Object-Oriented Data Modeling and Warehousing to Support Urban Design

Nabeel Koshak and Ulrich Flemming
Carnegie Mellon University
School of Architecture
Pittsburgh, PA 15213, USA

ABSTRACT

All over the world, local authorities are moving towards managing and storing urban data in digital form. But the data storage devices used are heterogeneous and typically include relational database management systems (DBMS), GIS and CAD files. As a result, data are present in different locations on different platforms and under different schemas. This poses a problem for software applications meant to support decision-making in urban design that require input from more than one data source. We demonstrate in our paper how data warehousing—combined with object-oriented data modeling—is able to provide a general solution for this problem. Data warehousing is a technique initially developed for business applications, but is equally useful for urban design: The data warehouse constitutes a communication layer between the urban design applications and data sources. It makes the data available through a unified interface that hides the sources themselves and represents that data in terms of a general-purpose, preferably object-oriented, model. We also describe an implementation prototype that supports different applications. The City of Makkah in Saudi Arabia provides us with real-world data and a context to test our prototype.

1 INTRODUCTION

One of the most important concerns in rapidly growing cities is the need for good, effective urban design. This need is especially acute for the city of Makkah, Saudi Arabia. More than three million people visit the city every year on a pilgrimage (Hajj), and with improvements in mass transportation, this number is increasing every year. Accommodating millions of visitors every year and providing them with adequate utilities, facilities, and services is a big challenge, as is the movement of vehicles and pedestrians during Hajj. Local authorities are also faced with the important task to provide the local population with appropriate housing.

Urban data needed as input to various urban design and planning applications are available from various sources in the city of Makkah. Currently, these data sources are heterogeneous because they are maintained by different authorities at different locations and use different platforms, data formats and schemas. This becomes apparent if we look at the major sources available to urban planners and designers in Makkah:

1. The ArcLand GIS project at the Municipality of Makkah is a GIS system that contains land parcel information and survey maps, using ArcView GIS software.

2. The Hajj Housing Information System at the Ministry of Hajj is a relational

database management system (DBMS) that provides comprehensive information about the housing entities available for pilgrims, using Oracle software.

3. The Saudi Military provides digital aerial maps in the form of AutoCAD files. These files comprise various layers that capture geometric data for land parcels, streets, contour lines, elevation points, buildings and other urban objects.

4. The Makkah Accommodation GIS project at the Hajj Research Center is a GIS system that provides comprehensive information (including location maps, pictures, and textual data) about hotels and furnished apartments available for visitors to the city.

If urban designers and planners want to use software that needs input from these heterogeneous sources, they must transform these data manually into the required format, a time-consuming and error-prone task that may, in fact, preclude using the software at all, thus depriving designers of possibly crucial decision support tools.

The situation in Makkah is not unique. Urban designers and planners are faced with similar problems in various parts of the world. Urban design software typically needs data in the following categories: zoning, administrative and political boundaries, topography, land parcels, buildings, monuments and historic landmarks, open spaces, traffic networks, facilities and amenities, water system, landscaping, accessibility, and utilities, including water, gas, electricity, and telecommunication (Dave & Schmitt 1994, Yin & Williams 1995). We cross-reference in Table 1 these data with the types of urban design applications that need them. The table shows that the same data may be needed by different applications. It also shows that an application may require data in multiple categories and from multiple sources.

We show in the present paper that a *data warehouse*—originally conceived for business applications—can also be used as a layer between urban data sources and urban design applications that hides the heterogeneity of the sources from individual applications and takes, at the same time, the many-to-many relations between heterogeneous data sources and applications into account by making the data available to all applications through a uniform interface. The interface is based on unified data model that integrates the data independently of their source.

In section 2, we briefly introduce the notion of a data warehouse. In subsequent sections, we introduce an object-oriented schema able to serve as data model for a data warehouse to support urban design (section 3); the overall architecture for such a warehouse (section 4); and a prototype implementation of this architecture for the City of Makkah (section 5). We identify open research issues in section 6 and state our conclusions in section 7.

Table 1. **Data needs for different urban design applications**

	APPLICATIONS																								
DATA	3d visualization	emergency service planning	vehicular traffic simulation	facade studies	shortest path analysis	pedestrian movement simulation	facilities distribution analysis	housing stock inventory	VR visualization	circulation analysis	land-use planning/management	urban redevelopment analysis	building views analysis	utilities needs analysis	air pollution analysis	sound pollution analysis	human behavior simulating	emergency response analysis	vegetation analysis	urban light/shadow analysis	choosing best building location	historic preservation	land ownership investigation	demographic analysis
3d building models	X			X					X			X	X							X	X	X		
origin & destination of cars			X							X					X									
street address		X			X					X											X			
origin & destination of pedestrians						X				X							X							
housing capacity		X					X	X						X										X
street segments		X	X							X							X							
land parcels outlines	X	X						X		X	X				X									
traffic network	X	X	X	X	X	X			X	X	X	X			X		X	X		X	X	X		
building condition												X										X		
building footprints	X	X		X				X	X			X	X					X	X	X	X	X	X	X
utilities' maps														X										
residents characteristics						X	X																X	X
vegetation	X						X												X					
sun movement				X																	X			
owner information								X															X	X
Year built								X														X	X	
right of way			X			X																		
attractions						X				X				X		X					X			
zoning		X																						X
sidewalks						X			X								X							
street intersections			X							X						X								
traffic control facilities			X							X							X							
facilities & services							X			X				X			X				X			
topography	X		X	X					X	X									X	X	X			

2 DATA WAREHOUSING

Large organizations or companies may have offices and facilities at many sites, where

each site may collect a large volume of data. This means that different data are present at different locations on different platforms and under different schemas. For instance, product data and customer data may be stored in separate databases at different locations. But corporate decision makers need access to information from all of these sources. Data warehousing provides a solution for this problem (Silberschatz et al. 1999).

A *data warehouse* is a repository (archive) of information gathered from heterogeneous data sources. The data are either stored under a unified schema (model) at a single location or extracted from the sources as they are needed, again based on a unified schema. The extracted data may be queried directly by a user or used as input to an application. A data warehouse thus presents clients (users or applications) with an integrated and uniform data source. Since data sources are constructed independently and likely to have different schemas, a data warehouse must perform schema integration before data are delivered to clients.

Data warehousing may also provide a solution for the problems raised by the heterogeneousness of urban data sources and urban applications as discussed in the preceding section. We envisage an urban data warehouse that is able to extract data from heterogeneous sources, to integrate the data using a unified data model, and to provide input to heterogeneous urban design applications or answer user queries through a uniform interface. The unified data model (schema) integrates specifically geometric and non-geometric data that are currently typically distributed over heterogeneous, non-coordinated sources (like a DBMS and CAD files). Furthermore, an application that has an interface able to receive data from a warehouse supporting a specific data model would be able to receive data from *any* warehouse supporting that model, that is, would be able to support urban design for any city or region maintaining such a data warehouse.

There are good reasons to demand that the unified data model used by a data warehouse to integrate urban data be object-oriented. Research and practical applications over the last decades have amply demonstrated that object-oriented representations are uniquely able to capture the attributes of designed artifacts in a natural and computationally efficient fashion. Furthermore, object-oriented programming has by now established itself as the paradigm of choice for the development of robust, extensible and reusable software (Meyer 1988). An object-oriented representation is therefore not only desirable in its own right, but also facilitates data import to object-oriented applications as they are coming into wide use.

3 UNIFIED OBJECT MODEL

3.1 **Overview**

An object-oriented (OO) representation is a collection of *objects* with attributes, where some attributes may be relations with other objects. The attributes that an object can

have are typically determined by the *class* to which the object belongs. Classes can inherit attributes from super classes and pass these attributes to all objects belonging to the class (these objects are often called *instances* of the class). An *object model* or *schema* is the collection of all classes describing the universe of discourse for an application or group of cooperating applications. All data handled by an application are captured by objects that are instance of classes in the schema. The unified object model underlying our data warehouse is a schema in this sense: its classes define the types of objects that can be instantiated to import data (via object attributes) to urban design clients.

Over the last decade, various notations have been proposed for the documentation and communication of object models. Among these, the notation developed by Rumbaugh (Rumbaugh et al. 1991) has become particularly popular and is now an integral part of the Unified Modeling Language UML (Booch et al. 1999). UML has become the standard language used for object-oriented software development (Naiburg & Maksimchuk 2001). We used Rumbaugh's notation, via UML, to develop and document the object model underlying our warehouse; examples are shown in the figures below.

We developed our schema by first looking at the data we wanted to export from heterogeneous sources. A DBMS, for instance, provides data in the form of flat tables with multiple columns recording various non-geometric attributes of urban objects. A CAD file may represent urban areas or building outlines as 2-dimensional polygons defined by the coordinates of their corners. A GIS source typically consists of flat tables linked with 2D geometries to describe buildings for instance. In short, the data models used by the data sources are structured collections of value attributes.

Our schema reflects these characteristics. It is a collection of classes, each of which collects *all* (geometric and non-geometric) attributes of an urban object. The schema captures no behavior; that is, classes have no method attributes, and objects that instantiate them are strictly passive data repositories. Furthermore, subclass (inheritance) relations are used sparingly; they are needed only to provide some flexibility in describing the various geometries that may be associated with an object like a building. There are no complex relations between classes or objects; that is, no object can be related to more than one other object. This eliminates a notorious translation problem for object-oriented representations with possibly complex many-to-many object relations: the translator receives objects one-by-one, maps each to a representation understandable by the client and sends it on. If such an object has relations to other objects, the translator must determine if the related objects have already been encountered and send out or not; it cannot simply start to translate the related objects immediately because that may result in duplications. The translator therefore has to keep a record of all objects already send out for further reference. In order to be able to do this, it has to maintain a mapping table between incoming and outgoing objects based on unique object identifiers. All of this is avoided if we eliminate all cross-references between objects in our model.

Note that the diagrams below are *class* diagrams that may show one-to-one, one-to-many, or many-to-many relations between *classes*; this only means that an object

instantiating a class may have multiple relations to several objects each of which instantiates the same class, but is a distinct instance of that class. For example, a line has two associated Point objects, each which is an instance of the Point class, but differs from the other point by its coordinate values.

The simplicity of the resulting schema is not only adequate for the data it has to capture, but also greatly facilitates parsing and interpretation at the client end. We illustrate this in the following sections with two example classes, representing respectively a land parcel and a building (see Koshak 2002 for a complete model specification).

3.2 **Examples**

3.2.1 *LandParcel*

A LandParcel object captures both non-geometric and geometric data describing a land parcel in a city (see Figure 1). The LandParcel object is associated with one Deed object and one or many Owner objects.

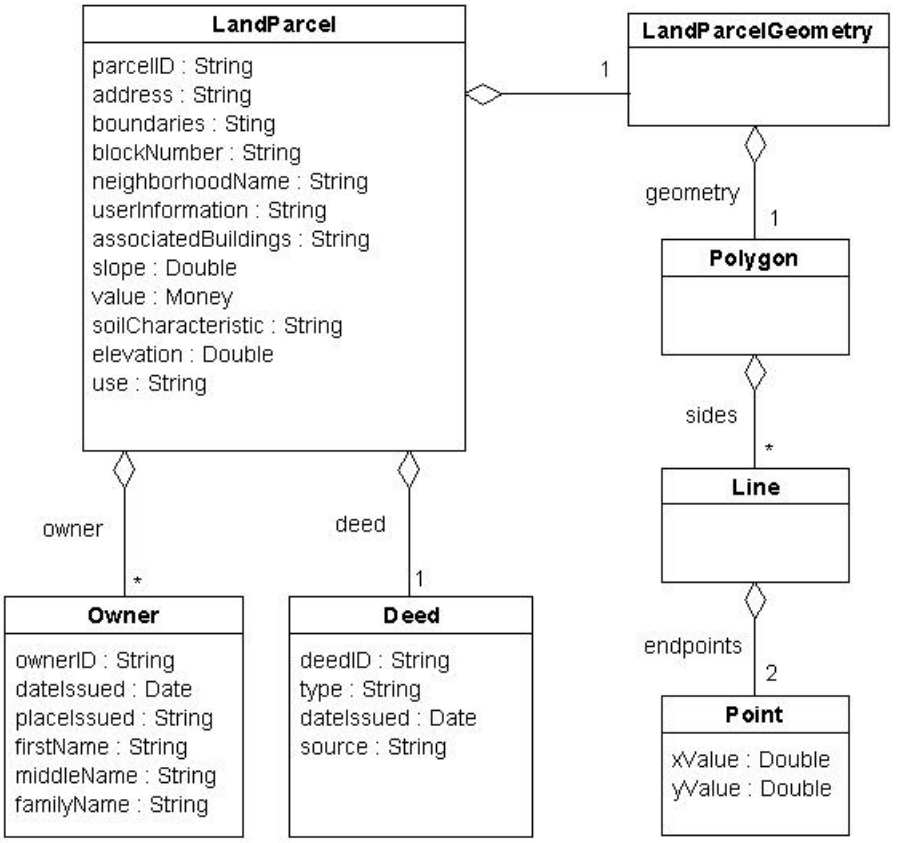


Figure 1. **LandParcel Class**

The boundary of a LandParcel is described by an associated LandParcelGeometry object. We assume—based on the GIS data available in Makkah—that the boundary of a parcel is described by a polygon, which, in turn, is described by multiple lines representing its sides. Each side is defined by the coordinates of its two endpoints. The LandParcelGeometry object therefore has an associated Polygon object. The Polygon, in turn, can have an arbitrary number of associated Line objects, and each Line object is associated with two Point objects.

Since each corner of a polygon is shared by the two adjacent sides, this representation of a polygon is redundant. But we decided to use this representation because we wanted to avoid many-to relations for the reasons stated above.

3.2.2 *Building*

A Building object captures all attributes of a building in an urban environment (see Figure 2). The BuildingGeometry is a collection of 3-dimensional solids of arbitrary size. Our present implementation includes prisms and extruded objects. This allows us to represent, for example, the Holy Mosque in Makkah, the building with the most complex geometry in the city, at a level of detail sufficient for many applications (see Figure 3). Developing a general model for the representation of buildings and other urban objects that would be able to cover the needs of most urban design applications is beyond the scope of our work.

4 A DATA WAREHOUSE TO SUPPORT URBAN DESIGN

4.1 **Overall Architecture**

We mentioned in Section 2 that there are two basic options for implementing a data warehouse:

The *persistent data warehouse* stores all data that may have to be exported to clients in a separate, integrated database. The data are collected frequently (weekly or monthly, for instance) from the heterogeneous sources and stored physically in a separate database, which is a central component of the data warehouse. Data requests by clients are sent directly to and returned by this database. This option is problematic when the amount of data coming from heterogeneous sources is large and consumes a large amount of storage resources. In addition, data stored in the warehouse are not always up-to-date.

The *virtual data warehouse* is more attractive in the present context. In this option, data are collected from heterogeneous sources as needed. When a client requires data, the data warehouse connects remotely to the appropriate data sources and returns the needed data to the client without storing any data within the data warehouse itself.

The overall architecture of a virtual data warehouse to support urban design is shown in Figure 4. A client requests certain data for direct inspection in a web browser or as input for a specific application. The client application sends the data requests to

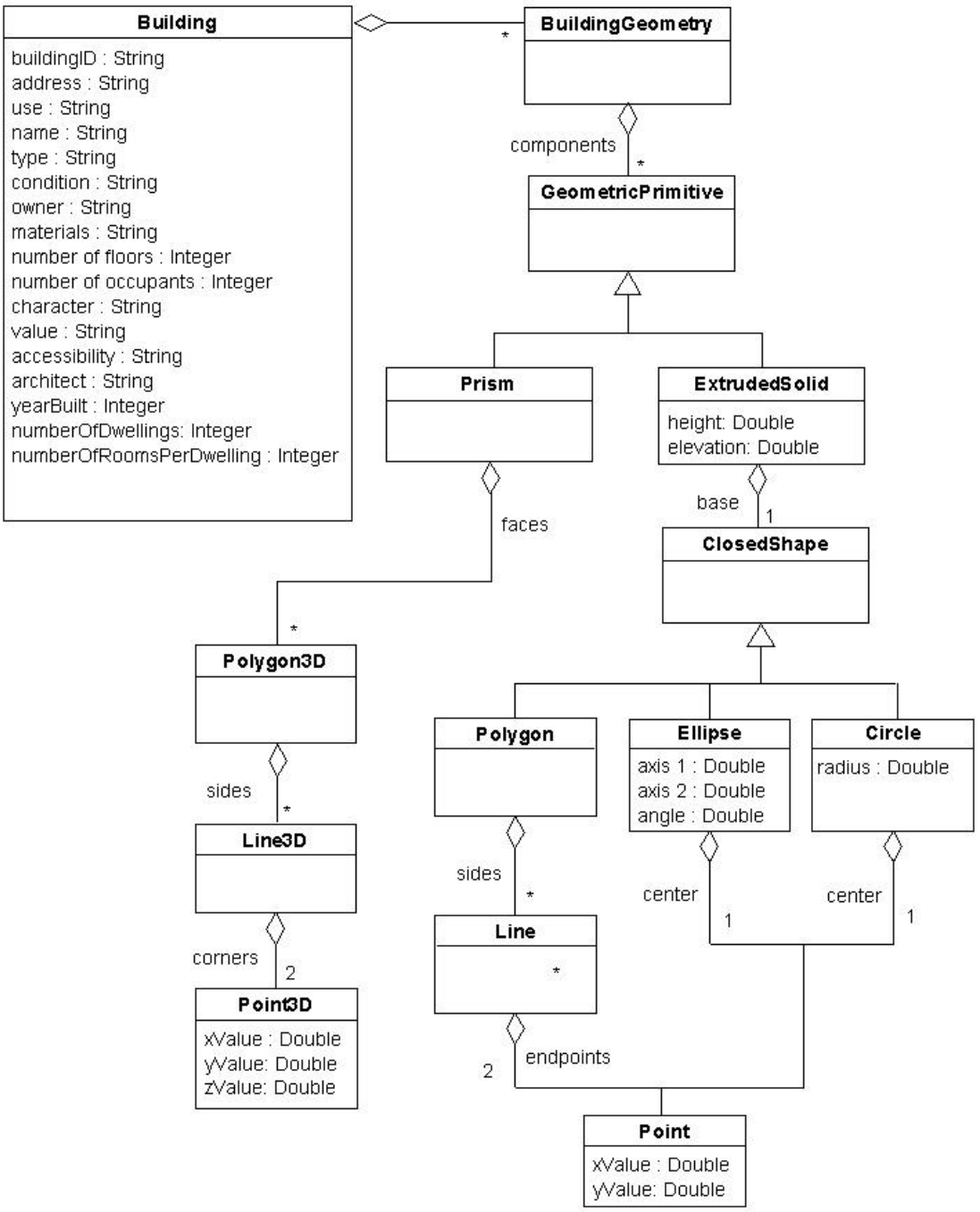


Figure 2. **Building Class**

Data Access Objects in the form of queries. A Data Access Object (DAO) is a software component that is able to import attribute data for a specific class in the underlying schema from various sources and to export the data to clients in a unified form

determined by the schema. A DAO uses Extractors to query the appropriate data.

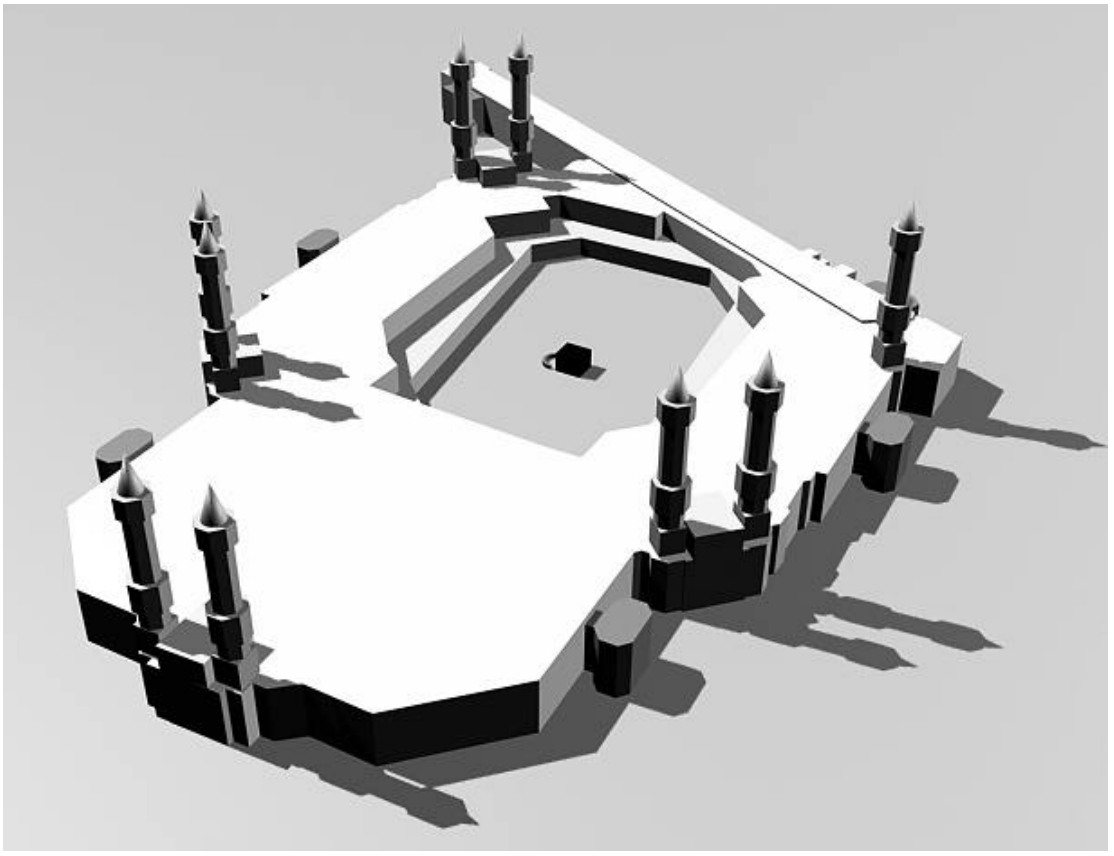


Figure 3. **Geometry of Holy Mosque**

sources (DBMS, CAD, or GIS). Solvers may be needed to handle particular data integrating problems, for example, when different sources return different values for the same attribute.

To preserve data integrity, the data warehouse does not accept any data *from* clients that would modify the state of the data sources. Only local authorities are allowed to modify persistent data or to add new data to the warehouse.

4.2 **Data Access Objects (DAOs)**

A *Data Access Object* (DAO) is able to answer queries on objects whose attributes are recorded by heterogeneous data sources. Our warehouse must contain a DAO for each class in the underlying schema. The DAO provides a query (get) method for each attribute in the associated class. It needs an Extractor for each of the heterogeneous sources to import the data recorded by that source. Accordingly, an extractor is selected and used to obtain the value for that attribute.

The DAO design pattern separates an application from its data sources. This has two advantages: The applications are isolated from changes in the data sources.

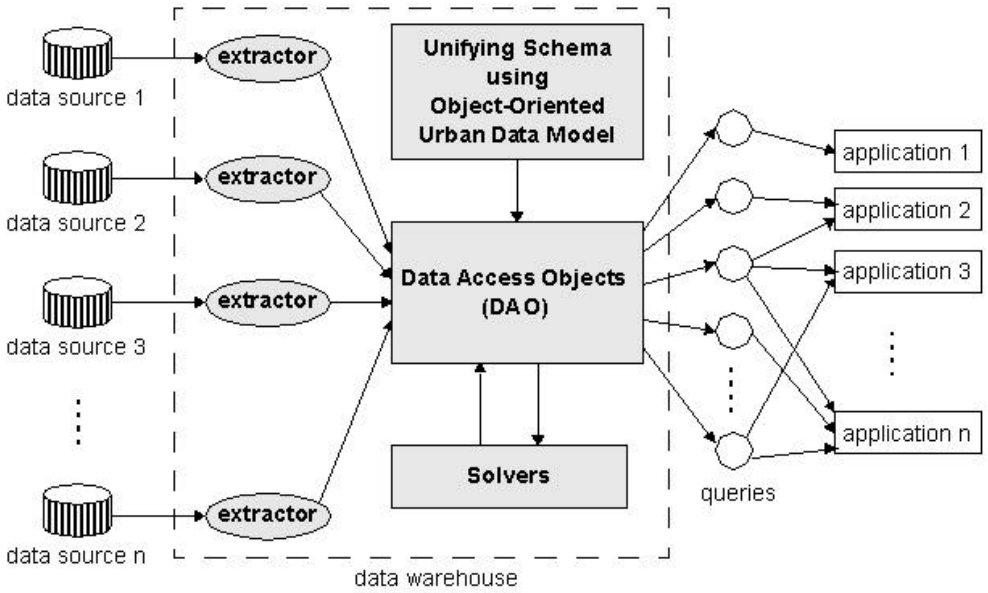


Figure 4. **The overall architecture of an urban data warehouse**

They are, in fact, able to retrieve data from different warehouses, as long as the warehouses support DAOs based on the same schema. Conversely, the data in the sources become available to different applications (Wheeler & Wheeler 2001). For example, the applications do not have to modify their data import procedures when the source for specific data items changes (although the extractors for this information will have to change). Moreover, applications that are able to import data from the warehouse are also able to import, without further modification, data from any other warehouse that supports the same schema and DAOs.

4.3 **Extractors**

An *Extractor* is a software component that receives a data request, connects to the appropriate data source, and returns the requested data. As illustrated in Figure 5, an extractor has to be implemented for each kind of data source, and the data warehouse may have to provide, for example, a DBMS extractor, a CAD extractor, and a GIS extractor.

An extractor receives two arguments from a DAO: a data source location and a query (SQL for instance). It then connects to the specified data source and returns data according to the specified query. For example, a DBMS extractor takes a URL (Uniform Resource Location) and SQL (Structured Query Language) statement. The URL indicates where the (remote or local) database is located. The SQL statement specifies

the selection criteria of the requested data. The extractor uses these arguments to connect to the database and retrieve the data. It then returns the requested data to the DAO.

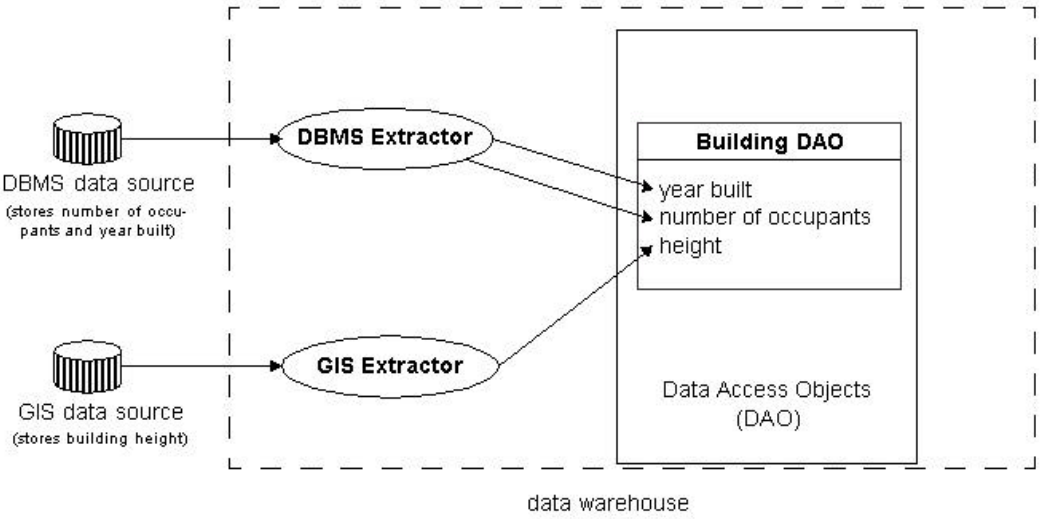


Figure 5. **Interactions between data sources, Extractors, and DAOs**

Our extractors extend the wrapper concept to data extraction. A *wrapper* is a software component that translates data and queries from one model to another (Papakonstantinou et al. 1995). In the same sense, an extractor translates data requests from a DAO into source-specific queries, that is, it plays the same role as a wrapper.

4.4 **Solvers**

A *Solver* is a software component able to perform data cleaning, transformation, and integration tasks (see Figure 6). The motivation to introduce solvers into the general warehouse architecture stems from observations like the following: Suppose that two different local authorities are using two different GIS map projections that represent the same urban area. A map projection maps locations on the globe onto the flat surface of a map. All map projections distort the shapes of the features being displayed to some degree; this also holds for measurements of area, distance, and direction (Kennedy & Kopp 2001). In this case, we need a process that is able to resolve the data conflict in some appropriate way. In our data warehouse, these processes are generally delegated to Solvers.

The data warehouse administrators integrate these solvers with the data warehouse architecture. The selection of solvers is left to the user and can be handled by preference features in a user interface through which the user is able to select the appropriate solver among the available ones whenever a data integration problem arises.

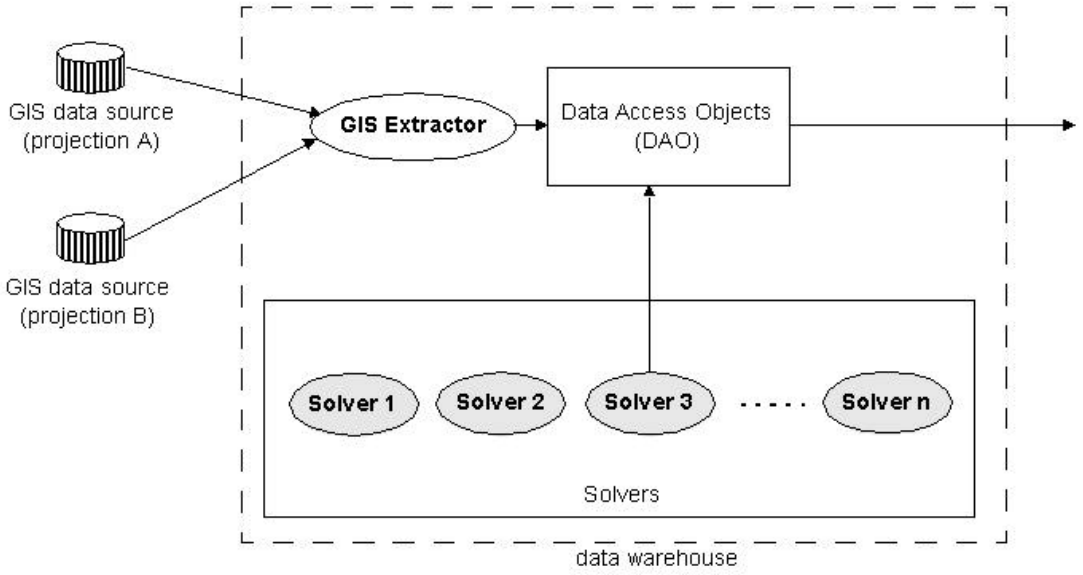


Figure 6. **Solvers in the data warehouse**

5 PROTOTYPE IMPLEMENTATION

5.1 **Languages**

To implement a first prototype of the proposed data warehouse, we used Java and the Extensible Markup Language (XML) as languages to provide, respectively, portable code and portable data. Java is not only portable through the Java Virtual Machine, but also supports object-oriented features that are needed to implement our unified object-oriented data model. Java also supports building web-enabled applications. XML provides an open standard data format that can be used to import data by different urban design applications.

5.2 **Heterogeneous Data Sources**

For obvious reasons, we could not exercise our first prototype directly on the data sources owned by various authorities in the City of Makkah. To be able to experiment with the various types of heterogeneous data sources commonly used in cities such a Makkah, we simulated existing sources by reasonable equivalents and populated the sources with real data:

- A DBMS data source using MS Access. This source simulates the relational database available at the Ministry of Hajj. Sample data were obtained from the Ministry and imported into the Access database.

- AutoCAD represents CAD-based data in our prototype. This source simulates the AutoCAD files available at the Ministry of Defense. Sample data are again obtained from the Ministry of Defense and captured in DXF files.

- A GIS data source is simulated using ArcView. This source is a proxy for the ArcView GIS available at the Municipality of Makkah. As with the previous sources, sample data were obtained from the Municipality of Makkah and imported into ArcView.

5.3 **Data Model and Data Warehouse**

The unified data model is implemented using Java classes. A DAO must be implemented for each of these classes. We use again Java to implement DAO classes. When the system is running, urban (domain) objects and DAO objects are created as instances of the corresponding Java classes. A DAO uses the domain object it serves directly to capture the retrieved data. For example, a BuildingDAO object uses a Building object as defined in the unified model to store the data retrieved from different sources.

Three kinds of data extractors were implemented in Java, one for each of the simulated data sources described in Section 5.2. We implemented a sample Solver to handle a typical data integration problem: the problem that arises when the key attribute for the same object is different at two different data sources, for example, the same building has building id b041 at data source A and id 3873 at data source B. A Solver handling this problem maps the building identification numbers between the two data sources. The Solver is implemented using Java and added to the data warehouse as a software component that is called by a DAO whenever this integration problem occurs.

5.4 **Client Interfaces**

5.4.1 *Web Interface*

A user can use the web interface to retrieve and inspect data of interest. We developed this interface using HTML (Hyper Text Markup Language) pages, which allow the user to specify the needed data (see Figure 7). Data requests are sent to the web server—through HTTP (Hyper Text Transfer Protocol)—where the data warehouse is located. DAOs receive these data requests and communicate with other components of the data warehouse to retrieve the requested data. The DAOs then return the data requested to the user through the web interface. The DAOs present the retrieved data as HTML tables. The user can then use the web browser to view the requested data (see Figure 8). The user can also download the data as XML files. These XML files can be parsed and used to import the retrieved data into an application.

Note that it would have been more elegant (and simpler) to implement DAOs that always return data in the form of XML files—let the client worry about what to do

with them! But this would have forced every client to create a parser and interface to some application. We avoided this by enabling the data warehouse to return data as web pages; this can be considered an extra service offered by the warehouse.

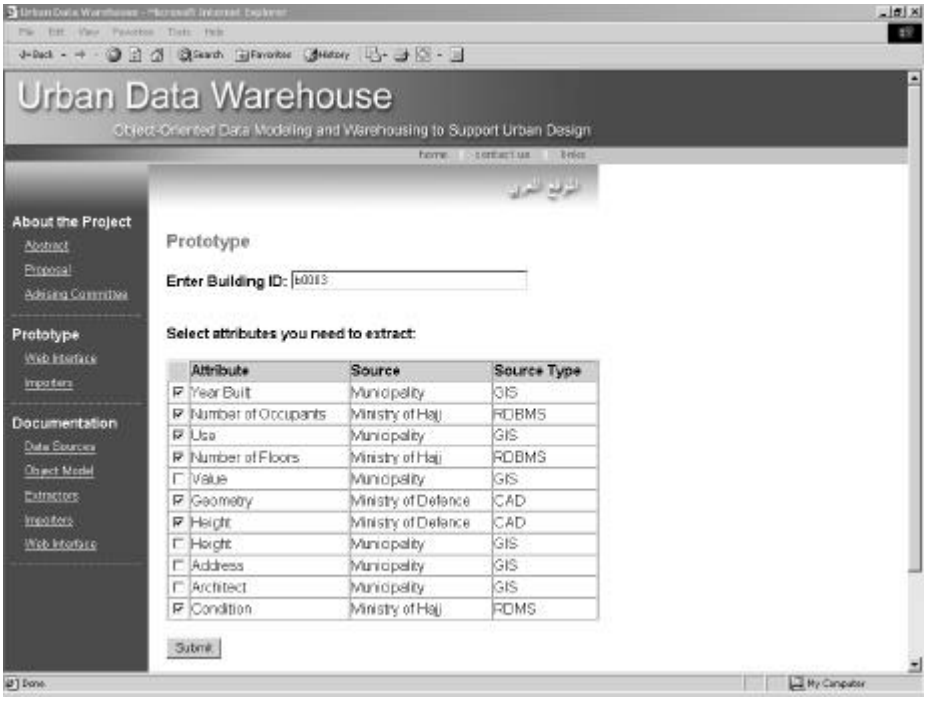
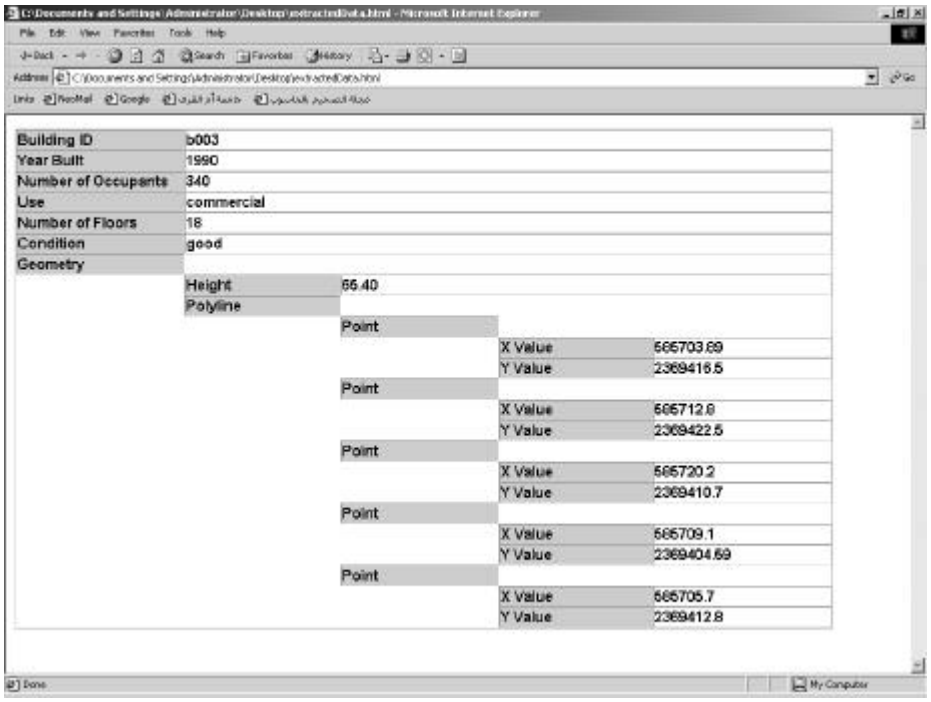


Figure 7. **Web Interface to the data warehouse**

5.4.2 *Application Interfaces*

We simulated an application that can be used to visualize in 3 dimensions existing and proposed buildings, where the existing buildings are modeled based on data received form CAD files and a DBMS. The proposed buildings are modeled locally in the same modeler so that their form and impact on the neighborhood can be assessed.

We use AutoCAD to simulate the visualizer. We use AutoCAD's API, Visual Basic for Applications (VBA), to parse an XML file returned by the warehouse and to import the data into AutoCAD. The application then constructs a local model of the existing buildings and renders them on the screen. Proposed buildings can be modeled and visualized using the same software.



Building ID	b003			
Year Built	1990			
Number of Occupants	340			
Use	commercial			
Number of Floors	18			
Condition	good			
Geometry				
	Height	66.40		
	Polyline			
		Point		
			X Value	566703.69
			Y Value	2369416.5
		Point		
			X Value	566712.8
			Y Value	2369422.5
		Point		
			X Value	566720.2
			Y Value	2369410.7
		Point		
			X Value	566709.1
			Y Value	2369404.69
		Point		
			X Value	566705.7
			Y Value	2369412.8

Figure 8. **Extracted data represented as HTML tables**

6 FUTURE WORK

The most important issue that has to be addressed in the context of the work presented here is how to represent generally the geometry of various urban objects. The representations of object geometries in our unified object model were developed ad-hoc based on the data available/needed for the City of Makkah. But the issue requires a more detailed study aiming at finding a consistent representation for object geometries that is flexible and general enough to cover all important cases while maintaining simple enough for easy parsing and interpretation.

Another open issue at the time of writing is how to construct and use solvers in general. Currently, solvers are built and added to the warehouse by its administrators. But this need not be the case. We envision a situation in which the data warehouse has become the backbone of a connected community of data providers and data users that cooperate and benefit from each other's work. At the most basic level, an authority maintaining one data source serving external clients may in turn become a client when it needs data maintained by other authorities. But we want to go a step further. Data providers as well as data users gain knowledge by using the data warehouse to support urban design decisions. We believe that Solvers can be used to make these insights available to the entire community.

In a more effective collaborative environment, solvers can be developed by urban designers and shared—via the Internet—through the data warehouse as distributed

reusable software components. The Internet has become an important medium for sharing information. Many Internet applications deliver data, but the web remains underused for remote processing of computational objects. Bhargava et al. have described how consumers can use, over the Internet, technologies that are located at and execute on providers' machines (Bhargava et al. 1995a/b). One application of this approach would be the use of the Internet to share Solvers for data integration problems. Solvers can be shared based on parameterized models set by the data warehouse administrator.

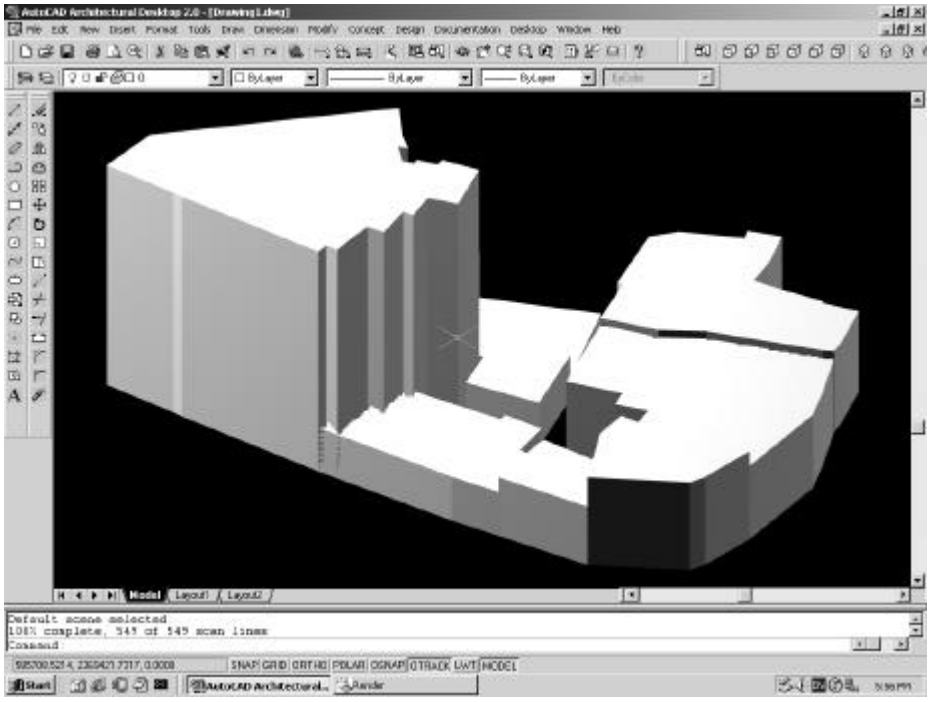


Figure 9. **3D Visualization of building data**

Finally, we want to point to possibilities that emerge once the problem of heterogeneous data sources for urban design applications has been solved. The data available through a unified interface can now be used to support more advanced decision making tools that *generate data from the available data.* We mention just two promising emerging techniques: On-line Analytical Processing (OLAP) systems and Data Mining systems. OLAP refers to an advanced data analysis environment (Rob & Carlos 1997). OLAP tools access the data warehouse to provide advanced multidimensional data analysis, for instance, in the present context, OLAP can be used to monitor and analyze urban growth in cities. Data mining is a computer-based method to analyze data with the intention of finding previously unknown data characteristics, relationships, dependences, and trends (Rob & Carlos 1997). For instance, data mining applied to a target data set extracted from the warehouse can be used to investigate the relationship between urban growth and traffic congestions.

7 CONCLUSIONS

We demonstrated that data warehousing can be used to solve a central problem in computer-supported urban design and planning: the heterogeneity of data sources needed by clients such as urban designers and the applications they use. The warehouse provides an intermittent layer between the sources and clients that isolates each side from changes and modifications on the other side. We presented architecture for such a warehouse that can be implemented by cities and municipalities based on readily available platforms and software and is portable except for the Extractors, which are site-specific. We have also presented a simple object model or schema that can be used to present data to clients in a unified format that is independent from the data sources themselves.

The data model and data warehouse establish a collaborative environment among the various authorities influencing urban design, planning, and management in cities such as Makkah and the planners and designers involved in urban decision-making. The warehouse architecture we presented can be extended to make the environment more cooperative and to generate data from the existing data to provide more advanced support to urban decision makers.

8 ACKNOWLEDGEMENTS

The work reported here was supported by The Custodian of the Two Holy Mosques Institute for Hajj Research, Umm Al-Qura University, Makkah, Saudi Arabia.

9 REFERENCES

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language. User Guide*. Addison Wesley Longman, New York.

Bhargava, H. K., King, A. S., and McQuay, D. S. (1995a). "DecisionNet: An Architecture for Modeling and Design Support over the World Wide Web". In *Proc. International Symposium on Decision Support Systems*, Hong Kong, June 1995.

Bhargava, H. K., Krishnan, R., and Muller, R. (1995b). "On Parameterized Transaction Models for Agents in Electronic Markets for Decision Technologies". In *Proc. Fifth Workshop on Information Technologies and Systems*, Amsterdam, Holland, December 1995.

Dave, B., and Schmitt, G. (1994). "Information Systems for Urban Analysis and Design Development". *Environment and Planning B: Planning and Design*, vol. 21, 83-96.

Kennedy, K. and Kopp, K. (2001). *Understanding Map Projections*. Environmental Systems Research Institute, Inc. (ESRI). Redlands, California, USA.

Koshak, N. (2002) *Object-Oriented Data Modeling and Warehousing to Support Urban*

Design. Ph.D. dissertation (in preparation). School of Architecture. Carnegie Mellon University, Pittsburgh, Penn.

Meyer, B. (1988) *Object-Oriented Software Construction*. Prentice-Hall, New York.

Naiburg, Eric J., and Maksimchuk, Robert, A. (2001). *UML for Database Design*. Addison-Wesley, Reading, Mass.

Papakonstantinou, Y., Garcia-Molina, H., and Widom, J. (1995). "A Query Translation Schema for Rapid Implementation of Wrappers". In *Proc. Fourth International Conference on Deductive and Object-Oriented Databases*, Singapore, December.

Rob, P., and Carlos, C. (1997). *Database Systems: Design, Implementation, and Management* (3rd ed.). Course Technology, Cambridge.

Rumbaugh, J. et al. (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, New Jersey.

Silberschatz, A., Korth, H., and Sudarshan, S. (1999). *Database System Concepts.* McGraw-Hill, New York.

Wheeler, J. and Wheeler, W. (2001). "Achieve Persistence Independence". *Java Pro Magazine*, November 2001, Volume 5, Number 11.

Yin, Z., and Williams, T. H. L. (1995). "GIS Analysis Model for Urban Strategic Planning". In *Proc. International Symposium on RS, GIS, and GPS*, Hong Kong, 746-753.