents, documents can be easily published on the web, tools can be specialized to support and/or enforce common office procedures, and specialized components can be easily reused. The reality of the state of development of office productivity tools is that many of these potential benefits cannot be achieved without the development of supporting structures and scripts.

Office 2000 consists of a suite of updated versions of the common Microsoft business productivity tools including Word, Excel, Access (database), Publisher (brochures and booklets), Outlook (for email), and Frontpage (for web page management). The most publicized new feature for these tools is that all the programs can export and import documents in a common web format, so that users connected to a network can share documents in a common format. Yet, as was mentioned in the previous chapter, the handling of such shared documents is not yet fully evolved. We expect further development in the use of XML encoding of documents, but that is not the focus of this chapter.

The features of Office 2000 that is of note here (which carry over from Office 97) is that the Office tools support scripting (programming using Visual Basic Script language) and the placement of controls, which are compiled objects (developed either in Visual Basic or other languages). Such scripts and controls can be used in several different ways. Commonly, they allow connections between different Office 2000 tools so that data in one form can be dynamically placed into another document. More important, the data aware controls allow connections to other non-Office data sources. In addition, scripts can implement enterprise rules for processing and transmission of information items. Use of both scripts and controls requires prior development. Such development is not yet appropriate for most end users. We present below two small examples that illustrate how controls and scripts can be used for connectivity and for enforcement of work-flow constraints.

If Office tools are used throughout KDOT, then connectivity between users and constraints of work-flow can be supported either by using third party (not from Microsoft) connectivity and workflow tools or by building controls and scripts. No matter which option is selected, the development will be strengthened by preliminary study of connectivity and work-flow techniques.

The recommendation of this section is that KDOT should pursue exploratory study and

development of the following:

(i) controls for data connectivity,
(ii) models of workflow,
(iii) scripts for enforcement of workflow constraints.

These are illustrated in the examples below.

Importing database information

A common connectivity requirement is to be able to import database information into other applications (into letters, reports, web pages, or spread sheets). In principle, the ADO control (Microsoft Active Data Object) should support connectivity of remote data objects within Office tools [1]. At KSU, we are using an Oracle 8 database server. As of early 2000, the ADO control did not correctly connect to Oracle 8 [2]. As an alternative, we found information for directly using OO4O (Oracle Object For OLE driver from Oracle) from within Visual Basic [3]. Following that guidance, we constructed a mechanism to access the Oracle database developed in Recommendation 2 from within a Word document. This is illustrated in Figures 3.1, 3.2, and 3.3 below. Figure 3.1. shows a macro enabled in the Word tool bar. The macro invokes the Visual Basic form shown in Figure 3.2. Code for the form is included in the file /KDOT/Report/contractcode.txt . The form is initially empty until the user requests the query data. If the data is acceptable to the user, it is then transferred to the Word document. The letter with embedded table of data is shown in Figure 3.3. The data was not typed manually and was not pasted from other Office programs. Rather the data was copied dynamically from the database by the VB script.

This example illustrates that information from KDOT databases can be directly connected into local spreadsheets (for example for budget and planning operations), into other forms (for normal work processing), and into letters and reports.

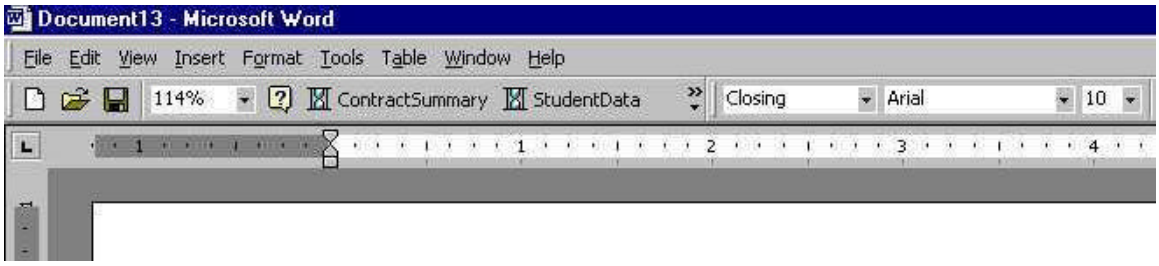


Figure 10. Macros in Toolbar

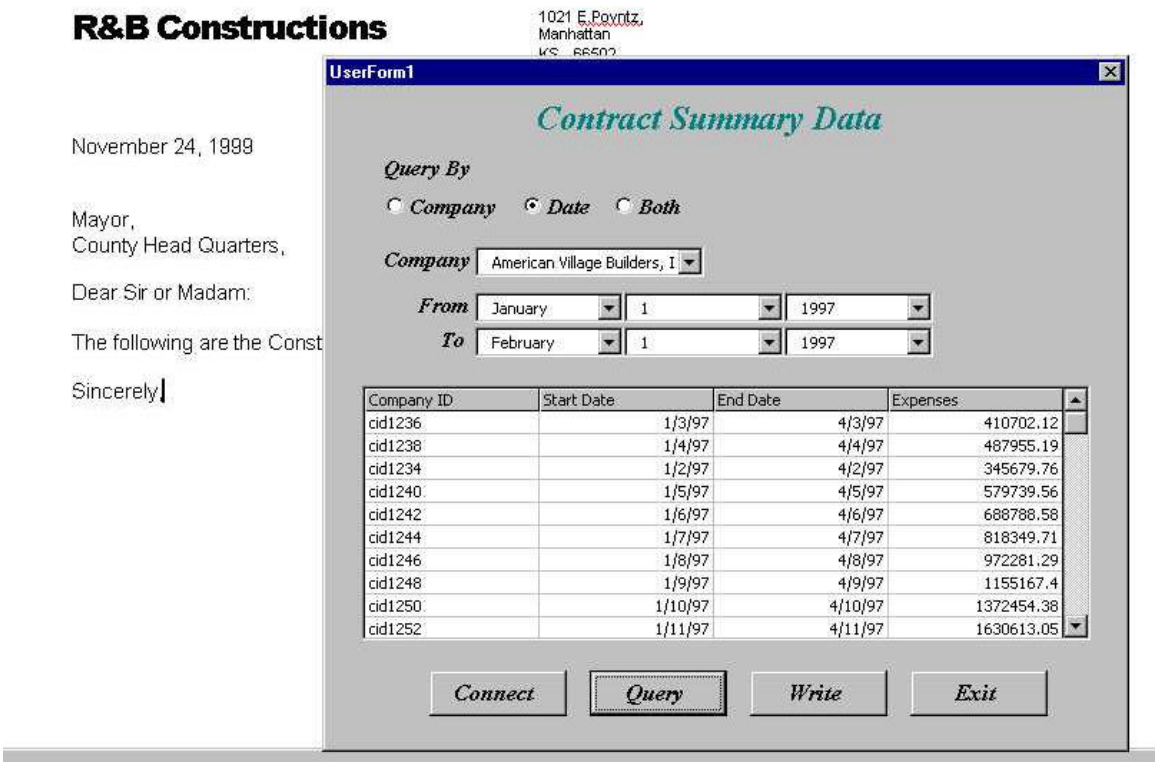


Figure 11. Form with Data from Database

R&B Constructions

1021 E.Povntz,
Manhattan
KS - 66502

November 24, 1999

Mayor,
County Head Quarters,

Dear Sir or Madam:

The following are the Construction details for the month of January (January 1st to February 1st).

Company ID	Start Date	End Date	Expenses
cid1236	1/3/97	4/3/97	410702.12
cid1238	1/4/97	4/4/97	487955.19
cid1234	1/2/97	4/2/97	345679.76
cid1240	1/5/97	4/5/97	579739.56
cid1242	1/6/97	4/6/97	688788.58
cid1244	1/7/97	4/7/97	818349.71
cid1246	1/8/97	4/8/97	972281.29
cid1248	1/9/97	4/9/97	1155167.4
cid1250	1/10/97	4/10/97	1372454.38
cid1252	1/11/97	4/11/97	1630613.05
cid1254	1/12/97	4/12/97	1937331.37
cid1258	1/14/97	4/14/97	2734701.33
cid1260	1/15/97	4/15/97	3249098.65
cid1262	1/16/97	4/16/97	3860254.11

Figure 12 Part of Letter with Embedded Data.

Enforcement of work flow rules

An emerging area of office automation is the management of enterprise rules for work processing. Figure 3.4 below shows constraints for workflow processing of transactions for a hypothetical parts warehouse. The example is taken from the text by Adamson [4]. To illustrate enforcement of such constraints, a demonstration program was constructed using the rapid application feature of Visual Basic and an Access database. The program and database are included in the CIS KDOT web space as www.cis.ksu.edu/Yasien . Full documentation of the database and program user interface are also include at the web site. The program opens a beginning form with connections to subforms for:

(i) administrative inspection of all data records,
(ii) input of parts receiving information,
(iii) input of parts inspection information,
(iv) input of shipping information.

The natural result is that no bad actions or data can be entered. Parts must be received before they are inspected and must be available in good condition before they are shipped. The program is meant to be a "throw away" prototype in that work flow rules are "hard coded" rather than being defined by tables of rules.

The point of the sample program is that once workflow rules are modeled, they can easily be enforced by scripts within Office tools.

In order to run the program, the Access database must be copied from the "Yasien/MSEproject/VB" folder and installed in the PC directory

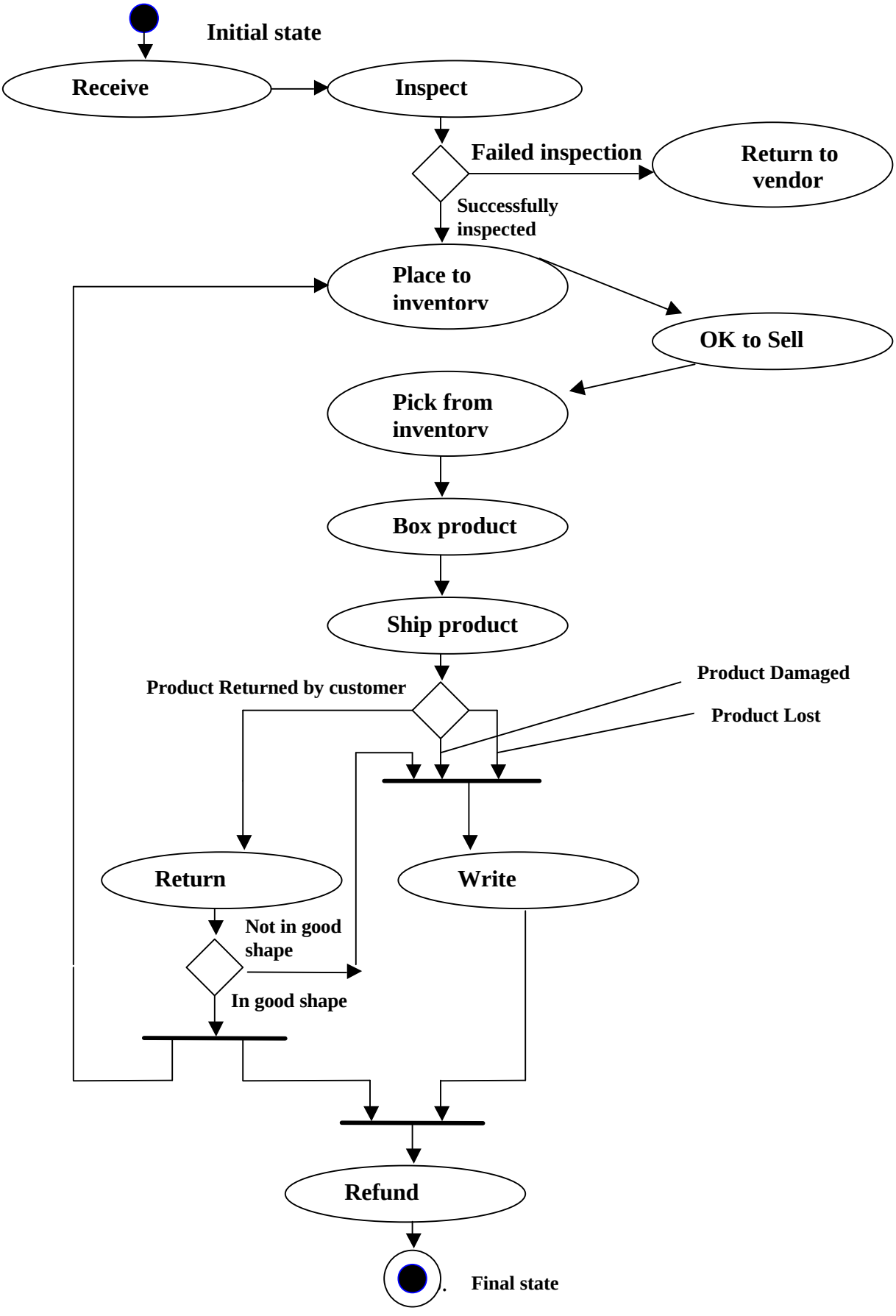
"C:\MSEproject\vb\db4.mdb"

The database is actually in the form of a data warehouse structure, but that is not material to this example. The Visual Basic project has not been compiled. It must be installed on the PC machine.

The user names and passwords are:

"Ahmed" and "ksu" for initial data operations
"Hsu" and "ksu" for receiving operations
"Maria" and "ksu" for inspection operations
"Bill" and "ksu" for shipping operations

Figure 13 A WorkFlow Model for a Parts Warehouse



Recommendation 4:

Develop a Data Warehouse for Support of Research about KDOT Data

Operational databases (such as KDOT's CMS and CPMS) are suited for handling the subject transactions, but they are not suited for analysis of data for decision-making. In contrast, a data warehouse is an integrated collection of data that supports analysis and data mining functions. For example, for KDOT, some kinds of analysis questions might be:

- What project factors (such as work type, jurisdiction, schedule, etc.) contribute to budget change over a certain amount?
- How many projects have their budget changed in a specific year?
- For a specific project, how much is the budget change and why?
- How many contracts were given to contractors from each state?
- For a specific contractor, how much has been paid in a specific year?

To develop a data warehouse, data from separate databases might be processed in several ways. Data may be: cleansed to purge inconsistent or omitted values, transformed to consistent structure and units, integrated with data from related databases, loaded into a common database management system. Transaction data may be aggregated to granularity of information within different units of measure (such as time, cost, spatial position).

In order to integrate data from different databases and data table, the relationships of the entities must be determined. In examining KDOT databases, we found that many likely relationships are not documented within the database structure. The following relations may hold:

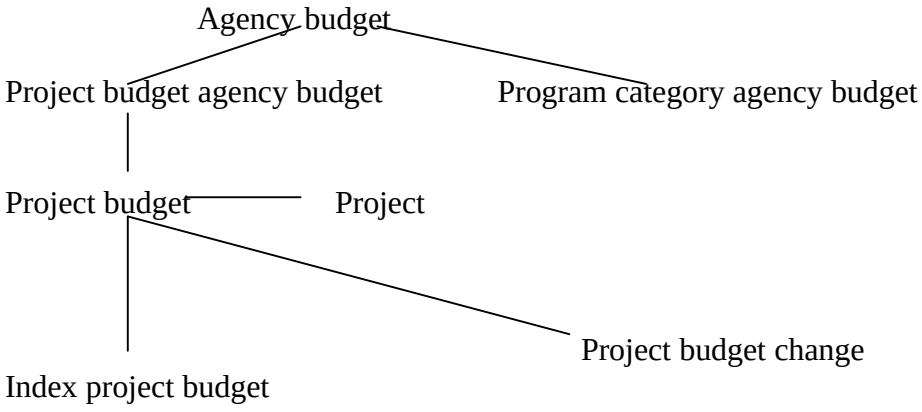


Figure 14. Budget Relations

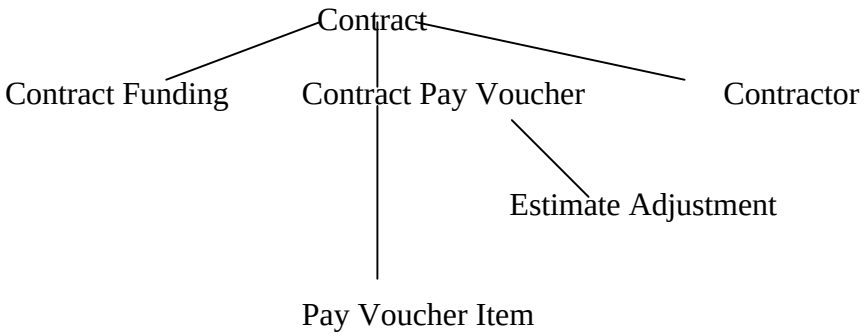


Figure 15. Contract Relations

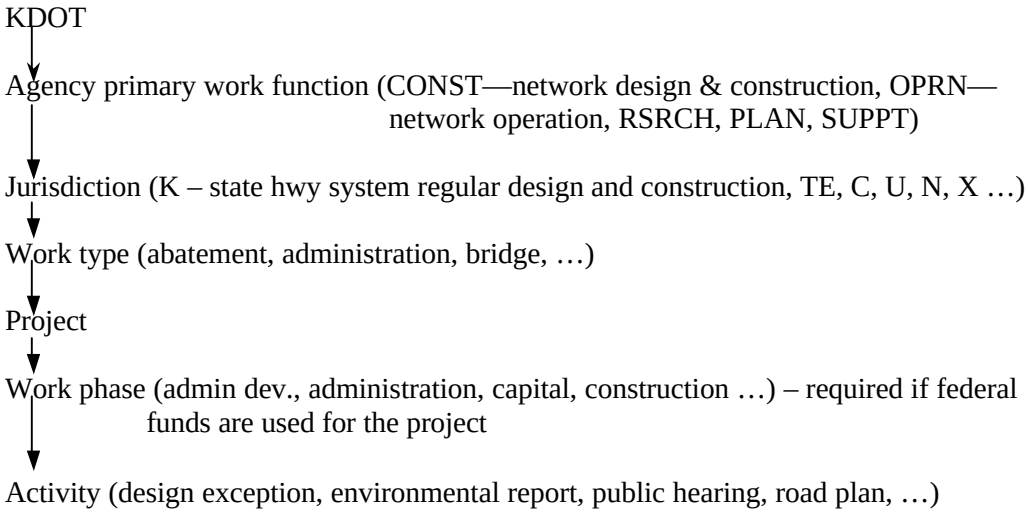


Figure 16. Responsibility Relations

In addition, at least three views or "slices" of information are evident for KDOT:

WBS: work breakdown structure ... "what" information
(pre-construction: road plans, bridge plans, right of way plans, environmental plans, utility plans; construction)
OBS: organizational breakdown structure.... "who" information.
CBS: cost breakdown structure.... "how" information
(equipment, contracts, materials, labor).

Project life cycles may include the following items:

i. Create and initialize the project
ii. Define, plan and organize the work
    Identify the project
    Define the work
    Identify the cost estimate
    Create the schedule
    Fund the project
    Establish accomplishment goals
    Define and allocate resources
    Establish and assign responsibilities
iii. Direct the work
    Record progress
    Record accomplishment
    Billing and expenditures
    Project changes
iv. Control the work
    Make inquiries and prepare reports
    Take corrective actions as necessary
v. Close the project
vi. Evaluate the effort

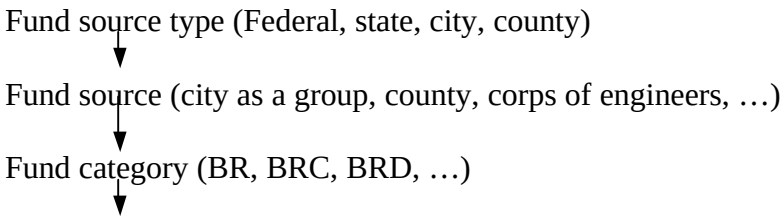


Figure 17. Fund Management Relations

Data Warehouse Star Schema

Data warehouses typically adopt a *dimensional* approach to information processing. In the dimensional model, data is divided into two categories: *facts* and *dimensions.* Facts are numeric measures that are the objects of analysis and dimensions are attributes about facts. This method of representing data is known as star schema. The facts are represented as a table in the center of the schema with multiple "joins" connecting it to the dimension table.

For KDOT, we suggest the following structures:

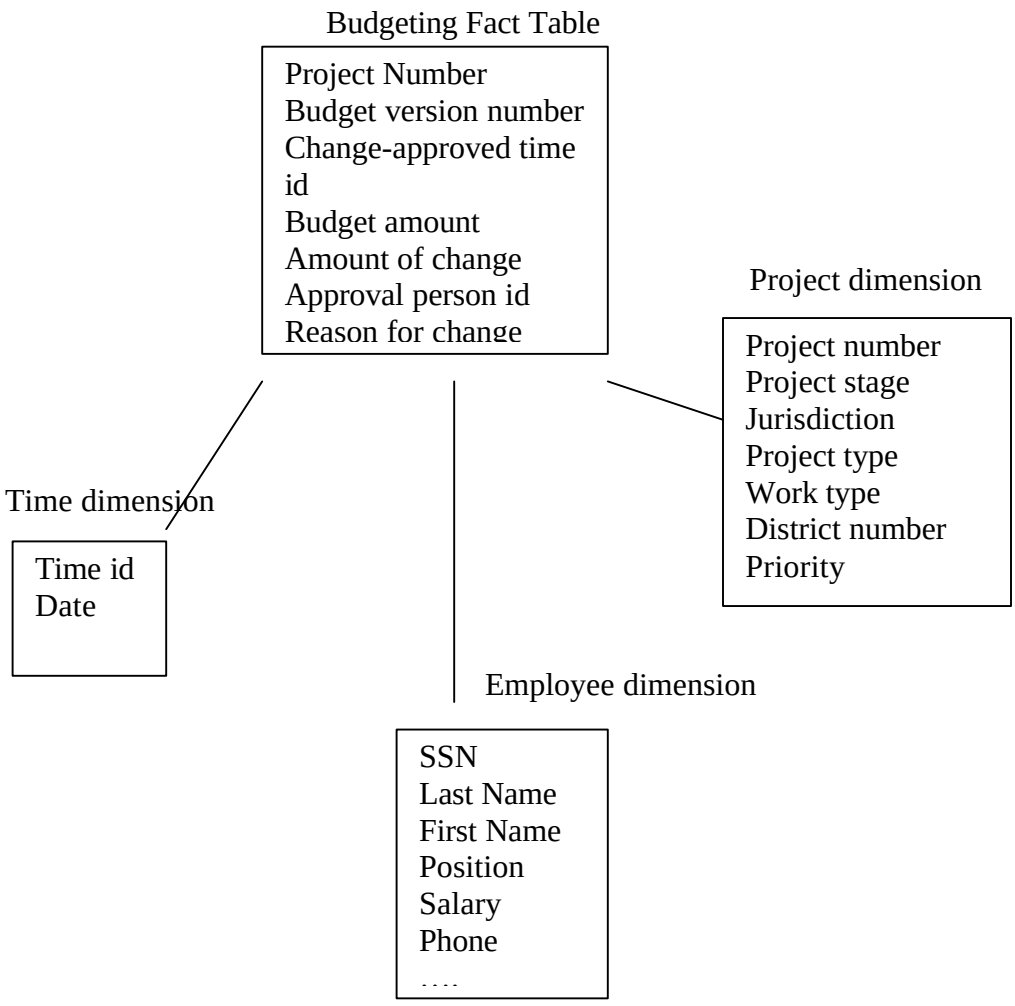


Figure 18. Budgeting Fact Table

Grain: Each budgeting transaction of a project

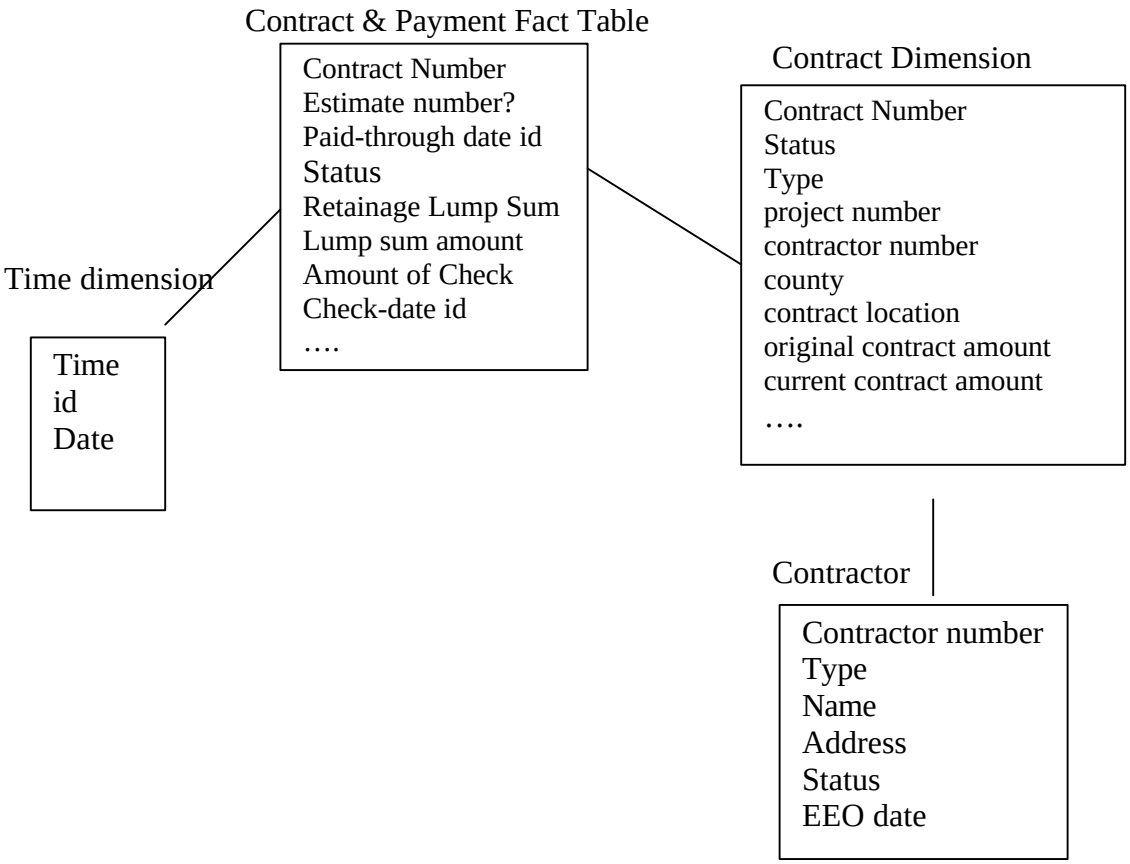


Figure 19. Contract and Payment Fact Table
Grain: Each transaction of payment

References

[1] UML1.3, http://www.rational.com/uml/resources/documentation/index.jtmpl

[2] Booch, G., J. Rumbaugh, I. Jacobson,
The Unified Modeling Language User Guide, Addison-Wesley, 1999.

[3] Penker, Magnus, Hans-Erik Eriksson,
Business Modeling With UML: Business Patterns at Work, 2000.

[4] Rational Rose, http://www.rational.com

[5] TogetherJ, http://www.togethersoft.com/

[6] Argo/UML, http://argouml.tigris.org/

[7] Bosak, J., T. Bray, "XML and the Second-Generation Web,"
Scientific American, May 1999,
http://www.sciam.com/1999/0599issue/0599bosak.html

[8] Chang, D., D. Harkey, Client/Server Data Access with Java and XML,
Wiley, 1998.

[9] Walsh, N., "A Technical Introduction to XML," Oct. 1998,
http://www.xml.com/pub/98/10/guide0.html

[10] Holman, G. K., "What is XSLT?" Aug. 2000,
http://www.xml.com/pub/2000/08/holman/index.html

[11] Extensible Markup Language (XML), Reference Page, 2000, WC3,
http://www.w3.org/XML/

[12] DataChannel, http://www.microsoft.com/Office/evaluation/solutions/datachan.htm

[13] "A Practical Guide to ADO and OLE DB," June 1999,
http://www.oledb.com/ole-db/guide.html

[14] http://www.vboracle.com/develop.htm, October 1999.

[15] VB Oracle OO4O Overview, October 1999,
http://www.vboracle.com/OO4O_intro.htm
This site has been superceded by
http://www.users.totalise.co.uk/~vboracle/
with password "White."

[16] Adamson, C., Data Warehouse Design Solutions, Wiley 1998.

[17] Elmarsri, R., S. Navathe, Fundamentals of Database Systems, Chap 26.
"DataWarehousing and Data Mining," Addison Wesley, 2000.