TOOLS AND TECHNIQUES FOR THE TECHNOLOGICAL INTEGRATION OF MULTI-HAZARD POST-INCIDENT ASSESSMENT

Final Report

NSF Grant # CMS-0353175

Dr. Debra F. Laefer
Dr. William Rasdorf
Anu R. Pradhan

DEPARTMENT OF CIVIL, CONSTRUCTION,
AND ENVIRONMENTAL ENGINEERING
NORTH CAROLINA STATE UNIVERSITY
RALEIGH, NC

JANUARY 2006

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The September 11, 2001 attacks on the World Trade Center (WTC) acutely demonstrated the need for an integrated management information system (IMIS) to assist in simultaneously assigning governmental services and resources for regular programs and emergency response (Flax et al. 2002). Although the majority of American cities now have electronic base-maps to facilitate urban planning, few of these base-maps have disaster management capabilities. Even amongst those that do, there are significant functional limitations with respect to (1) accessing data collected by non-disaster management branches of government and (2) supporting data analysis capabilities. Impediments exist because of organizational, computing, and charter characteristics of these systems (e.g. Uddin and Engi 2002).

The challenges of data access, spatial querying, and information sharing across multiple users, particularly during disaster response have been identified as major technical challenges (Goodchild 2003, Mansourian et al. 2005, Zerger and Smith, 2003). These obstacles exist in large part due to the proprietary, standalone, and segregated nature of the current disaster management systems (DMSs), which prevents participation in efficient data gathering and sharing capabilities beyond the purview of individual departmental authority. Consequently, these systems are of circumscribed utility, when responding to major disasters (Greene 2002, Flax et al. 2002, Cutter 2003). A new approach to the design and implementation of IMISs for managing disasters is needed. This report assesses various disaster related phases in terms of the emergency response community, outlines current system limitations, defines essential system features, and introduces a portion of DIsaster Operations, Risk, Abatement, and MAnagement (DIORAMA), a recently prototyped disaster management system (DMS) developed by the authors.

The initial motivation to develop DIORAMA was to create a system capable of supporting highly detailed, blast-damage analysis for urban areas that had been subjected to high explosive detonations. Patterns of energy dissipation and blast-damage cannot be understood thoroughly, without powerful three-dimensional (3D) querying capacities. Since inclusion of actual pre-and post-blast condition data of building components, as well as entire structures, was deemed crucial to comprehending vulnerabilities, extensive data collection of the built-environment became paramount. Such information was essential both in terms of post-incident performance assessment and in applying those assessment outcomes to other structures. As such, a continuous cycle of information was needed to analyze urban explosions and their attendant hazards throughout all phases of the disaster management process. In this report, the requisite computer architecture for such a disaster management system is illustrated and multi-hazard, disaster analysis needs are used to illustrate how the concepts here presented are widely applicable.

1.1 Disasters

A disaster is defined as a “sudden calamitous event bringing great damage, loss, or destruction” (Merriam Webster 2003). A sampling of disasters may be generalized into interrelated or similar groupings based on the cause, location, and type of community affected (Table 1). Although these broad categories include events that are highly distinct in their nature, environment, and impact, many have commonalties with respect to the data sets required to address these disasters in a multi-phase context (Figure 1). A simple example is in the planning and execution of large-scale evacuations. Such evacuations may be necessary during earthquakes, hurricanes, forest fires, and nuclear contamination. To find proper evacuation routes, the same road network data set is required, irrespective of the disaster. By focusing on common data sets, although not fully reflective of any particular disaster or specific phase of the disaster management cycle, an initial framework can be created from which to prototype a system to address a wide range of disasters.

The premise of basing disaster management systems on common data sets was recently confirmed by a group of over 30 emergency management personnel in the state of North Carolina (Harris et al. 2005).

Such an approach holds promise for addressing the complexities of disaster management. Unfortunately, as will be fully described below, the emerging geographic information system (GIS) based solutions being developed by governmental agencies and private companies do not accommodate the multiple phases that occur as part of the disaster management process.

Table 1 Sampling of Disaster Categories

Disaster	Cause		Location			Impacted Community	
	Natural	Manmade	Land	Water	Air	Urban	Rural
Tornado	X		X	X	X	X	X
Hurricane	X		X	X	X	X	X
Flood	X		X	X		X	X
Ice storm	X		X			X	X
Earthquake	X		X	X		X	X
Building collapse	X	X	X			X	
Explosion		X	X		X	X	
Fire	X	X	X			X	X
Hazardous liquid spill		X	X	X	X	X	X
Biological outbreak	X	X	X	X	X	X	X
Nuclear contamination		X	X	X		X	X
Oil spill		X	X	X		X	X

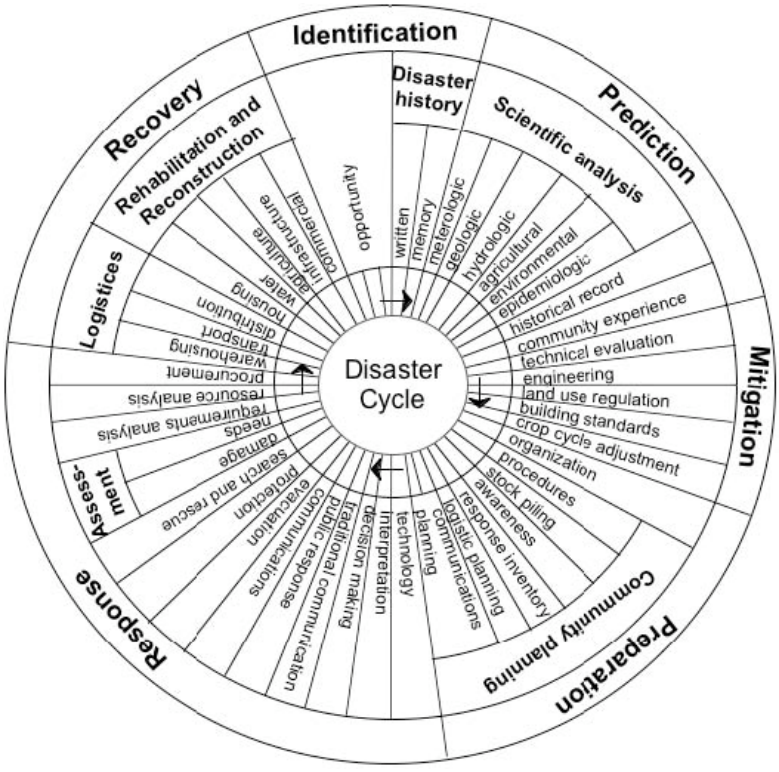


Figure 1 Disaster Preparedness Phases (adapted from Johnson 2003 and USDFA 2003)

1.2 GIS Use in Disaster Management

Increasingly, community vulnerability to natural and manmade disasters is being recognized (Godschalk 2003). The level of vulnerability is an indication of the extent of the community's disaster readiness. Although disasters occur in a wide variety of manifestations, all happen within a specified geographic area, and, thus, all disaster analysis must be viewed in relation to that location (Johnson 2003). Understanding geographically-based information is critical to protecting communities from disasters (Radke et al. 2000).

The general availability of computing as a routine part of community planning and management facilitates an information technology role in decision-making before, during, and after disasters. For instance, during a disaster, multiple emergency management responders need to share information, which requires access to databases and computer-generated maps. Simultaneously, they need to perform mission-critical, data analysis to assist search and rescue operations (ESRI 1999). GIS-based tools have proven to be important aids in disaster preparation and rapid response (Johnson 2003). Digital maps and spatial analysis tools (enabled by GIS) offer a common template on which a wide variety of data can be quickly placed, measured, and analyzed, which can result in better informed decisions (Greene 2002). Thus, to be truly prepared for a disaster, emergency managers need a GIS-based tool to organize, integrate, analyze, and distribute data associated with various phases of disaster management.

The applications of GIS range from identifying evacuation routes during flooding to determining hazard zones during forest fires. GIS is being increasingly favored, because of its capacity to associate existing databases with geographic features, to query existing databases based on the geographic features, and to render multi-dimensional visualization of spatially queried data. These capabilities are especially useful in relation to disaster events, because of the dual spatial and temporal nature of these incidents. For instance, GIS can help both identify the likely path of a flood based on topographic features and determine the level of flood at a particular time. Similarly, for forest fires, GIS can help both identify areas of highly flammable vegetation near populated areas and calculate the evacuation time based on the rate of fire expansion.

Advances in GIS can play an important role in smart decision making before, during, and after a disaster. For example, urban planners need detailed information related to arrangements of buildings, roads and utilities to conduct risk assessments, while emergency personnel require much of the same basic information for search and rescue operations during and after a disaster. Digital maps and spatial analysis of this kind can be produced by GIS-based system, where a common template is created on which varied data can be quickly placed, measured, and analyzed – resulting in more well-informed decisions (Greene 2002).

1.3 The Six Disaster Phases

The authors propose that the success of any disaster management system is dependent upon capabilities in six disaster-related phases: (a) Identification, (b) Prediction, (c) Mitigation, (d) Preparation, (e) Response, and (f) Recovery (Figure 1). Within DIORAMA these terms have been defined as follows.

- **Identification** is an inventorying of a community's assets focusing mainly on the physical infrastructure (e.g. roads, buildings, bridges), although human-oriented capacities such as fire stations and hospital beds may also be included.
- **Prediction** is the conceiving of a danger and assessing its potential impact on the community, which may include property damage, service interruption, and loss of life. Prediction also estimates losses for a particular disaster of a certain magnitude for a specific community.
- **Mitigation** is a lessening of the negative impacts of the event and may range from altering parking and traffic patterns around high-risk buildings to changing the building code.

- **Preparation** includes those activities needed to prepare for emergencies that cannot be fully mitigated against either because of cost or the nature of the disaster. Preparation may include expanding emergency services, identifying emergency housing areas, or posting of evacuation routes.
- **Response** occurs during and immediately following a disaster and is typically focused on rescue activities, fire fighting, and looting prevention.
- **Recovery** is the rebuilding of the community and restoring services.

To illustrate these six phases and to demonstrate the multi-disaster applicability of the proposed structure, a hurricane evacuation scenario is provided. To minimize hurricane-induced losses, each of the above-defined phases must be addressed within the disaster management system (DMS). The DMS must host the required data and support relevant analytical capabilities.

- Identification designates at risk communities and other key infrastructure elements vulnerable based on past hurricanes and anticipated weather trends.
- Prediction establishes the level of vulnerability of a particular town or region to a hurricane of a specific magnitude; for a hurricane this would include damage caused by both high winds and flooding, and their impacts on transportation, housing, and commerce.
- Mitigation includes changing the zoning laws thereby decreasing the number of at risk facilities, altering the features of the coast by dredging, beach restoration or installation of wave breakers.
- Preparation determines safe routes for hurricane evacuation and makes such information available. Changing conditions must be reflected and communicated without any delay (e.g. due to alteration of the hurricane's path).
- Response encompasses the immediate rescuing of individuals and arranging of their subsequent housing and feeding, as well as road closing.
- Recovery includes debris removal, building inspection, and bridge reconstruction. Prioritization schemes need to be developed based on real-time information.

The four phases of Identification, Prediction, Mitigation, and Preparation can be considered temporally as pre-disaster. In contrast, the phase of Response occurs while the disaster is happening, and Recovery is strictly a post-disaster phase. These six phases illustrate the importance of the interrelationship of the various phases before, during, and after a disaster. Having all temporal aspects addressed is essential, as the information pertinent to one phase becomes vital to other phases. For instance, the analysis done during the Preparation phase informs the Response phase. Choices during Response impact Recovery strategies. Similarly, post-damage data collected during the Recovery aids in the Identification, Prediction, Mitigation and Preparation aspects for future disasters. A DMS must be able to address systematically all six phases, because of their intrinsic inter-dependency.

2. LITERATURE REVIEW

The required system must accommodate several levels of data entry and access for each temporally-based phase in a systematic way. At this most fundamental level, there must be a map depicting streets, utilities, buildings, and other major components of infrastructure in an electronic format. The next level of functionality is integration of associated attribute data with these key features of the infrastructure. A variety of feature attributes must be assigned to each component of the infrastructure (e.g. for a utility, its size, shape, age, and depth of embedment). These key features of the infrastructure should be linked to their visual representations in an identifiable manner (e.g. gas lines distinguishable from water lines, locatable emergency services). Recording of data must be mutable (as external factors change) and capable of integration with other mutable data. Thus, in its most elementary form the conditions, collectively known as a baseline data system of a community, must be represented. The baseline data system can be defined as the scope and nature of the overall informational resources needed to prepare for and respond to a variety of disasters.

Apart from the establishment of baseline system, the disaster management system should support complex spatial analysis and querying (e.g. finding the shortest path for the safe evacuation during a hurricane). GIS-based systems help to consolidate the establishment of such baseline system and provides powerful spatial analysis capabilities thus making them indispensable for disaster management systems. Finally, the system must be configured to meet real-time data collection and dissemination needs of disparate agencies that occur during a disaster (Figure 2).

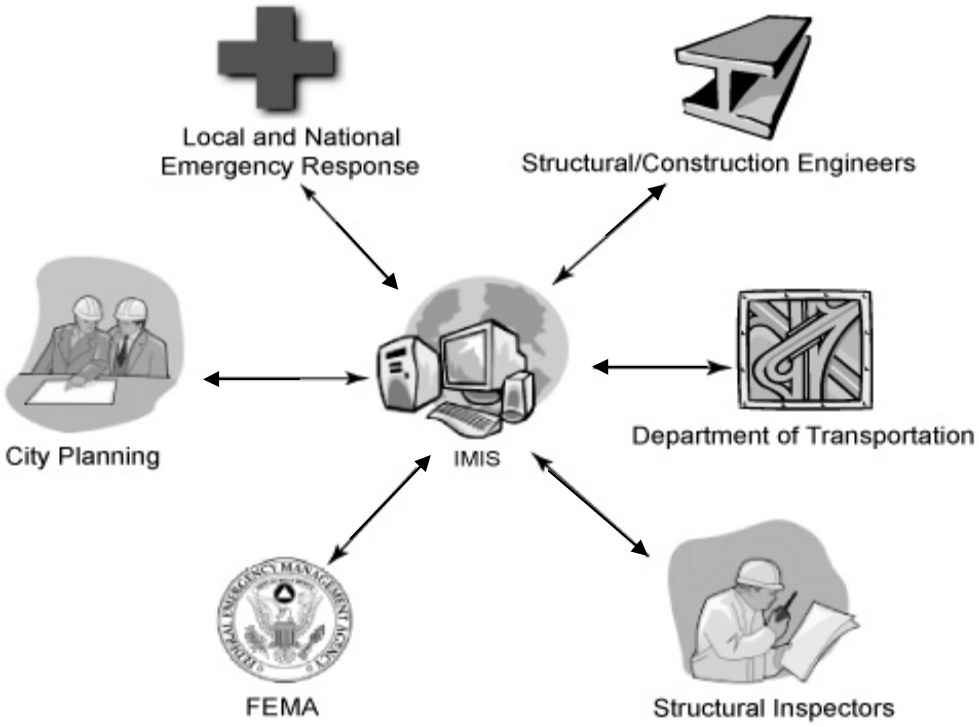


Figure 2 Disaster Community Need

2.1 Existing Systems

Unfortunately, despite multi-billion dollar investments in preparedness and simulated training exercises (GAO 2003), the majority of American cities and municipalities are still inappropriately or minimally prepared for disasters as clearly demonstrated by emergency response to hurricanes Katrina and Rita. Reasons range from the fiscal to the political, but the computer architecture of many of the best known DMSs have inherent computing limitations that prevent their full applicability to disaster management. Existing systems are typified by discrete, non-networked arrangements that are largely ineffectual in disaster prediction, and completely incapable of assisting in various phases of disaster preparedness as depicted in Figure 3. Thus, a system founded on robust enterprise-level architecture capable of supporting multiple users concurrently and processing complex analysis in real time is required, while providing adequate redundancy for possible data transfer failures.

2.1.1 Inadequacies of Current Systems

Some cities such as New York have established baseline systems with digital maps of public utilities, road networks, and building footprints. Unfortunately these systems are typically standalone and of circumscribed functionality, because they lack the capability to exploit the stored data for analysis (Flax et al. 2002). The establishment of a baseline system represents a necessary but insufficient condition for a viable, fully functional, multi-phase system. Of those cities with baseline systems, only a small percentage of them have disaster related capabilities, and of those, no city has yet designed and implemented a computer system capable of supporting the requisite Response and Recovery that are critical during a major natural or manmade disaster. Instead, emergency aspects of Response and Recovery are strictly supported by 911 calls and aerial surveillance, which do not support other aspects of disaster management.

Existing systems are typified by discrete, non-networked arrangements that are largely ineffectual in many aspects of disaster management. Two of the best-known DMSs are known by their acronyms HAZUS and CATS. Evaluations of these and the Denton and Frost commercial system are presented below, with respect to multi-phase disaster needs.

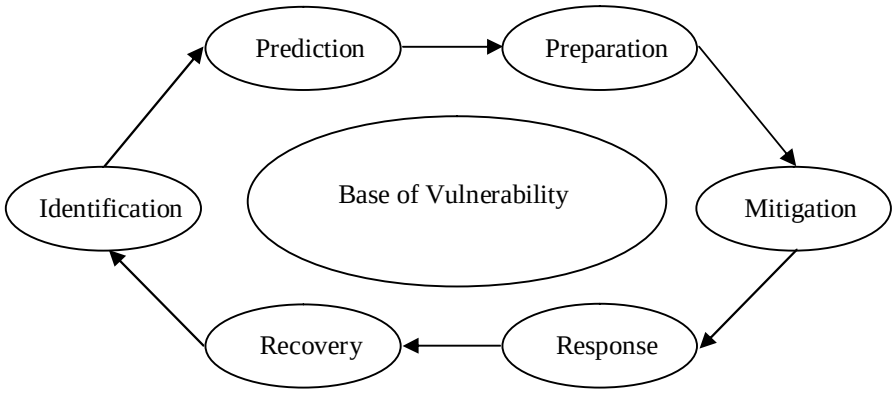


Figure 3 Disaster Preparedness Phases

2.1.2 HAZUS-MH

The Federal Emergency Management Agency (FEMA) developed a hazard prediction program under contract with the National Institute of Building Sciences: Hazards United States – Multi-hazard (HAZUS-MH) uses GIS software to map and graphically depict hazard data, economic losses, and buildings and infrastructure damage from hurricanes, floods, and earthquakes (HAZUS 2005). The GIS technology combines hazard layers with national databases, including datasets of demographics, building inventory, critical facilities, transportation systems, and utilities. The system applies both loss estimation and risk assessment methodologies for a limited set of disasters.

Although HAZUS-MH is equipped to model the probability and level of pre-incident risk for a variety of disaster events, a major limitation of the system is its incapability to support real-time response and recovery activities and multiple device types (ubiquitous computing) following a disaster. Thus the software addresses only identification, prediction, and preparation phases of Figure 3.

2.1.3 CATS

The Consequences Assessment Tool Set (CATS) is a product of the Defense Threat Reduction Agency in conjunction with FEMA (DTRA 2002). CATS is an ArcView application for predicting hazards from hurricanes, earthquakes, weapons of mass destruction, and conventional explosives and for assessing the ensuing consequences. CATS allows the user to simulate scenarios for training and planning, assess affected population and infrastructure based on real-time weather information, initiate resource deployment, evaluate further needs, and locate resources for a sustained response. As CATS is primarily a training tool, it cannot support real-time Response and Recovery activities (Table 2).

2.1.4 DMI-Services

Similarly, Marine Corps Systems Command and FEMA have initiated another multi-million dollar project named Disaster Management Interoperability Services (DMI-Services) to provide a shared "information backbone" regarding emergency information management among emergency responders, government offices, and authorized non-government organizations during a crisis (DMI-Services 2003). The intent of this service is to leverage existing databases and applications to provide an interoperable suite of tools that organizations can use to obtain needed capability, information, and seamless connectivity to other stakeholders in the incident response community. Though this system provides a strong information background, it lacks the tools needed to do proper damage assessment before, during, and following a disaster. Therefore, this system provides the basic "information backbone" infrastructure to address response and recovery phases of Figure 3, but does not provide the adequate tools to do proper analytical assessment of any of those phases.

2.1.5 Deaton and Frost Commercial System

There are also commercially available programs for post-disaster data consideration, such as the one developed by Deaton and Frost (2001), which integrates a global positioning system, GIS software, digital photography, and handheld computing technology. The system allows both quantitative and qualitative data collection, but primarily focuses on baseline and post-disaster data collection. Thus, this system addresses only the Identification and Recovery phases (Table 2).

In summary, readily available public and private systems each support only some of the six disaster management phases. Thus, they cannot process disaster-related information within the context of an integrated system.

Table 2 Existing System's Capabilities

Existing Systems / Disaster Related Phases	Hazus-MH	CATS	DMI-Services	Deaton & Frost	DIORAMA (Present + Future)
(a) Identification	**P**	**P**		**P**	**P**
(b) Prediction	**P**	**P**			**F**
(c) Mitigation	**P**	**P**			**F**
(d) Preparation	**P**	**P**			**F**
(e) Response			**P**		**F**
(f) Recovery			**P**	**P**	**F**

(Note: P stands for present features of the system, while F stands for future additions to the system)

3. SYSTEM REQUIREMENTS

An ideal DMS should be enabled for performance in all six disaster management phases within a single integrated system. To achieve this, a DMS must have three components: (a) a baseline system, (b) spatial querying, and (c) ubiquitous computing.

3.1 Baseline System

A baseline system establishes the basic information backbone required for a DMS (Laefer et al. 2005a). To best achieve deployment and use of a disaster management system, it should be sharable across local, state, and federal jurisdictions, thereby reducing duplication of data collection efforts across organizational units. The development of data collection methods for pre- and post-disaster related activities using a multi-hazard approach is critical to this effort (Uddin and Engi 2002, Uddin et al. 2003). All of this, however, is predicated upon a strong consensus amongst the necessary players as to what data is to be collected, the collection mechanism, and its format.

3.1.1 Common Data Set

The multi-event nature of a disaster can be considered in a more systematic method in terms of commonalities between disaster events. Disasters can be generalized into interrelated or similar groupings, such as cause (i.e. natural or manmade), location (e.g land, water, air), and the type of community impacted (urban or rural). Although these broad categories include events that are highly distinct in nature, intensity, geographic distribution, and duration, many share commonalties related to the data sets required to address these disasters in a multi-phase, multi-event context.

For instance, evacuation may be a required process during earthquakes, hurricanes, forest fires, and nuclear contamination. To find proper evacuation routes, the same road network data set is required, irrespective of the disaster. Yet an effective DMS system must be more than a repository of street layouts and fire hydrant locations. A more detailed level of knowledge is needed. For example, safe congregation areas should be identifiable for emergency medical treatment administration and decontamination. This requires identification of empty lots, barricadable roads, and public assembly structures, as well as an assessment of their occupancy capacity and their proximity to water and electricity.

The conclusion of a 2004, two day, National Science Foundation workshop of over 30 participants from North Carolina's academic and emergency response communities resoundingly concluded that the vast majority of both pre-and post-disaster needs from an emergency response perspective could be met effectively by adopting a multi-phase, mutli-event set of generalizations (Harris et. al. 2005). Instead of relying upon a disaster-specific paradigm, by focusing on common data sets, although not fully reflective of any particular disaster, an initial framework to prototype a system to address a wide range of disasters could be successfully established. Since some indispensable data sets are pertinent to only specific disasters (e.g. elevation data to establish water heights for flooding), all relevant data sets (both common and disaster specific) must be integrated into one common repository system to fully manage disasters (Figure 4). The primary motivation for an integrated data repository system is to ease data access across multiple organizations (Kevany 2003).

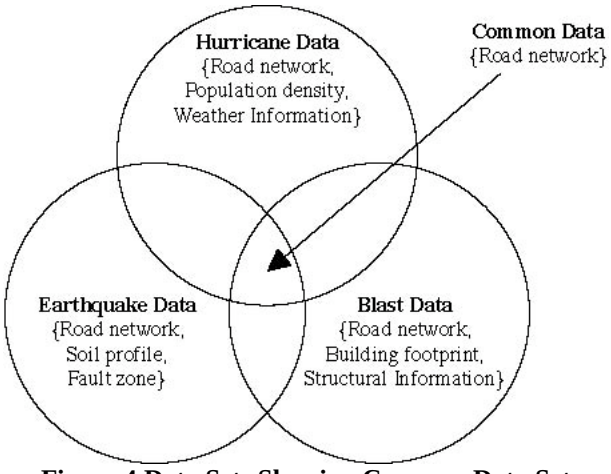


Figure 4 Data Sets Showing Common Data Set

All of this, however, is predicated upon a strong consensus amongst the necessary players as to what data is to be collected, the collection mechanism, and its format. What is presently needed is guidance in the prototyping of a data collection system that is appropriate for urban planning, risk analysis, and disaster management. This report outlines the requirements to prototype such a data subset and the general characteristics of the data beyond its mere content, specifically in the four areas of **comprehensiveness**, **accuracy**, **timeliness**, and **accessibility**. The applicability of each is shown through its application to a common data set.

3.1.2 Common Data Characteristics

For an effective DMS baseline data system, the following common data needs were identified as part of the 2004 National Science Foundation workshop: (1) unique designators for infrastructure elements, (2) access routes, (3) safe congregation areas, (4) building egress points, (5) building structural support systems, and (6) high vulnerability elements. These six common needs are used to illustrate the common data characteristics of **comprehensive, accurate, timely,** and **accessible**. The emphasis in this report is on buildings, because in many ways they represent the greatest data collection challenge, because their ownership is more distributed than publicly financed projects, and because their construction exhibits an extraordinarily high level of variability in composition, geometry, and usage.

3.1.2.1 Comprehensive

The term **comprehensive**, in relation to data collection, should not be misunderstood to signify everything that can be collected. If a community collected and electronically stored each detail of every constructed facility in all of its minutia, the likelihood that the system could be accessed quickly and efficiently for disaster response would be low – simply as a direct outgrowth of the magnitude of the data being stored. What is needed to adequately depict the multi-phase, multi-event nature of a disaster is a subset of the infrastructure data. Such data relates to road networks, utility systems, building layouts, and the relationships between all of these elements. Collection of such disparate data requires consensus as to both scope and format. Cities and municipalities cannot mandate data collection from facility owners and utilities without adequate knowledge of the *scope* and *format* in which to specify the collected data. The concepts are illustrated below using the above listed, six common data needs.

3.1.2.1.1 Scope

Although the topic of urban disasters is broad in subject area, it is limited in its geography. As such, most urban disasters can be seen to have a large common data set. The scope of such data can, thus, be fairly

clearly specified. As an example, unique designators for infrastructure elements would consist of street addresses (or cross-streets) and GPS positions. Access routes between points of interest for emergency vehicles require information on the transportation grid of roads, tunnels, and bridges, especially height, width, and weight restrictions due to the potential need to transport heavy rescue and recovery equipment. Information for safe congregation areas must extend beyond knowledge of street layouts and fire hydrant locations. For staging temporary medical treatment areas and creating post-evacuation meeting points, parking lots, wide sidewalks, and other unobstructed areas should be identifiable, as well as information about their proximity to water and electricity. To facilitate search and rescue, immediate identification of all above-ground and subsurface egress points should be known for all multiple-occupancy and public assembly structures. Building floor plans and elevations are needed to identify doors, windows, and stairwells. Subsurface transportation routes and large-diameter, utility line layouts are also required.

The structural aspects of buildings impacted by disasters also need to be known for potential damage assessment, as well as for risk evaluation of adjacent structures. The type and layout of the structural support system (e.g. unreinforced masonry, steel skeleton, prefabricated concrete) and its key features must be known, along with all high vulnerability elements (e.g. glazing and glass cladding) via plans and section drawings. A full set of structural drawings is ideal to secure this knowledge but may not be reasonable due to data storage requirements. Furthermore, post-disaster building inspectors could preview each building's structural system and cladding details just prior to its post-disaster, field inspection for improved accuracy and safety. As personal digital assistant systems continue to evolve in their applicability to disaster management (Deaton and Frost 2001), the value of such information to on-site rescue and recovery personnel will continue to increase (Huyck and Adams 2003). The scope of the available data in an IMIS system should be reflected in a detailed catalogue viewable by entity, quantity, source, and format.

3.1.2.1.2 Formats

A major impediment to an effective IMIS is finding existing data in a format that can be stored and retrieved in a rapid and understandable manner. Data formatting is especially important as it may restrict or promote cross-format sharing and ultimately usability. For example, the data harvesting and integration efforts in New York after September 11 were severely hampered by the wide variety of data types and formats collected by each of the contributing entities (Huyck and Adams 2003). As a result, only the most rudimentary conversions and connections were made between the data (Flax et al. 2002). To facilitate ease of use for both data entry and retrieval, there should be a predefined set of data categories available for each database field, in addition to open querying.

Presently, data may be in the form of a spatial representation (where a structure or element is physically located, as in the case of access routes and safe congregation areas), an attribute (e.g. sizing, material characteristics, age for high vulnerability elements and unique designators for infrastructure elements), or a graphical depiction (photographs, videos, or drawings of building egress points and building structural support systems). Of particular interest are the spatial and geometric features of the total infrastructure and of specific building geometries, structural systems, and cladding materials. Relevant information should appear in a relational database so that it can be queried. The information should also appear in 3D, GIS manifestations.

In many GIS software packages, both spatial data and attribute information are stored in proprietary file formats (e.g. ESRI shapefile, ESRI coverage, AutoCAD dxf). Proprietary file formats are not based on standards, thus integration is hindered. International geospatial standards for GIS minimize the number of geospatial data formats in the marketplace, with the goal of reducing duplication, optimizing resource expenditure, and facilitating information exchange (Clément and Larouche 2000). Recently developed standards by both governmental and non-governmental organizations promote data sharing and

integration. Among the most useful standards (in terms of completeness, flexibility, and life expectancy) are the Simple Features Specification and the Spatial Data Transfer Standard (SDTS 1997). The Open GIS Consortium (OGC) organization, was established by the United States Geological Survey USGS for standardized data exchange (OGC 2003). These standards aid data digitization, modeling, and storage (SDTS 1997).

3.1.2.2 Accurate

Accuracy is the second key characteristic for baseline data (Rasdorf 2000). Broadly speaking, data errors emerge from two sources: (1) new data entry and (2) errors in existing records. The second scenario is complicated by the fact that the data is retrieved from disparate sources (because of incomplete existing records), which may themselves contain inaccuracies. For older cities, where buildings may date as far back as the eighteenth century, much of the data may not be available in any format. Even important landmark buildings may be without known as-built drawings, as was the case with Carnegie Hall until late, twentieth century renovations (Laefer et al. 2002). In reference to egress points, structural support systems, and high vulnerability elements, the lack of as-built drawings poses a substantial impediment. Accurate as-built drawings hold within them the data that empowers the base IMIS and all supplementary pre- and post-disaster analyses. Much of this data is available from existing sources.

3.1.2.2.1 Existing Data

Existing data sources may include Sanborn maps (ProQuest UMI 2003), local colleges, design offices, government agencies, architectural organizations, historical societies, older community residents, and Internet sources. Unique designators, and access route information may be available from governmental records, and maps (both digital and non-digital). Identification of safe congregations areas requires those above-listed elements, as well as building typology data and capacity, not dissimilar to the base information needed for egress points, structural support systems, and high vulnerability elements. The focus is the building's geometry and nature and how it is assembled. Critical data should be verified via multiple, published and web-based sources to minimize new errors and to prevent further propagation of old ones.

3.1.2.2.2 New Data

To reflect the evolving nature of a community, new data must be collected continuously. The occurrence of events (both disaster and non-disaster related), from flooding to new road construction must be documented promptly and accurately to exploit the full potential of the IMIS. This new data must be combined and integrated with the existing data within the IMIS. Each update establishes a new baseline condition (Lim et al. 2001).

3.1.2.3 Timely

The third criterion is timeliness. With existing and new data constantly being added, a baseline system must provide efficient data storage, because of the need to obtain real-time data as a disaster is occurring. Decision makers, incident commanders, emergency responders, and city managers simultaneously need time-sensitive, spatial information. Since timing is critical for evacuation, search and rescue, and prioritization of utility service restoration, all required data sets must be pre-stored in a centralized repository system. Such pre-emptive activity ensures that mission-critical data can be retrieved and disseminated amongst the necessary personnel and agencies for timely analysis.

A few communities currently require certain classes of buildings to be inspected on a periodic basis for public safety [e.g. New York City's Local Law 11 related to façade inspections (NYC DOB 2005)]. Where available, damage information from such reports should be incorporated into all relevant portions of the IMIS (i.e. CAD, photos, relational database), with an indication as to the date of observation and

the severity of the observed damage. Additionally, all images must be date stamped, and data must be entered into the system contemporaneously with any changes that occur. The building permitting process provides a likely venue for this. Currently, construction information is submitted to the local building authorities as the basis for permitting and inspections. Depending upon building usage, different levels of documentation are required for framing, general layout, foundations, and utilities.

Once a permit is issued, many accuracy-related problems may arise. Plans may change after the permit is issued, either because of differing site conditions or the owner's evolving needs, all while the information on file remains unchanged. The accuracy of as-built drawings is a regular problem in the construction industry, even for non-digitized drawings required by contract at the end of a project.

A fully functioning IMIS requires a high degree of compliance in the submission of accurate, as-built drawings to the permitting agency. This will require a change in culture, as well as procedure. The current manual method still used by most communities is dated and inefficient, resulting in delays and higher costs from unnecessarily slow processing time. As documents wait for approval, so does the construction. Manual data entry offers few benefits and requires enormous resources from both owners and public officials. Consequently, data is often simply not updated, limiting the usefulness of the file records. Keeping the system updated, however, is a major task, which requires constant inputting of new and changed data, as well as regular error checking. Structures and sites are dynamic. Thus, record systems must reflect these changes, whether this is a modification to access routes in the road network or alterations of egress points in a school building, or the unique designators of major infrastructure elements due to bridge widening.

The permitting process has the potential to become highly automated and entirely electronic. The permit issuer could connect to the IMIS, allowing for an immediate updating of proposed changes for local structures (both existing and new), with electronic copies of plans and specifications attached for ease of retrieval. The IMIS could be internet-enabled and accessible on a limited basis via a username and password. This would allow secure access at virtually any internet-enabled location. Automation of the permitting process has the dual benefit of ameliorating the aforementioned accuracy issues and keeping the IMIS updated. As part of the attribute data, drawings and photos can be archived to reflect the evolution of a building or site, potentially providing long-term planning benefits for disaster and non-disaster scenarios. Changes in the building's structural support systems, and high vulnerability elements aspects need to be accurately reflected, especially in the aftermath of a crisis, when a multitude of changes may need to be documented. Instead of being overwritten, the data can be archived in the same manner as the drawings and photos are, allowing the user to see both a visual and text oriented timeline of the evolution of the building, including any disasters that have damaged the building and any subsequent repairs.

3.1.2.4 Accessible

Easily accessed data capable of being shared across federal, state, and local jurisdictions is fundamental to the decision-making capability of those charged with community protection, as there are many infrastructure elements that are not wholly controlled by local building authorities. These range from the obvious (e.g. state highways), to high profile targets such as federal buildings. Yet, without the real-time capacity to visualize activity patterns, maps, locations, and major features, disaster management will not be effectively achieved (FGDC 2002). Rapid access to and application of accurate geospatial data is critical to these activities. Similar to the electronic permitting process, during a crisis a community may find it highly beneficial for the IMIS or portions of it to be accessible via the Internet, such as access routes, safe congregation areas, and high vulnerability elements. Each community must individually assess security concerns related to having such information on an externally-accessible network, as security breaches are a potential downside to easy access. Additionally, issues of large-scale web

accessibility do not address a wide variety of computer processing concerns related to concurrent data entry, data conflict resolution, and redundancy (Pradhan 2003).

Finally, an in-depth understanding of images must be established as to an appropriate subset of data so that content can be readily archived in a manner that is meaningfully catalogued and available for retrieval, analysis, and display. A methodology of pictorial collection standardization is a critical component for a fully functioning IMIS. More difficult than culling visual components for textual entry into the relational database is having the capacity to query graphical elements in a highly exact manner. There must be a visual marking and querying system based on both a point and click system and with a textual tagging component. To date such a system is not yet available.

3.2 Spatial Querying

A DMS should support the complex spatial analysis required for all of the six disaster-related phases. As mentioned earlier, from identification to prediction, mitigation to preparation, response to recovery, each phase inevitably leads to the next, in a continual quest for a safer environment in which to live and work (Johnson 2003). Complex spatial queries offer unprecedented opportunities for advances in community safety. Examples of useful spatial queries include locating the shortest evacuation path and optimizing post-disaster resource distribution (e.g. shelters, food depots) for displaced people.

3.3 Ubiquitous Computing

Ubiquitous computing enables multiple computing devices (e.g. workstations, laptops, and handheld devices) to be simultaneously and seamlessly available as a composite system (Weiser 1993). Such architecture would maximize existing resources and provide the highest level of functionality for various end users, as different types of devices have various capabilities from simple data collection to resource intensive computing. For example, handheld devices can play an important role in post-disaster data collection, because of their size and portability (Deaton and Frost 2001). Similarly, resource intensive analysis (most pertinent to identification, prediction and response) needs to be done with powerful workstations. Thus, a system is required that facilitates the use of multiple computing devices of varying power levels and capabilities, because a DMS should be accessible to many different clients. The concept of ubiquitous computing should also be expanded to include peripheral hardware including data collection and measurement devices and gages. At present, such systems exist to a limited extent at specific academic institutions (e.g. Liu et al. 2003, Pena-Mora 2001), but are not part of standard disaster management systems such as HAZUS-MH and CATS.

4. COGENT DATA FRAMEWORK

To have an effective disaster management system, three criteria must be met:

- Cogent data framework (CDF),
- Efficient infrastructure management information system architecture, and
- Adequate data analysis capabilities.

This section addresses the first criterion, the establishment of a cogent data framework, by defining the roles that data play in the construction and usage of an infrastructure management information system. The basis on which to judge the framework's cogency is exclusively reliant upon the rules, context, and format in which the data is collected, stored, and updated, as will be outlined in detail in this section.

4.1 Requirements

For a data framework to be cogent, it must accommodate four key components: **depiction, integration, connection**, and **evolution**.

- **Depiction** is the physical representation in a digital format of the physical environment including streets, utilities, buildings, and other major infrastructure components.
- **Integration** is the combination of various required data sets into a single, integrated repository (e.g. layouts of all utility systems).
- **Connection** is the establishment of relationships between key infrastructure features and various attribute data (e.g. size, shape, age, and depth of embedment of a section of a utility line) and their linkage to identifiable visual representations (e.g. a gas line versus a water line).
- **Evolution** is the periodic updating of a data set reflective of changes instigated by various external factors (e. g. replacing or extending utility lines).

For an IMIS to be effective, a community's infrastructure must be represented including building, utilities, roads, and bridges. In many ways, buildings represent the greatest challenge, because their construction and ownership is more distributed than publicly financed projects or private utilities (Kevany 2003). This distributed ownership leads to an extraordinarily high level of variability in their composition, geometry, and usage. As such, the topic of buildings will be used to provide a detailed illustration of the four key components listed above.

4.1.1 Depiction

The first step towards establishing a CDF is the depiction of the built environment. Depiction includes data selection, collection, and digitization. The data may include various data sets including photographs; digitized and paper drawings of architectural and structural elements; transportation network information; utility maps; periodic building or façade inspection reports; GIS maps; and GPS building and street locations. The first subtask in depiction is selection.

4.1.1.1 Selection

Intelligent data selection is critical. If a community collected and electronically stored each detail of every constructed facility in all of its minutia, the likelihood that the system could be accessed quickly and efficiently for disaster response would be low – simply as a direct outgrowth of the magnitude of the data being stored (Laefer et. al. 2005a). What is needed is a subset of the essential infrastructure components required for effective disaster management system.

Consensus across the disaster management community must be established as to requirements for data selection, both scope and format. Depending upon the needs, concerns, and resources of a community, this may be as basic as all structural drawings and all floor plans for every structure, except single-family homes. In contrast, the needs may be as elaborate as identifying glazing specifics (e.g. casement type, glass thickness), hazardous materials locations, and protruding architectural details for every multi-story structure. Data selection must be based on perceived or established needs. Once data is selected, it must then be collected.

4.1.1.2 Collection

Data collection encompasses establishing requirements, committing resources (staff, hardware, and software), and developing a project plan (Figure 5). Critical to the feasibility of the establishment of an IMIS system is the incorporation of current data and existing systems to develop the baseline system using the following: GIS, a relational database, photographs, and CAD drawings. There must be recognition that many communities already have a wide variety of disparate local data repositories. A new framework must accommodate and integrate the unique characteristics of existing local archiving and address the non-digitized nature of most available data, which is especially relevant for older buildings and smaller communities. Historical data also tends to be highly disparate and extremely decentralized, as it commonly exists in a paper format in city planning departments, older design firms, local libraries, colleges, and historical societies. The dispersed character of the data presents logistical complications for cost-effective collection.

Even data currently being regularly collected is typically warehoused in many physical locations, stored in incompatible formats, and commonly inaccessible to other agencies (Zerger 2003). The existence of relevant data possessed by one agency may even be unknown to another, even though the data may be directly pertinent to the mission of both agencies. To create and implement an effective GIS-based disaster management system, such technical and administrative barriers must be eliminated. The extensive challenges related to data collection include identification, acquisition, verification, and storage.

4.1.1.2.1 Identification

The first step is to evaluate the wide variety of current resources, both manual and digital. Determining the location and extent of both historical and current information is a substantial topic, which requires both the identification of existing resources and the creation of new ones (Laefer et. al. 2005a).

4.1.1.2.2 Acquisition

Gathering all necessary baseline data represents a formidable challenge. Required steps include physical retrieval and logging of what has been obtained, from whom, from where, and what remains to be acquired, as well as a detailed listing of all locations where data has been sought and/or obtained to prevent duplication of effort.

4.1.1.2.3 Verification

All information gathered must be checked for internal consistency and against any other information with which it may be in conflict. As an example, not only must utility and transportation networks be continuous, but they must be physically connected to other elements of the built environment for them to be accurate.

4.1.1.2.4 Storage

At a minimum, to adequately represent a building, architectural and structural drawings and photographs should be collected and stored for each story of each building side, as well as glazing details; a two-story building was selected as typical. A rough estimate of computer storage is approximately 400GB (Table 3) for the above-mentioned items for a municipal area of 100,000 people that incorporates approximately two buildings per three individuals (residential, commercial, and public sector structures).

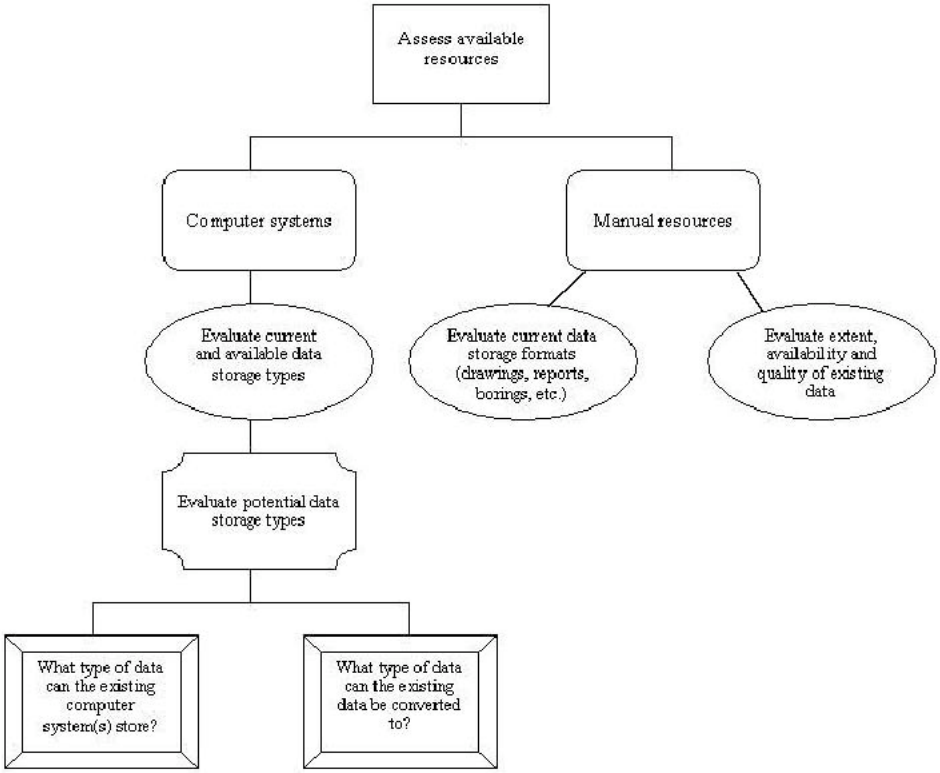


Figure 5 Data Collection Plan

Table 3 Storage Requirements

Items	Buildings	Stories	Images per story	Other	Memory per item (kb)	Total memory (GB)
Photos	60,000	2	4	windows	200	103.0
Architecture drawings	60,000	2	4	windows & plan	200	114.5
Structural drawings	60,000	2	2	floor & section	200	114.5
Glazing drawings	60,000	2	1		200	46.0
					Total (GB)	378.0

An in-depth understanding of images must be established to designate an appropriate subset of data so that content can readily be archived in a manner that is meaningful and available for retrieval, display,

and analysis. A robust and efficient database system is required to store, retrieve, and analyze excerpts from a potentially enormous multi-media data set. Current GIS data storage file systems (e.g. ArcInfo Coverage format, ArcView Shapefile format) do not allow storing multimedia data sets. Recent advances made in relational databases allow efficient storing and retrieving huge multimedia data sets, but analysis of such data is not yet possible.

4.1.1.3 Digitization

As a prerequisite for retrieval and analysis, meaningful storage of graphical data requires complete digitization. Digitization is the process of converting information from analog to digital information (Smith 2001). In an IMIS system, inspection reports and photographs may need to be digitized, as well as the information pertaining to the road networks, utility lines, building footprints, structural and architectural drawings. Prior to digitization, guidelines need to be developed to achieve necessary data accuracy, consistency, quality, and storage size. Without guidelines, it is difficult for large-scale digitization to meet long-term project goals (UNC-CH 2003). Guideline adoption is standard procedure for libraries prior to digitization of collections (Hazen 1998). As an example, the adopted library digitization guidelines from the University of North Carolina at Chapel Hill propose that images be stored in a Tag Image File Format (TIFF), a type of raster data format.

Digitized data are typically stored in either vector or raster data formats. Vector data is comprised of lines or arcs defined by beginning and end points (Longley 2001). Examples are land parcels, county boundaries, road networks, and utility lines, which are represented as a series of connecting arcs. In raster data, each area is divided into rows and columns, which form a regular grid structure (AGI 2003). Each cell within this grid contains an attribute value, as well as location co-ordinates. The spatial location of each cell is implicitly contained within the ordering of the matrix, unlike a vector structure, which stores topology explicitly. The surface of an area is represented in raster format, where each cell represents the elevation of the area. Output format selection as vector or raster depends on feature types and usage requirements. Geometrical transformations and visualization can be performed faster in a vector format than in raster one, while operations requiring information concerning surface coverage can be performed more easily in a raster format. Roads, utility lines, and building footprints are normally digitized to vector formats, while raster formats are commonly used for elevation data.

4.1.2 Integration

After depiction, the various data sets must be integrated into a single repository. Presently, such integration is rare, even for communities that have already committed extensive resources to related activities. For example, New York City (NYC) has a highly detailed GIS-based base map but has had no plans to: (1) link GIS features to a database of site relevant properties, (2) use the post-incident building inspection reports and documentation (as well as owner-based repair records) related to the September 11 attacks to create detailed, location-based records of changes in baseline condition or (3) compare the state of pre-existing conditions to that of buildings in post-disaster scenarios (Kevany 2003, Dorf 2002).

An integrated database would help to facilitate rapid and detailed disaster response scenarios that could not previously be considered with traditional tools, because the database would allow unprecedented use of multiple data types (attribute data, text, drawings, maps, and photos) in combination with electronic spatial data fusion to link together otherwise inaccessible data. The processes result in new ways to share and distribute data. A single integrated repository system requires standardization of two important components: data formats and data access mechanisms. To enable multi-organizational data access and sharing during a disaster, standardization is needed for the formats and access mechanisms to be compatible.

4.1.2.1 Data Format Standardization

A data format is defined as a specification that defines the order in which data is stored or a description of the way data is held in a file or record (AGI 2003). The selected formats must support community-developed disaster management programs and must be sharable across, federal, state, and local jurisdictions, as there are many elements of the infrastructure that are not wholly controlled by the local building authorities. These range from the obvious (e.g. interstate highways), to high profile targets such as federal courthouses. The problem is difficult to surmount and is exacerbated by decades of non-coordinated data generation. To prepare for disasters, the scope and format of pre-incident data collection efforts need to be both identified and mandated in order to facilitate data incorporation into an IMIS (Laefer et. al. 2005a).

Standardized data formats across multiple organizations, promote cost-effective data sharing. In order to achieve this, standards are needed that allow users to find information and processing tools when and where needed; to understand and employ information and tools independent of platform and location, and help expand GIS capabilities without dominance by a single commercial vendor (Carroll 2003). Such standards are called open standards and they promote data sharing and integration from various sources. Contemporary GIS data formats are inadequate, because both spatial data and affiliated attribute information are stored in a variety of proprietary file formats (e.g. ESRI shapefile, ESRI coverage, AutoCAD dxf). Proprietary file formats hinder integration as they are not based on open standards. Industry consortiums such as the Open GIS Consortium (OGC) have formulated the "simple features specification" (OGC 2003), and the United States Geological Survey has established the "Spatial Data Transfer Standard" specification explicitly for standardized data exchange (SDTS 1997).

Over the past several years, international geospatial standards for GIS have been pioneered to reduce work duplication, optimize resource expenditure, and facilitate information exchange (Clément 2000). Even, the Federal Geographic Data committee has identified as a part of national security concerns (i) development of guidelines and standards for a comprehensive spatial data infrastructure, (ii) interoperability of the systems that process this information, and (iii) guideline establishment for commonality of the processes that collect, manage, and disseminate geospatial information (FGDC 2002).

The common standards help fulfill a key aspect of an enterprise GIS system that of non-proprietary data storage formatting. This, however, only aids data storage. To make the data available to user or client applications, data access mechanisms must also be standardized.

4.1.2.2 Data Access Mechanism Standardization

Standardized data access mechanisms aid the data retrieval process. Structured Query Language (SQL), established by American National Standards Institute, is a comprehensive database language that provides statements for data definition, querying, and updating (Elmasri 2000). Most relational database management systems support SQL as a common standard to facilitate data querying and manipulation, but relation database management systems do not support spatial data. For a CDF to succeed, support for spatial data types is mandatory. OGC's "Simple Features Specification for SQL" defines a standard SQL schema to handle spatial attributes. Use of both standards for data formatting and manipulation are vital to the success of a cogent CDF to ensure maximum data exchange using minimum resource allocation.

4.1.3 Connection

The third major requirement for a CDF is connection, which affiliates spatial features (roads, utility lines, and buildings) with attribute data of varying sizes, shapes, and materials (Figure 6). Attribute data constitutes the characteristic information about the spatial features. For instance, a road, which is represented by a line feature, has various attributes such as width, bedding configuration, pavement

condition, and speed limit. Likewise, a building, which is represented as a spatial feature, has attributes such as height, street address, and number of stories. By establishing connections between each spatial feature and relevant attribute information, the existing system can be transformed into a substantially more powerful tool than a simple digital map, because of the capability for spatial querying based on the attribute data. For instance, since glazing comprises 90% of all blast-related injuries from an explosion (Mays 1995), window information including glass type, glazing thickness, casement type, damage percentage, and breakage patterns (in the form of photographic images) are requisite for meaningful post-disaster analysis and pre-disaster planning.

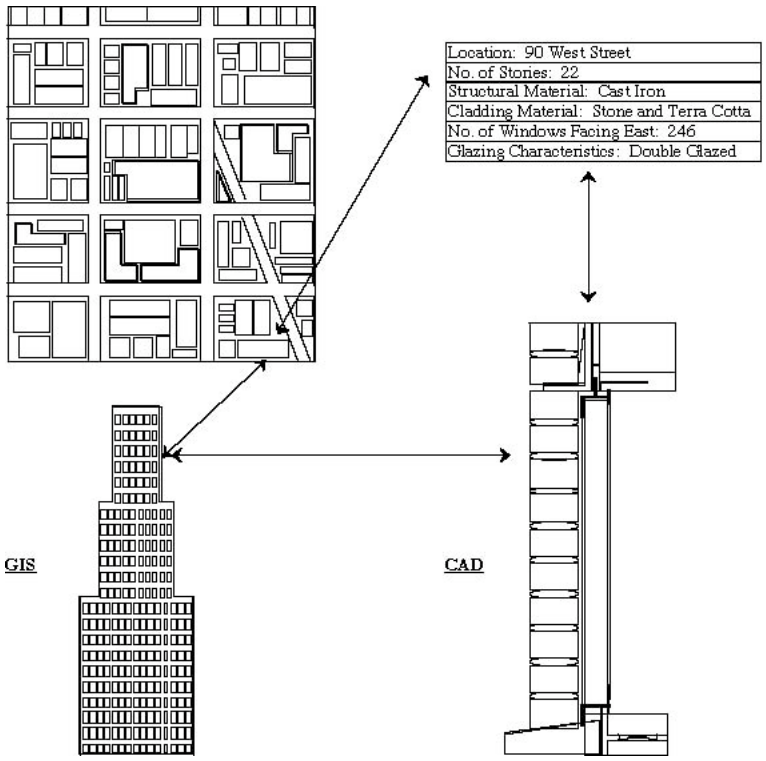


Figure 6 Graphical and Tabular Information Representations

For blast analysis, total building attribute information such as physical location, number of stories, structural support system, and cladding information must be also housed in the database (Figure 6). Additionally information such as story height, wall thickness, percentage damage, and distance from an explosion are needed and must be systematically stored. Yet, a building is comprised of multiple facades, floors, and windows, and attribute data may need to be simultaneously affiliated with multiple elements. To create multiple, simultaneous connections between spatial features and attribute data, a robust and effective data model is required.

4.1.3.1 Object-relational Data Model

A computer data model is a set of constructs that describe and represent selected aspects of the real world (Longley 2001). There are many existing GIS data models to store geographic data. Current GIS data models include CAD, raster/grid, triangulated irregular network, vector/geo-relational topologic, and network. All of these models are geometric-centric and representing real world object using either a polygon, line, or point. For instance, a building is usually represented as a polygon feature, a road as a line feature, and a utility pole as a point feature. Unfortunately, these representations are not sophisticated enough to represent complex aspects of geographical features.

This complexity needs to be captured by the data model for successful analytical GIS operations. Concurrently such operations require systematic and easily retrieved stored data. Distinct attribute information needs to be stored for the building's individual façades, but a polygon feature does not allow storing the attribute information of an individual façade (Figure 7). Thus, a better representation of a façade is as an individual line feature, where its attribute information can be represented as the attribute data of that line feature (e.g. façade orientation, cladding material, number of windows, etc). Unfortunately, unlike a polygon feature, a line feature representation does not by itself allow complex relations to be established, such as finding the area enclosed by a group of lines that represent the perimeter of a building (an operation supported by a polygon feature). Thus, if a building is treated as a group of line features, such representation requires the development of customized and complicated applications (programs or scripts) to enable the execution of typical spatial operations related to polygon features.

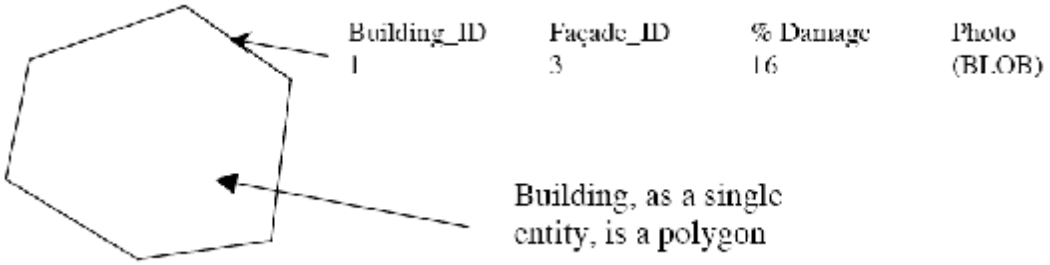


Figure 7 ORDM Building Representations

In order to model such complexity, an object-relational data model (ORDM) is required. An ORDM possesses the capabilities of both object-oriented and traditional relational models (Darwin 2000). With an object-oriented data model, the real world entities are captured as objects, where the data model collectively captures the associated data and the methods to manipulate that object. In contrast, in a relational data model, only the attribute data is captured without the methods to manipulate the entity. In an object-oriented data model, the building's attributes data (height, address) and methods to manipulate the information (calculating the area of a building) are both encoded in the data model.

The objective of an ORDM is to collect geographic objects and define the relationships between them to allow representation of a mini-world entity (regardless of complexity and structure) in a database by a single object. Since a geographic entity can be composed of multiple sub-entities, an ORDM allows designing and implementing a geographic entity as composed of sub-entities (Dittrich 1988). For instance, a building, which is represented as a single entity, is composed of facades as sub-entities. An ORDM allows the entry of attribute information for individual façades and considers the building as a single entity without the need for sophisticated and customized programs to handle such relationships. Such required sophistication is inherent to the data model, thus eliminating the need to develop customized applications.

A conceptual view of the data model for a prototype IMIS (represented using an entity relation model) is shown in Figure 8. An entity relation technique is used as a de facto standard for representing a conceptual view of the data model (Silberschatz 2001). The data model presented is the conceptual model of the post-explosion building attributes required for blast wave analysis.

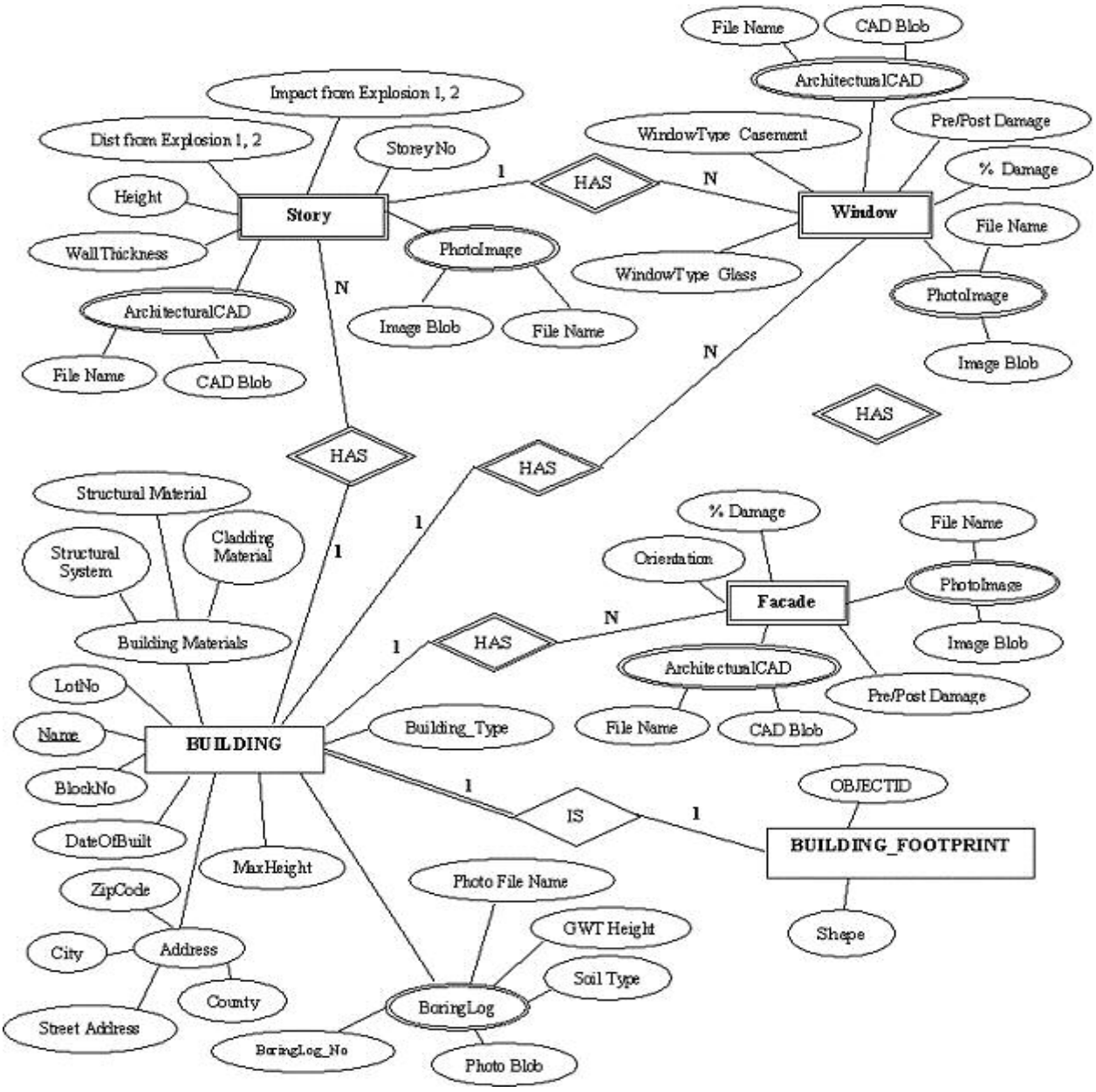


Figure 8 ER Diagram for a prototype IMIS

4.1.4 Evolution

The fourth requirement for a CDF is the capacity for evolution. As communities grow, age, and experience disasters, the configurations and conditions of the infrastructure changes. To address the changeable nature of the data, the CDF data model should be extensible to accommodate future data fields, and the database system should be able to time stamp changes. To facilitate evolution, both an abstract data model and a temporal database are essential.

4.2 Abstract Data Model

An exhaustive list of all attributes of a data model for current and future usage is difficult to generate. Thus, the data model must be sufficiently flexible to accommodate changes. As additional data attributes are defined over time, the data model must be easily expandable. New window types may or may not

share all attribute information of previously defined windows. As an example, if there is a bomb blast, the windows before and after the blast share common attributes such as width, height and location. They do not, however, share other attributes such as percent of damage (Figure 9). A well-abstracted data model defines the universal set of properties that are true for a broad class of situations. As the data model requires capturing specific classes, sub-classes are extended from the abstract model (e.g. pre-blast window and post-blast window) (Liskov 2000). The extended sub-classes inherit all the base attributes of the parent class (Figure 9). This is known as inheritance in object-oriented modeling. An object-oriented data model provides such abstraction, which is invaluable in the design of a flexible data model to accommodate future and unforeseen needs.

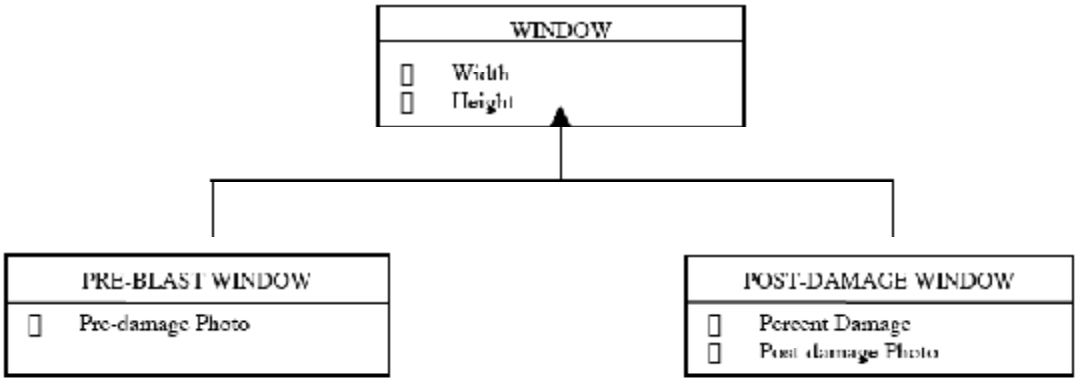


Figure 9 Inheritance

4.3 Temporal Database

A major goal of the IMIS is complete integration with various agencies and entities to accommodate future updates or changes (e.g. architectural and structural renovation of a building). Normally, such changes override the previous versions of the data. In an IMIS, however, maintaining previous versions is imperative to understand the temporal evolution of the data. The system should be able to reflect the culmination of a community's changes over time and be viewable at previous points in its evolution. This capacity requires time stamping of the data. Spatial features and attribute information that mutate in both disaster and non-disaster environments must have the capability to be updated. Incorporating temporal evolution mandates time stamping all data and maintaining it as queriable information. To achieve this goal, two data time stamps are required: (i) the time when a user enters the new or updated data into the database and (ii) the time when the data was created (i.e. a disaster may occur on May 15, but some of the data may not be entered until three months later). The temporal database should be queriable by date of entry, date of creation/mutation, a queuing mechanism that defaults to retrieve the most recent entry, and one sufficiently flexible to query a user defined point in time. As an example of the applicability of a CDF to multiple disasters, a hurricane scenario is be presented as the application.

4.4 DIORAMA Design Outline

An ideal disaster management system should be able to assist during the six critical disaster related phases as a one integrated system (Figure 1). The efficacy and establishment of such a system would help to minimize the loss of life and property, as well as provide economic recovery. The following is an outline of the design and implementation of the key components to such a system as exemplified by DIORAMA: a new architecture, multi-client user scenarios, data integration, and extreme data processing environments (all of which are vital for the success of disaster management). Any DMS system is

accessed by hundreds of users depending on their intended use. Ordinary civilians access the DMS for information related to safe routes, available disaster-relief facilities, while emergency managers use the same system for proper and timely reallocation of the resources such as food, drinking water, medicine, etc. Similarly, the baseline system requires a various data (map of roads, pipelines, rivers) from multiple sources, these data need to be integrated as a single integrated repository system. As the data is analyzed by many users for different purposes simultaneously, the system should have the capability to address reliable and efficient data processing and analysis capabilities. Thus, in order these issues, the system architecture for a future disaster management system needs to rely on an enterprise-level GIS framework. The concept of enterprise GIS is to maximize the utilization of GIS capabilities by channeling timely spatial and non-spatial data from various sources within a government/organizational setup.

5. ENTERPRISE GIS ARCHITECTURE

Current disaster management systems are standalone and, thus, do not harness the power of an enterprise GIS framework. These systems are designed either as stovepipe or client-server architecture. In stove-piped applications, the three logical components of GIS: {data, business logic, and user interface (also known as client)} are tightly coupled and distributed as a single entity (Figure 10a). The data component contains the organization's data, while the business logic implements the functionality or business rules of an application. The user interface component interacts with the business logic component to access the data component. In contrast, in client-server architecture, the data management and the user interface responsibilities are distinct, with no defined place for operations and business logic (Figure 10b). A profound departure from these arrangements is a three-tiered with the capability of being an N-tiered design (Figure 10c), where there is a loose coupling and separation of at least three logical components, which is typical of an enterprise framework.

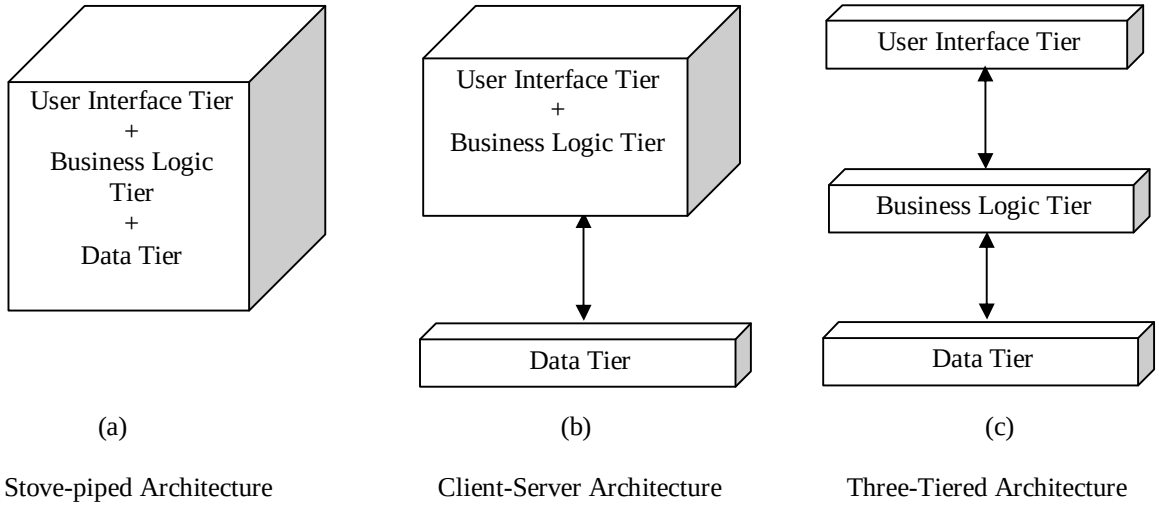


Figure 10 Types of Generic System Architecture

The three-tiered approach, commonly referred to as N-tiered architecture, is an allusion to the unlimited number, N, of intermediary layers between the client and server, which reduces redundancies in both data entry and processing. In N-tiered architecture, the operations and analysis components are explicitly restricted from being coupled to the user interface and data components (Morais 2000). The introduction of the business logic tier enhances performance, flexibility, and maintainability by centralizing process logic. With other architectural designs, a change to a function (service) would need to be written into every application (Eckerson 1995). N-tiered architecture circumvents this requirement, thereby promoting a substantially more efficient and flexible system. The NCSU IMIS system architecture has Oracle 9i database servers as a data tier, application servers (Apache HTTP server, Esri ArcSDE™, Esri ArcIMS™) as a business logic tier, and client applications (customized client application, Esri GIS applications, and Intranet/Internet clients) as a user interface tier (Figure 11).

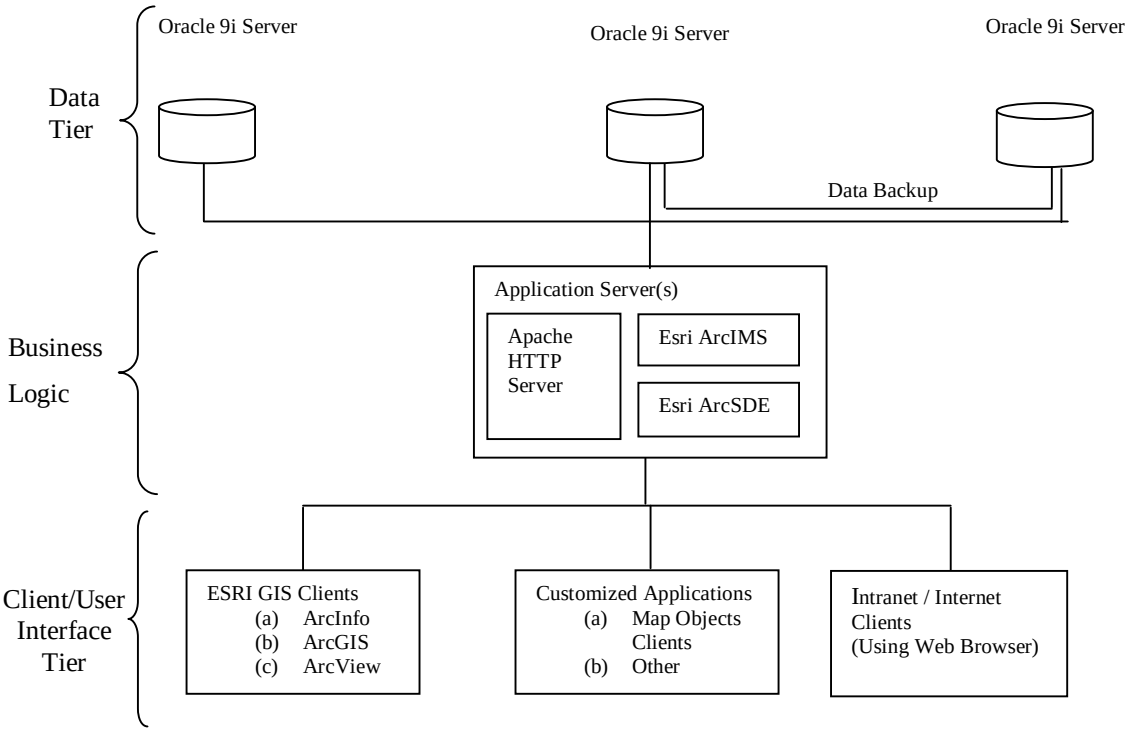


Figure 11 NCSU 3-Tiered System Architecture Design

5.1 Backend Database System (Data Tier)

A data tier in a three-tiered architecture is represented by a backend database system (Figure 11). The primary function of a backend database system is to store data. Depending upon the number of users who access the data and the complexity of the data, different database systems are used. Small and simple data, commonly used by a small number of people, can be stored in standard data files, while large, complex data with many users require special database management system (DBMS) to ensure its integrity and longevity (Longley 2001). As IMIS data is accessed by hundreds of users spanning across different organizations and geographic locations, and the data is complex in nature, the IMIS data tier needs to implement a special database management system. The subsequent sections address various issues such as data format, data retrieval mechanisms, data integrity that are the essential features of the special database management system for the success of IMIS system.

5.1.1 Importance of a Standard Data Format and Retrieval Mechanism

For a disaster response and planning system, a contemporary desktop GIS system is inadequate, because spatial data and its attribute information are stored in a proprietary file format. The proprietary file format hinders integration within an enterprise framework as it is not based on open standards, such as those recently developed by both governmental and non-governmental organizations. As examples of standardization, industry consortiums such as Open GIS Consortium (OGC) have formulated the “simple features specification” (OGC 2003), while the United States Geological Survey (USGS) has established the “Spatial Data Transfer Standard” specification for standardized data exchange (SDTS 1997). These common standards help to fulfill a key aspect of an enterprise GIS system that of non-proprietary data storage formatting. This, however, only aids the process of storing data. To make the data available to user or client applications, data access mechanisms must also be standardized. Structured Query Language (SQL), established by American National Standards Institute (ANSI), is a comprehensive

database language that provides statements for data definition, querying, and updating (Elmasri and Navathe 2000). Most relational database management system (RDBMS) supports SQL as a common standard to facilitate data querying and manipulation. OGC's "Simple Features Specification for SQL" advances the state of the art by defining a standard SQL schema to handle spatial attributes, which does not exist in the present version of ANSI SQL. Use of both standards for data formatting and manipulation are vital to the success of an enterprise GIS system to ensure maximum data exchange with ease.

5.1.2 Importance of an Integrated Repository System

One of the major problems faced during the New York City (NYC) rescue operations after September 11 was the lack of availability of data sets in an integrated repository system. Data sets from different organizations first had to be collected, prior to any computerized-based search and rescue operations. The enormity of the situation highlighted the acute and immediate need for accurate and compatible spatial information. During a disaster, decision makers, incident commanders, emergency responders, and city mangers need spatial information instantaneously. Since timing is critical during search and rescue operations, all potentially required data sets must be pre-stored in a centralized repository system. This ensures that mission-critical data can be retrieved and redistributed to different agencies or personnel for analysis in a timely manner.

The NYC problem was exacerbated by the fact that although there had been a significant investment in the establishment of an Emergency Mapping and Data Center (EMDC), the center was located at WTC and was entirely destroyed during the attack. The EMDC had to be completely reestablished off-site to aid the search and rescue operations. What should be understood from this example is that an integrated repository system must not be not confused with a repository system having a single point of failure. The system's actual implementation should comprise multiple repository hardware systems that are physically distributed across many locales. From the user's perspective, however, all the required data sets are visible, as if it was a single physical repository, and any updates made on one data set are immediately reflected throughout the distributed system. A distributed database system known as a integrated repository system precludes the loss of data access or data loss, but presents other challenges.

5.1.3 Challenges of an Integrated Repository System

Creating a integrated repository system demands (a) multi-version concurrency control, (b) recovery capabilities, (c) system security, and (d) support of multi-format data. Concurrency control is the activity of synchronizing operations issued by concurrently executing programs on a shared database. The goal is to produce an execution that has the same effect as a serial one. In a multi-version database system, each write on a data item produces a new copy (or version) of that data item (Bernstein and Goodman 1983). Since multiple users access and update the same data simultaneously, this concurrency control property is extremely important to ensure data integrity. If user A and user B are accessing and modifying the same data simultaneously, the concurrency control mechanism must ensure that each user receives the most recent copy of data and that the updates made by each user are properly stored in a sequential manner.

A recovery mechanism is also crucial to ensure the atomicity property of a transaction, since failure can occur whenever a network connection to the database severs or during the editing or updating of data. If a transaction fails before completing its execution, the recovery mechanism must ensure that the transaction has no lasting negative effects on the database (Elmasri and Navathe 2000); the system must be capable of restoring the last whole data entry.

The third important component of a integrated repository is security, which is crucial since multiple users will be accessing the same repository system. The system must protect the database against persons who are not authorized for partial or total access. These properties are not inherently supported by the file-based GIS data sets typically comprised of only a set of simple files in which the stored information is

often separated by a delimiter and cannot, therefore, support mechanisms for concurrency, recovery, and security.

Another major challenge of a integrated repository system is the hosting of multimedia data formats: images, video clips, and drawings collected both before and after a disaster incident. All of the information needs to be integrated and stored for analysis. The repository system must act, thus, as a robust multimedia database management system.

5.2 Middleware System (Business Logic Tier)

The next essential portion of a disaster based IMIS is the middleware, also known as business logic tier. Middleware is defined as a set of common services that enable applications and end users to exchange information across networks (Umar 1997). These services reside in the middle portion of the architecture, above the operating system (OS) and networking software and below the distributed applications (Figure 12) (Bernstein 1996).

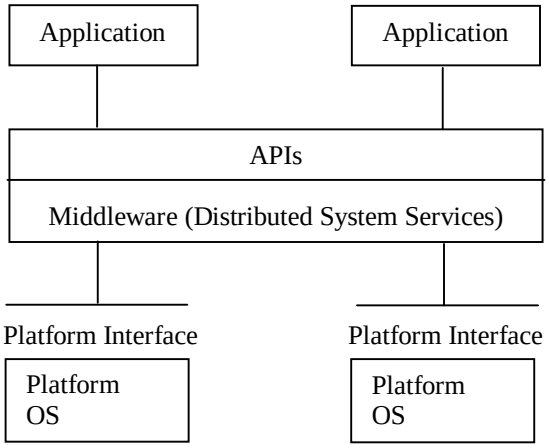


Figure 12 Middleware (adapted from Bernstein 1996)

Bernstein (1996) identifies how middleware services are distinctive from and more advantageous than applications or low-level services in the following aspects:

- Operates on different OS and network services and, thus are not tied to a specific platform such as computing hardware, communication networks, operating systems, or other software components.
- Serves multiple purposes at a time, since they are available to multiple applications and users, while not being embedded in a single application for a specific industry.
- Are distributed, which permits remote accessibility of them or their applications.
- Supports standard interfaces and protocols and thus, are specified by well-defined "Application Programming Interfaces" (APIs) and communication protocols. Such interfaces facilitate porting applications to different platforms, and standard communication protocols allow applications on systems with different machine architectures and operating systems to exchange data.

Middleware technologies are generally categorized as either general or service-specific. A general middleware technology provides a common framework that helps build a middleware component, irrespective of its intended usage, while service specific middleware technology can be customized to facilitate a particular task. For instance, current middleware technologies such as Java 2 Enterprise

Edition (J2EE) and Microsoft .NET have provided common frameworks to develop business logic while circumventing the need to know about the technical details about the underlying hardware, operating systems, and communication protocols. These frameworks are not tied to any service specific technology and are as such known as general middleware technologies. A service specific middleware technology is designed to serve a specific domain. For example, contemporary RDBMS cannot handle the spatial data in an efficient and intelligent manner due to issues of topology and geometry, whereas a spatial-database-specific middleware system may be designed explicitly for that purpose. Many commercial spatial-database-specific middleware products are available, but the following criteria must be met for their use in a full functional disaster management system:

- Provide the required infrastructure to manage multiple users.
- Provide business logic software to support advanced GIS data types (e.g. images, networks, features with integrated topology and shared geometry) and associate these various data types with rules, behaviors, and other object properties.
- Allow GIS data to be directly maintained in the format of "spatial types" supported by the Database Management Systems (DBMS) vendors.
- Integrate spatial (geometric) search capabilities provided by the DBMS vendors within the GIS client software applications.

In general, disaster management systems require a number of middleware systems that are both general and service specific. The efficiency of the entire system is highly dependent on the efficacy and proper use of the middleware system. Such middleware-intensive arrangements foster the development of client applications with minimal functionalities. To perform such spatial queries, which make use of huge network data sets, can be resource-intensive tasks for the middleware system, as well. This arrangement mandates that the disaster management system exploit the power of processing advancements such as parallel or distributed computing.

5.3 Parallel and Distributed Computing

Beyond the architecture, the manner in which data is processed impacts the feasibility of a disaster response system. Modern computer simulations of complex natural phenomena, such as forest fire growth or wave propagation due to blast explosions, require extensive computer capabilities for both processing and storage. Limitations of a single processor can be severely felt while doing such simulations. Both physical and practical reasons pose significant constraints to the obvious solution of simply building ever faster serial computers. The constraints are (a) transmission speeds - the speed of a serial computer directly controls how fast data can move through the hardware, (b) limits to miniaturization – although ever improving, processor technology currently allows only a finite number of transistors to be placed on a chip, and (c) economic limitations - it is increasingly expensive to make a single processor faster. Absolute limits are the speed of light (30 cm/nanosecond) and the transmission limit of copper wire (9 cm/nanosecond). Increasing speeds necessitate increasing proximity of processing elements.

A solution is the simultaneous use of multiple computing resources, also known as parallel computing. There are two primary motivations for using parallel computing: to save time and to solve larger problems. The use of parallel computing mandates significant improvements be made in both the hardware and software aspects of the computing. The parallel architecture should be comprised of multiple numbers of interconnected processors to solve a common problem. The software should employ parallel algorithms that explicitly take advantage of a parallel architecture. A parallel algorithm is any algorithm in which computation sequences are performed simultaneously, while a serial algorithm is any algorithm in which computation sequences are performed sequentially.

In disaster management system, a wave propagation model linked to a GIS-based system could be extremely valuable for pre-disaster planning, crisis management, and post-disaster recovery, but the success of such an effort would be highly reliant on access to parallel processing capabilities (both hardware and software aspects) to achieve adequate processing capacity, within a reasonable time frame. During a crisis, a system that employs parallel computing would allow real-time communication between engineers, civil protection agencies, and emergency services, thereby rapidly informing them of changing hazard conditions. So far, only limited research has been conducted to exploit the power of parallel computing to solve this complex class of problems. The field of parallel processing is still considered novel and lacking uniform support in academic and commercial circles (Mineter et. al 2000). Commercially available GIS software does not presently utilize multi-processors for analysis, thus relying on continuing advancements of the single processor computer to meet evolving needs. The major hindrances towards using parallel computing include the following:

- Complex Parallel algorithms implementation and
- Hardware costs.

5.3.1 Parallel Algorithm Implementation

Parallel database management and computational geometry make GIS operations in a parallel environment an advanced topic, one beyond pedestrian GIS usage. Mineter and Dowers (1999) conclude, "although parallel computing can be considered to be maturing technology, its exploitation in geoprocessing is still its infancy". The complexities of implementing a parallel algorithm and the lack of parallel libraries specific to this field have limited progress. Mineter and Dowers (1999) propose creation of a parallel software framework, one that encapsulates the complexity of the implementation of parallel algorithms and enables the reuse of existing parallel libraries in different applications. Such a framework will foster the use of parallel algorithms to solve problems that require heavy computation, without users having to achieve a mastery of complexities associated with the design and implementation of such algorithms.

5.3.2 Hardware Costs

Although parallel computing harbingers a powerful and optimistic future for disaster management systems, the cost of the hardware platforms required for parallel computing remains a major obstacle. Platforms, such as SGI/Cray Origin and IBM SP demand a heavy investment in capital and maintenance. Since a reliable and robust disaster management system should be affordable to every community, irrespective of size, such computing requirements must be met in a cost efficient manner. With recent developments in distributed computing on clusters of workstations, such as Beowulf clusters (Beowulf 2000), localized, high-performance parallel computing capabilities can be built at an affordable price. Such clusters are constructed primarily out of commodity hardware components, running a free-software operating system (e.g. Linux or FreeBSD), and interconnected by a private high-speed network. The use of clusters to prototype, debug, and run parallel applications is becoming an increasingly popular alternative to using specialized, high-investment, parallel computing platforms (Buyya 1999). With such an approach, the system cost can be dramatically reduced with little performance degradation. Cost is, thus, a motivating force for cluster computing.

5.4 Customized Client Applications

Apart from the numerous benefits of middleware system (e.g. support for multi-user access and standard protocols) , the middleware fosters customized client applications in the system which is vital to the success for a disaster management system. Current GIS client applications used in disaster management systems are designed for the concept of client/server architecture and, hence, can be termed as "fat clients". In a client/server environment, a client that conducts most or all of the processing leaves little, if

any, to be done by the server. Relying on a client to process the data, which is highly resource intensive in terms of cost and computing power, is generally an uneconomical solution when there is a need to service hundreds of clients, as would be the case for a disaster management response system. Thus, the concept of a customized client in which different client applications are designed and implemented as per the required functionalities is preferable to "fat clients".

One of the greatest strengths of the N-tier architecture is its support for customized client applications. Since the business logic to manipulate the data is provided by the middleware system, customized client applications can simply access any preexisting business logic that is needed. This characteristic helps to develop client applications with minimal business logic (that is not already provided by middleware system), thus reducing development time and cost. For instance, the middleware system may provide a service (business rule/logic for disaster management system) to find the shortest route to the nearest hospital from a given point of geographical location. To accomplish this example, the system must identify hospital structures, locate the point of initiation, and conduct a spatial analysis to optimize routing options. A GPS supported, handheld device with limited computing power carried by ambulance personnel or a powerful personal computer carried by other rescue personnel can use the same service irrespective of its computing power, because the logic to process the complex spatial query resides in the middleware system and not in the handheld device or personal computer. Such middleware-intensive arrangements foster the development of client applications with minimal functionalities.

6. NC STATE DISASTER MANAGEMENT SYSTEM (DIORAMA)

The infrastructure management information system DIORAMA was designed as a multi-phase DMS. Prototype development occurred at the rate of approximately 10 hours per week over 18 months with the following objectives:

- Look at a single aspect of one phase of a disaster, specifically the data entry and retrieval necessary for the categorization) of blast damage from the 2001 WTC terrorist attacks.
- Develop a GIS-based map of the portion of Manhattan that could support the inclusion of the over 300 blast-damaged buildings.
- Build a database sufficiently detailed to collect and categorize the necessary data to conduct blast damage categorization.
- Spatially query the system.

To these ends, the main objective technical was to build a system that separated the three critical tiers (i.e. data, business logic and client). Prototype development required knowledge of system design, GIS, database data modeling, and Java. DIORAMA's database system, middleware, customized client interface, and general functionality are described below.

6.1 Database Server (Data Tier)

DIORAMA's system architecture utilizes Oracle 9i database server for its data tier, which was chosen for its capacities to both serve as a central repository for spatial data and support integration of spatial data with other core organizational data. The server can supports multiple users, exploits enhanced database management features (e.g. administration and maintenance utilities, replication, and fast backup and recovery), and employs Structured Query Language (SQL) – an open application, programming interface; SQL is a comprehensive database language that standardizes data definition, querying, and updating tasks, as established by the American National Institute (Elmasri and Navathe 2000).

Such a database has broad storage capacity (e.g. vector data of building footprints, street centerlines, and other linear features in an Environmental Systems Research Institute (ESRI) geodatabase formats in use with ESRI's ArcSDE application server). A geodatabase physically stores spatial data inside a database management system, thus enabling spatial data to be kept in standard relational databases (MacDonald 2001). ArcSDE uses the default binary schema for an Oracle database server, which is fully compliant with the OGC Simple Features Specification for SQL's binary geometry (ESRI 2003). At the time of DIORAMA's development there were three database systems with adequate capabilities in common usage (IBM's DB2, Oracle's and Microsoft's SQL). The widespread availability of Oracle's database was the main reason for its selection.

A relational data model [a conceptual representation of the data structures required by the database (Adams et al. 2002)] was designed and implemented to store building attribute information. The attribute information was that, which was required to conduct post-blast building damage identification. Blast analysis in urban areas ultimately requires highly sophisticated, 3D capabilities to predict energy propagation and dissipation (Figure 3). Supporting this type of application is critical in a DMS and was primary in the prototyping DIORAMA.

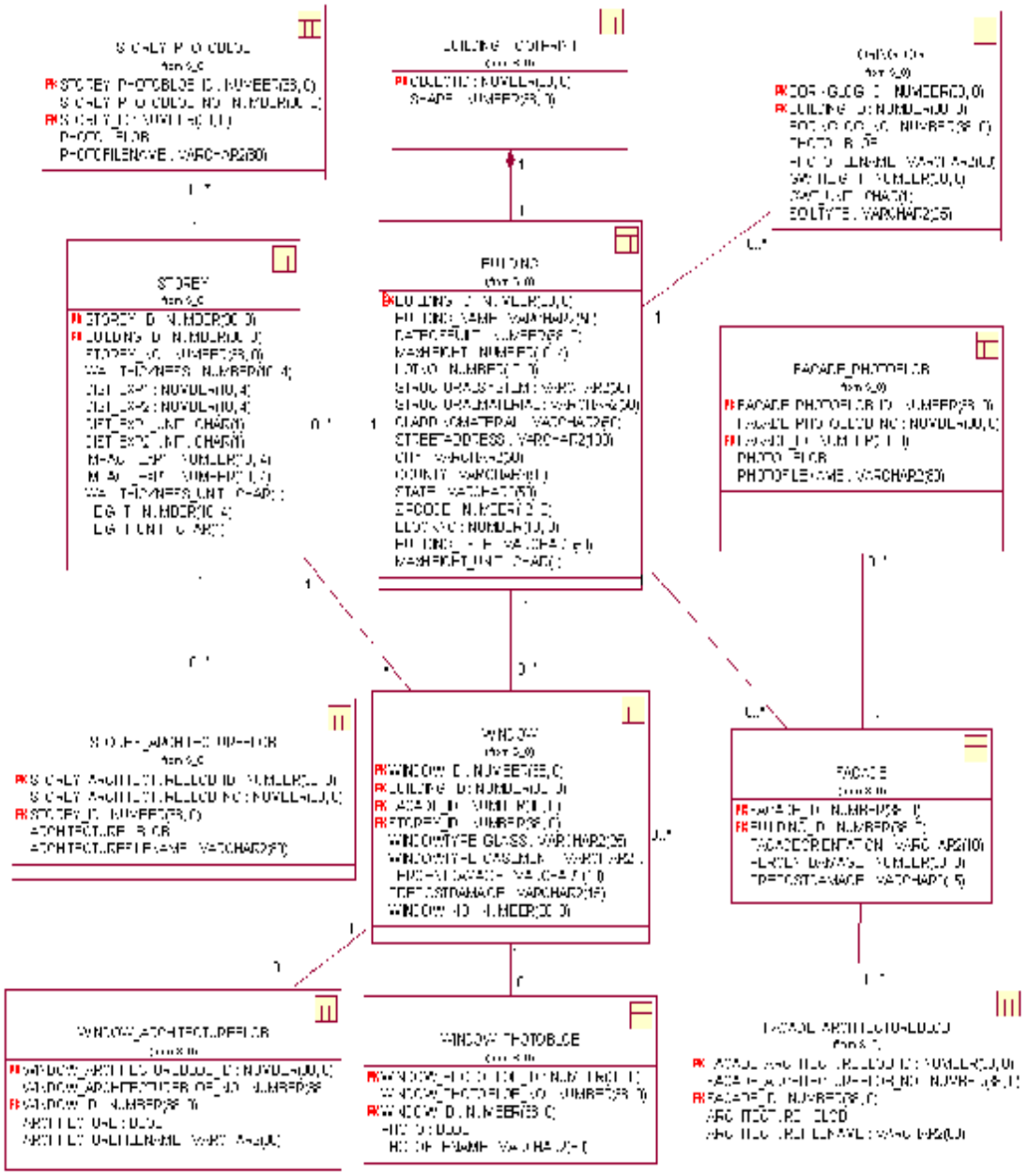


Figure 13 UML Conceptual Schema for DIORAMA's Database (Pradhan 2003)

Figure 13 shows a simplified conceptual data scheme of DIORAMA's database represented using the Unified Modeling Language; an industry standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems (Rumbaugh et al. 1999). Within the scheme are various entities (e.g. BUILDING entity, STORY entity, FACADE entity) that model a building as an entity consisting of a sequence of façades, stories and building components. The entity BUILDING captures the attribute information of a building (e.g. building name, street address, structural system type, date of construction), while BUILDING_FOOTPRINT stores its plan as spatial data (i.e. geometry and topology). The entities STORY, FACADE, and WINDOW capture attribute data of the building's story,

façade, and window, respectively using simple text and numerical data types. Computer aided design (CAD) drawings and images are stored as a binary large object (BLOB) data type (e.g. WINDOW_ARCHITECTUREBLOB, FACADE_PHOTOBLOB). Unfortunately, the BLOB data format cannot be read as meaningful spatial information without a spatial middleware system, as described below. Figure 13 illustrates a few attributes required to support one application – blast damage identification. As such, special emphasis is given to window and façade features. A complete enumeration of the functionality of this data model is provided elsewhere (Pradhan 2003).

6.2 Application Server (Business-logic Tier)

DIORAMA's middleware is comprised of ESRI ArcSDE server, an ESRI ArcIMS server, and a HTTP server from Apache Software Foundation for its business logic component (Figure 14). The ESRI ArcSDE server functions exclusively as a spatial database-specific middleware and enables a standard database management system to store and manage geographic data by adding a spatial data type to a relational data model. The server facilitates the storage of a geographic feature via a row of the database table and the coordinates that represent the geometry of the feature as a single binary object value in a column of the table (Miller and Shaw 2001). Additionally the server supports significant functions and capabilities, including the geometric data management required for complex topological analysis. The business logic to perform spatial analysis is also embedded in the ArcSDE server. Thus, the client applications need not provide the business logic, only invoke the required functionalities.

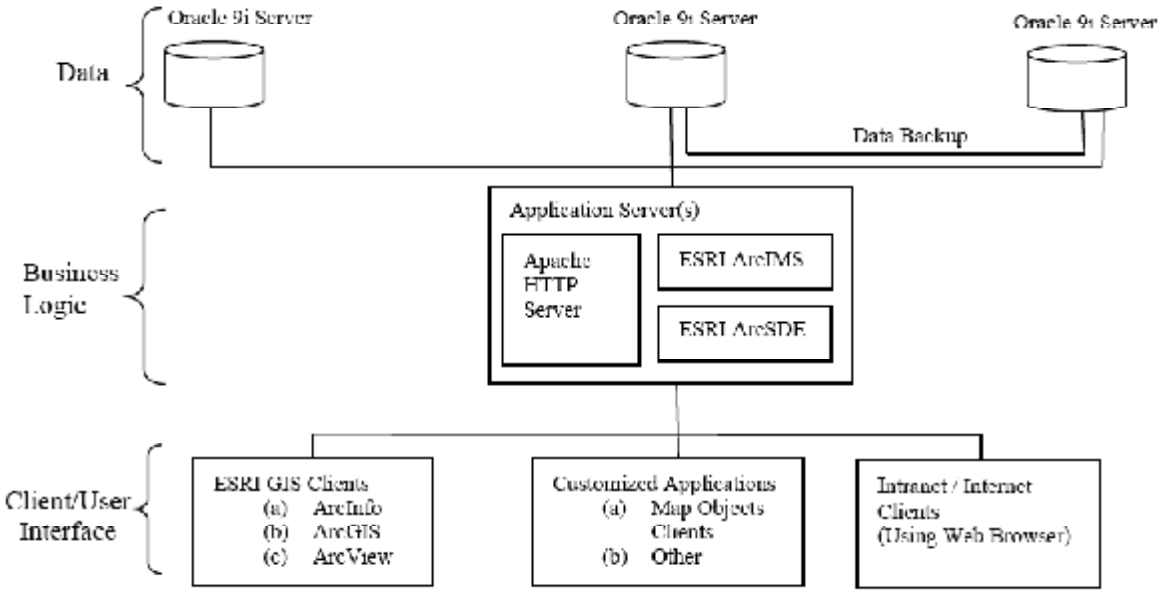


Figure 14 DIORAMA's Three Tiered System Architecture Design

ArcIMS provides the foundation for distributing high-end GIS and mapping services via the Internet (ARCIMS 2003). ArcIMS, with the Apache HTTP server, enables users to integrate local data sources with Internet data sources for display, query, and analysis in an Internet browser. Thus, they provide an infrastructure to web-enable DIORAMA. Unlike most GIS vendors, ESRI provides a wide variety of GIS software based on either two-tier to three-tier architectures. Additionally, at the time of DIORAMA's development none of ESRI's competitors offered GIS software that could simultaneously support three tiers and an Enterprise system. As the main focus of the prototype was the development of a three-tiered, Enterprise DMS, ESRI products were generally selected: DIORAMA relies on concepts implemented in

other Enterprise GIS, as the objective was not to create a new Enterprise GIS architecture, but to understand and prototype an existing GIS architecture to enhance DMSs.

6.2.1 Capturing Business Logic

"Use Case" modeling is the de facto technique for performing software requirements analysis, thereby capturing the business logic for a given system. Use Case modeling describes the system behavior of the target system from an external point of view (Fowler et al. 1999) by capturing the functional and behavioral requirements of the system that help the users perform their tasks. Use case diagrams contain actors (anyone or anything interacting with the business logic) and use cases [actions pertaining to business logic that benefit the initiating actor(s)]. Examining the actors and defining their capabilities with the system establishes how best Use Case deployment. Use Case diagrams are typically represented in Unified Modeling Language (Rumbaugh et al. 1999) to (1) capture system requirements and (2) communicate with end users and domain experts.

Since all system needs can rarely be covered in one Use Case, there is typically a collection of Use Cases. Figure 15 depicts the process for querying information from the database to the end user (database administrator or data entry personnel). The user logs into the system and verifies user credentials. As per pre-specified access privileges, the user queries information (spatial and attribute) stored in the database system. Results are displayed on the client application.

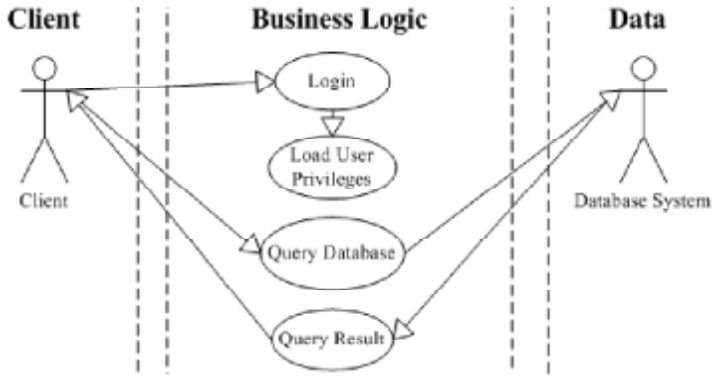


Figure 15 Use Case Modeling for Querying Database

After Use Case modeling, realization is conducted, which describes how a specific Use Case is realized in terms of the action sequence the actor invokes by notifying the system (Figure 16). A complete enumeration of DIORAMA's various use cases is provided elsewhere (Pradhan 2003).

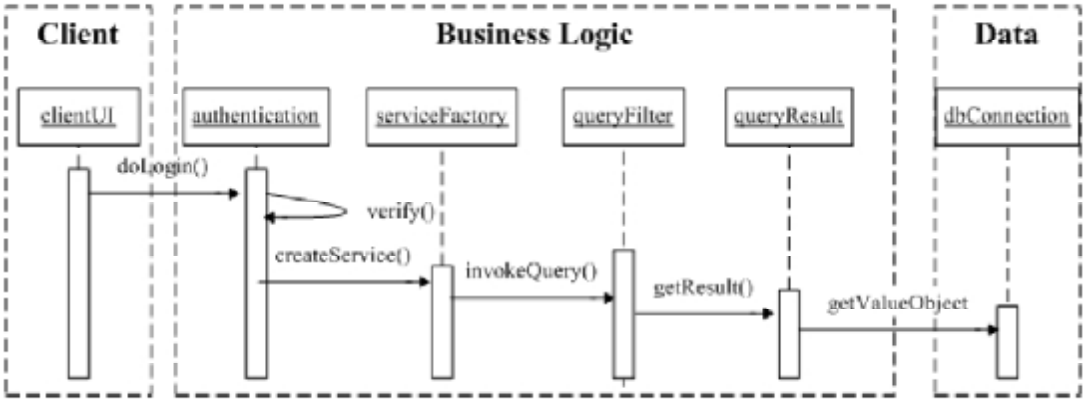


Figure 16 Sequence Diagram for Querying Database

6.3 NCSU Customized Client Application

For DIORAMA, a GIS client application was developed for a specialized solution to enter/update/delete data and perform spatial queries (Figure 17). This was the prototype's main emphasis to eventually assess comparative building performance from the WTC attacks. The application provides a graphical user interface (GUI), with mapping features enabling insertion, updating, and deletion of various data (e.g. photographs and CAD files). The GUI also supports the formulation of spatial and attribute-based queries. Query results are displayed graphically and tabularly. The application uses Java, a purely object-oriented language designed to enable the development of secure, high performance applications on multiple platforms in distributed networks (Gosling and McGilton 1996).

The GUI was developed using the Java Foundation Classes library, as it provided an extensive set of technologies to create an interactive format for client applications that run on multiple platforms (JFC 2003). In addition, ESRI's Java MapObjects 1.0 library was employed to perform geographically-based display, query, and data retrieval activities (MAPOBJECTS 2003). This application was based on a Model View Controller design pattern similar to the Java Application Framework (Sunkpho and Garrett 2003).

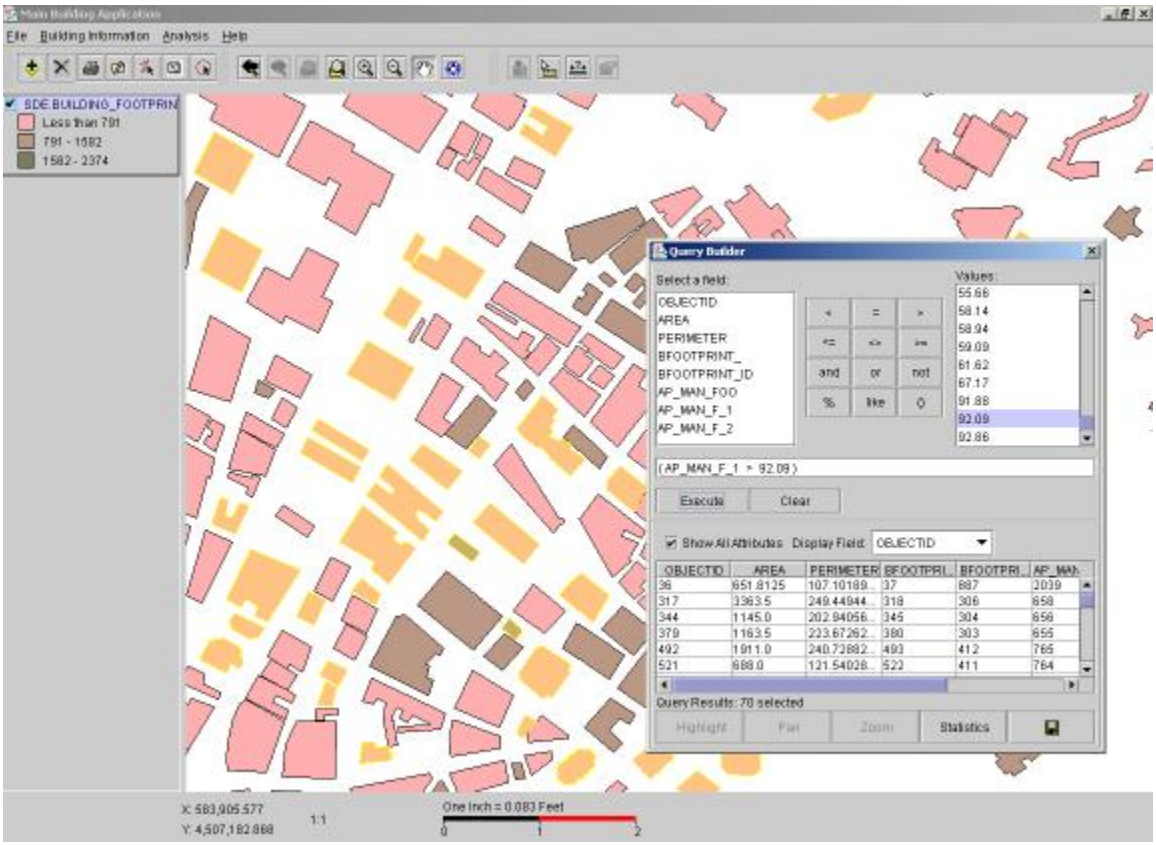


Figure 17 Example of Querying Portion of Graphical User Interface

6.3.1 Features of NCSU Client Application

The prototype customized client application was designed for usage ease and sophisticated spatial and attribute-based queries as illustrated below.

6.3.1.1 Ease of Use

Because real systems employ data entry personnel with minimal computer training, usage ease is critical. Apart from standard GUI components (e.g. menus, toolbars, textboxes, window interfaces), the DIORAMA's GUI provides visual maps displaying building footprints, street networks, and other infrastructure. While entering/updating data, personnel can visually associate data with a particular geographic entity – selecting a specific building from the map and displaying a list of available menu options to enter/update/delete building attribute data through multiple, interconnected data entry frames. For instance, data entry personnel can select a particular building from the map and then right-click the mouse button to display a list of options available. The list includes entering, updating, and deleting building information (Figure 18). Upon selection a particular option, a window frame opens up to add/update/delete the building information. Figure 19 shows a GUI frame to enter building information. This provides an intuitive and easy way to use the software.

The initial frame focuses primarily on building location, thereby establishing a spatial reference frame. Subsequent screens request increasingly detailed levels of information predominantly related to building materials, geometry, and configuration. A complete enumeration of DIORAMA's functionality, including flow diagrams, data structure definition, and user interface screen, is available elsewhere (Pradhan, 2003).

6.3.1.2 Robust Error Checking Mechanism

Apart from ease of use, error-checking mechanisms preclude the insertion of incorrect information in database. The error-checking mechanisms employed by the client GUI include combo boxes, radio buttons, and data type validation of text fields. The use of combo boxes is recommended for enumerated types (Figure 20a). A data type validation of text field ensures compliance with a particular standard. For instance, if a user enters a string data type into a field that accepts only integers, the incorrect data is rejected, and the user is given a warning message (Figure 20b).

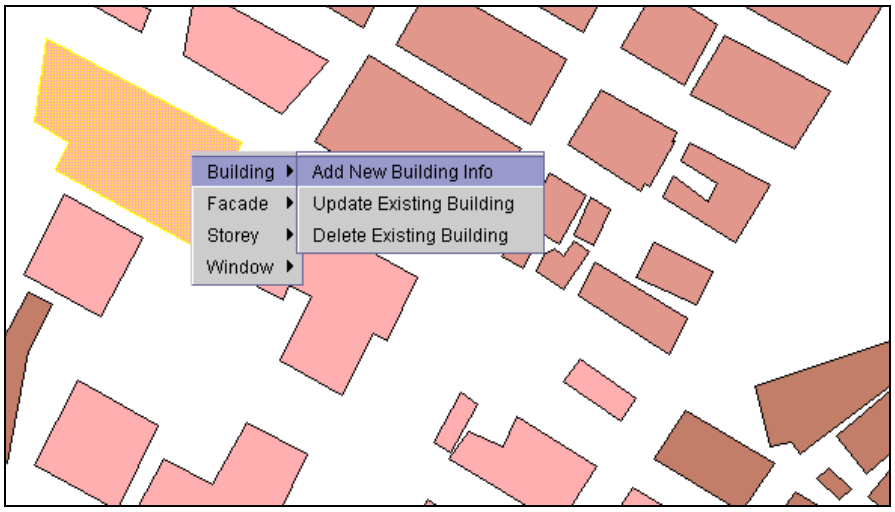


Figure 18 Popup Menu for Adding/Updating/Deleting Building Information

6.3.1.3 Support for Sophisticated Spatial and Attribute-based Queries

The client application enables complex spatial and attribute queries based on a single attribute table. Without customization, data must be distributed among multiple tables, which requires user comprehension of the underlying data structure to query, which may require programming skills. For instance, ESRI ArcMap users need to understand concepts of "Joins" and "Relates" to associate data

stored in tables with geographic features. When the user "Joins" two tables, the attributes from one table are appended onto the other, based on a field common to both tables. "Relates" only defines a relationship between two tables (also based on a common field) but does not append the attributes of one to the other.

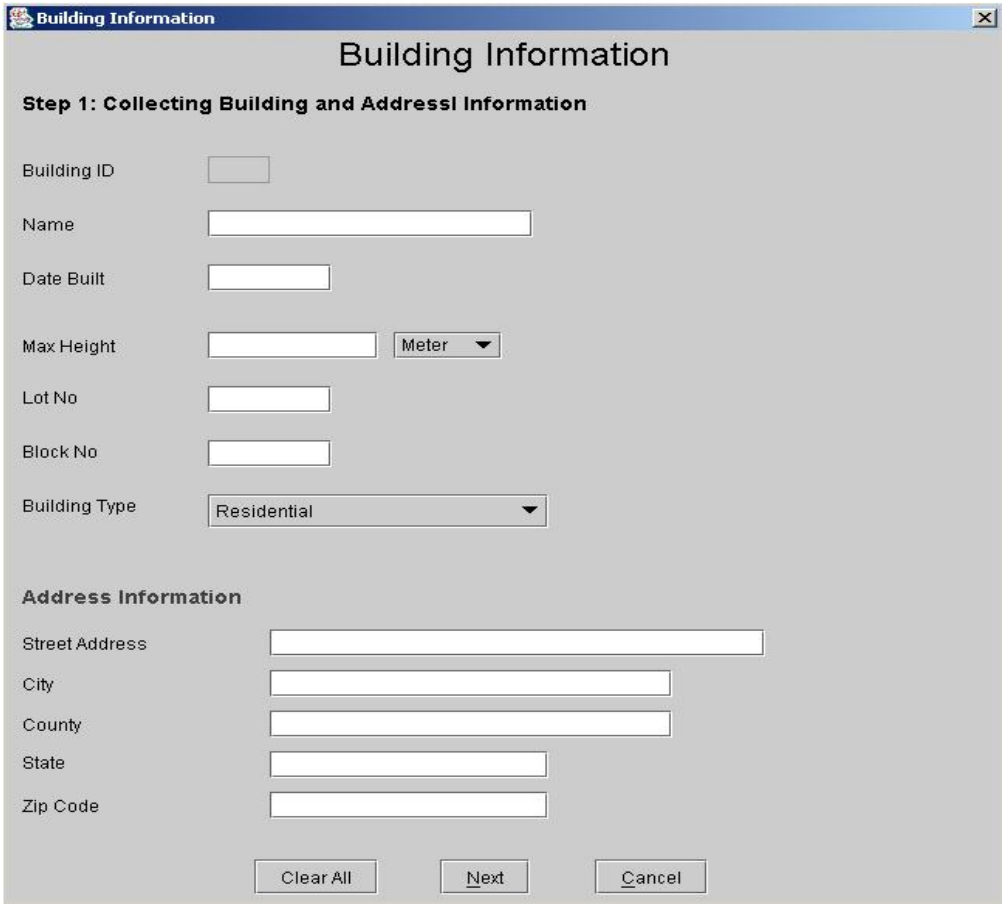
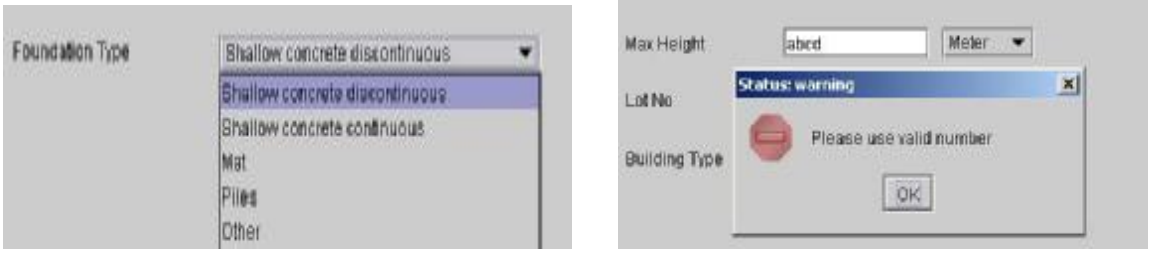


Figure 19 GUI Frame to Enter Building Information



(a) Combo Box (b) Text Field Validation

Figure 20 Error-Checking Mechanism

If the attribute data is stored in multiple tables, query formulation becomes an arduous task using Join and Relates. The DIORAMA client application precludes the need to know SQL to achieve complex spatial

and attribute queries (Figure 21). The user can simply checkbox the proper fields and then specify the conditions related to that field. Only the selection criteria need to be known. The following are the steps required for spatial query formulation.

(a) Selecting appropriate fields to display

A user can select the appropriate fields required for the query analysis by selecting the checkboxes (Figure 11). Figure 11 shows that the Building Name, Structural System, Structural Material, Façade Architectural CAD and Storey Height fields are selected.

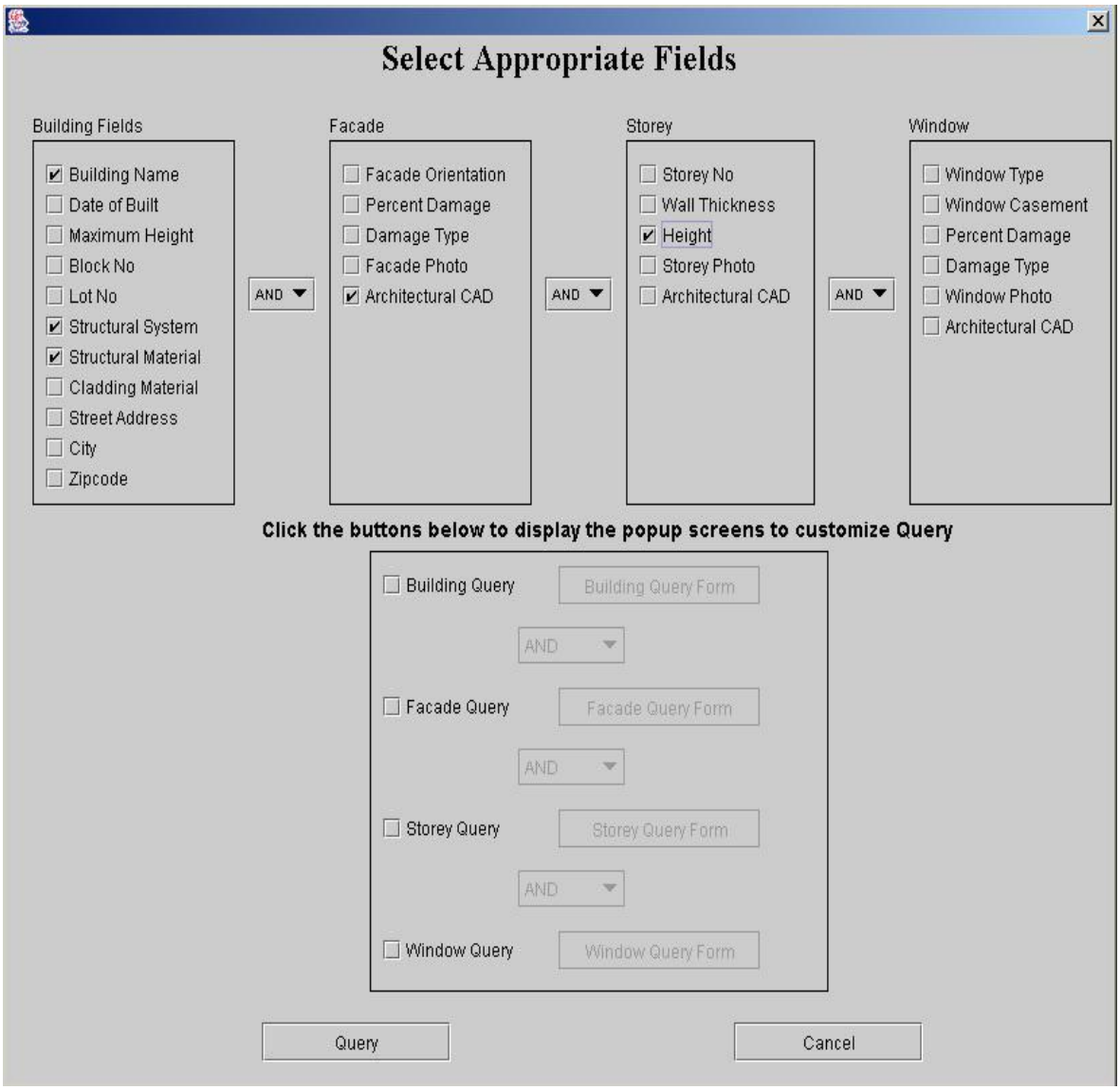


Figure 21 Query Interface

(b) Formulating Query

The lower section of the GUI contains a set of buttons that are used to formulate the customized query. The user can check the boxes adjacent to Building Query, Façade Query, Storey Query and Window Query label, in order to enable these adjacent buttons. For instance, by selecting the checkbox adjacent to Façade Query label, the user can enable the Façade Query Form button. The Façade Query Form button is

used to display the GUI, where the user can customize the query related to façade attributes (Figure 22). Figure 22 allows formulating a query to find the buildings that has any façade with post-blast damage percentage between 30 and 90. After making the proper selection, the user needs to click the Accept button. Similarly, other forms such as Building Query Form, Window Query Form and Storey Query Form can also be used to formulate the query.

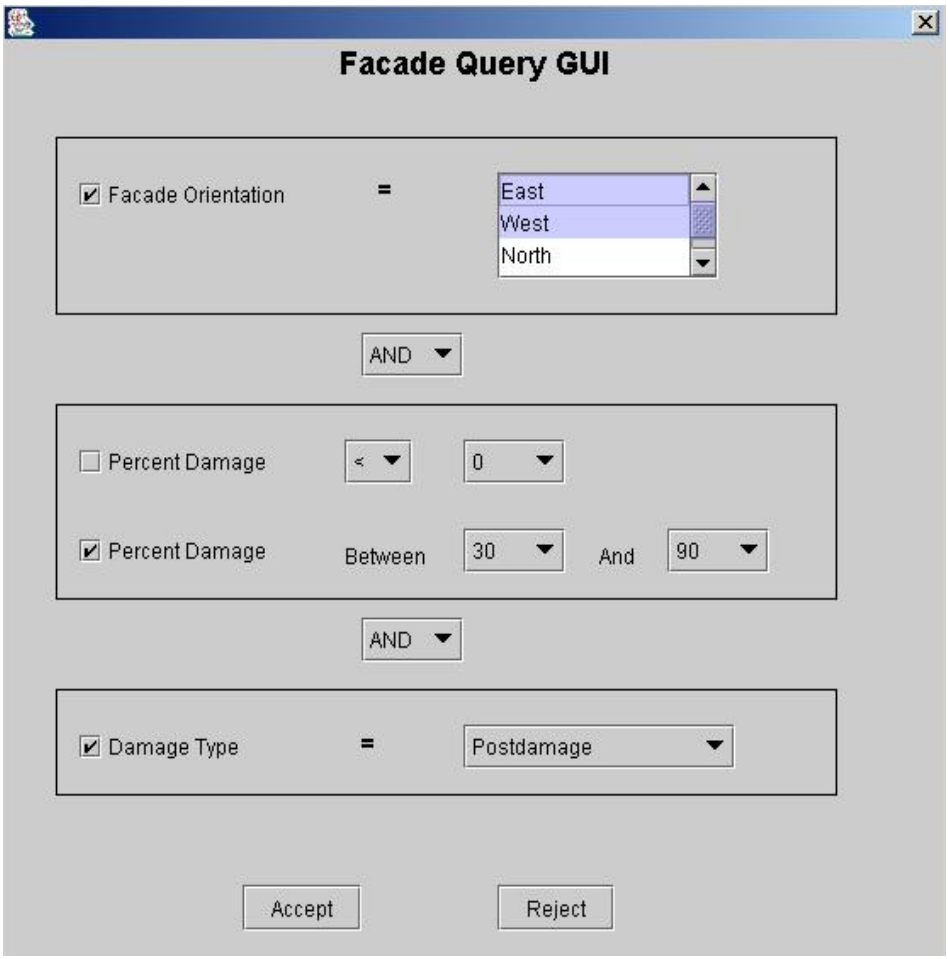


Figure 22 Façade Query GUI

(c) Viewing results

The database is queried by pressing the Query Button. A popup GUI with the selected attributes is displayed (Figure 23). In this example, the query returns two results. As shown in the Figure 23, the user can download the architectural drawing by clicking the button under the architectural drawing column. The selected buildings are graphically displayed after clicking the Done button. The selected buildings graphically appear as selected by being displayed colored yellow (Figure 24).

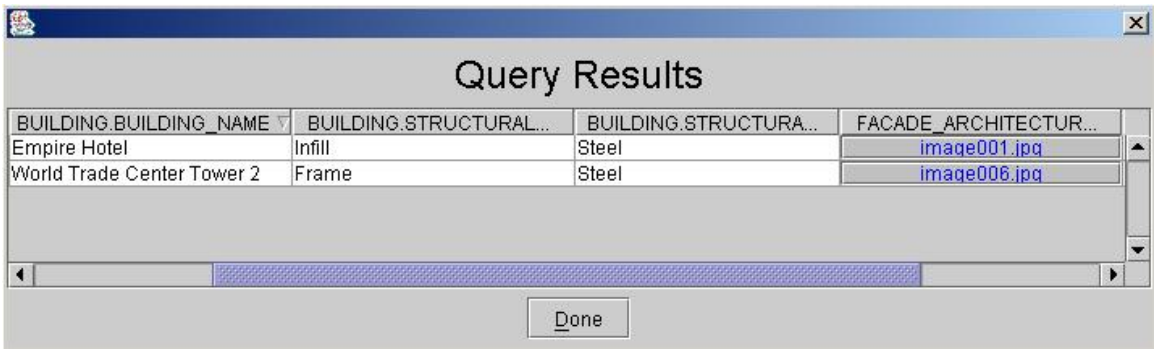


Figure 23 Query Results

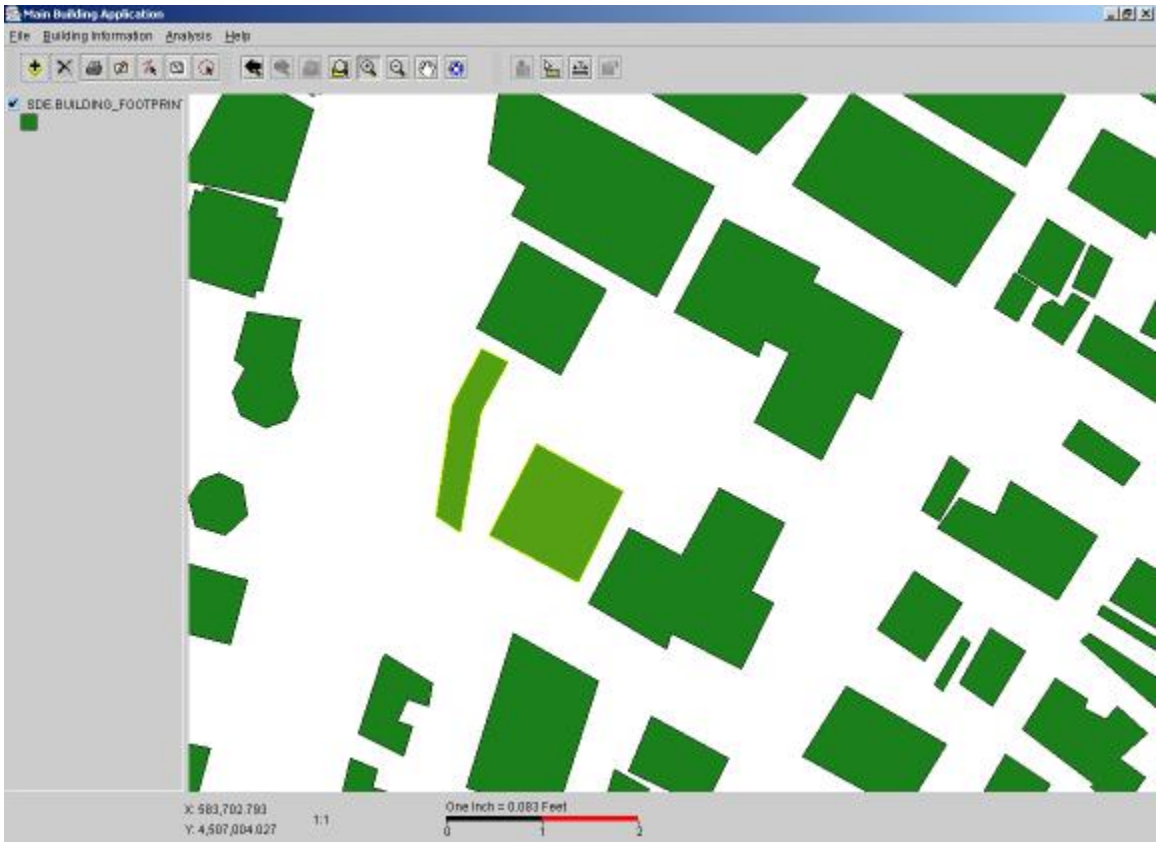


Figure 24 Displayed Query Results

As an example, DIORAMA is able to query damaged and undamaged structures based on extremely general or highly specific criteria. For instance, the system can search for every building that had at least 10 percent of its windows damaged, or it can search for every building built between 1914 and 1928 that lost between 20 and 30 percent of its windows at the height of at least 10 m above the ground on the façade that faces away from the World Trade Center complex. The system can display the results both graphically and tabularly, which facilitates establishing the perimeter of the impacted zone. Following large amounts of additional data entry, the system will be able to provide key pieces of fundamental information related to blast damage distribution. This will be done by coupling the results fields with algorithms that determine least distance from a single point of interest, such as where each hijacked plane

was estimated to be when the fuel tank exploded. Such information will greatly aid understanding of both existing building vulnerability and blast energy dissipation in a non-free field environment.

6.4 Parallel Computing

One of the primary objectives of the NCSU research is to design and evaluate the parallel software framework for prominent graph algorithms that are potentially useful for disaster management. The NCSU research team has not done cost-benefit analysis of using clusters over expensive parallel platforms like IBM SP, SGI Origin. These tasks can be (a) finding the shortest paths between two points in a road network, (b) predicting dispersion of hazardous material, and (c) estimating volumes of debris removal based on LIDAR data sets. These algorithms can be implemented either in serial or in parallel modes, but in serial implementation, only one processor is used to process the computation at a time, while in parallel implementation multiple processors are used simultaneously to perform the same computation. The use of multiple processors helps to solve complex problems in a real time, which is difficult to achieve with a single processor.

The NCSU research is mainly concerned with the design and implementation of parallel algorithm framework. As mentioned above, such a framework helps the programmers who are not the experts on parallel algorithms to use the software with great ease. NCSU's current framework supports different types of 'All-Pairs' Shortest Paths algorithms (Cormen et. al 1990) namely Dijkstra (Dijkstra 1959) and Floyd-Warshall (Cormen et. al 1990) algorithms. These algorithms are implemented using both Message Passing Interface (MPI) and OpenMP specifications, which are widely used in the parallel computing domain. The MPI model posits a set of processes or tasks that have only local memory but are still able to communicate with other processes by sending and receiving messages (Gropp et. al 1994), while OpenMP supports multi-platform shared-memory parallel programming model so that the running processes can access the common shared memory (OPENMP 2003).

The use of parallel computing can substantially reduce processing time, especially when a significant number of processors are available. For example, a typical 4500-nodes transportation road network (2 nodes required for each road segment) took 2231.4 seconds to find 'All-Pairs' shortest paths (using Parallel Floyd-Warshall) with one processor, while the same network required only 24 seconds with 144 processors (FIG. 15). A speedup of approximately 91.5 times is achieved. A similar level of speedup (92.3 times) was achieved when finding the 'All-Pairs' shortest path using Parallel Dijkstra algorithm, when the number of processors is increased from 1 to 128 (Figure 25); these algorithms were run on an IBM RS/6000 SP Supercomputer. A substantial amount of time savings occurs when multiple number of processors are used for data analysis. Appendix 10.3 consists of detailed documentation on both parallel Floyd and Dijkstra algorithms. Thus, parallel computing should be the preferred configuration for disaster management system, in order to achieve results of complex analysis in real-time.

6.5 DIORAMA's Contribution

Presently, DIORAMA supports data entry, update, retrieval, spatial querying, and two-dimensional spatial visualization. One component that distinguishes DIORAMA from off-the-shelf GIS software is DIORAMA's ability to directly query from multiple normalized tables, as in the case of normalized database schema. Off-the-shelf software accommodates only simple table structures. In DIORAMA, complex queries using multiple normalized tables can be performed directly with an intuitive GUI, which obviates the user having to understand complex table relations. Similarly, data entry/updating is conducted via a GUI – customized to a level not available in off-the-shelf products. To achieve these capabilities, as well as those outlined in the general criteria, the major area of customization implemented was in the system architecture, which was based on a distributed, three-tier architecture (data, business logic, and client). Each tier was designed and implemented individually and is installable on separate node. Node system heterogeneity is not a main issue as communication between the data tier and middle

tier is done using SQL (supported by most existing database vendors) and there is support for Java – the selected (platform independent) programming language.

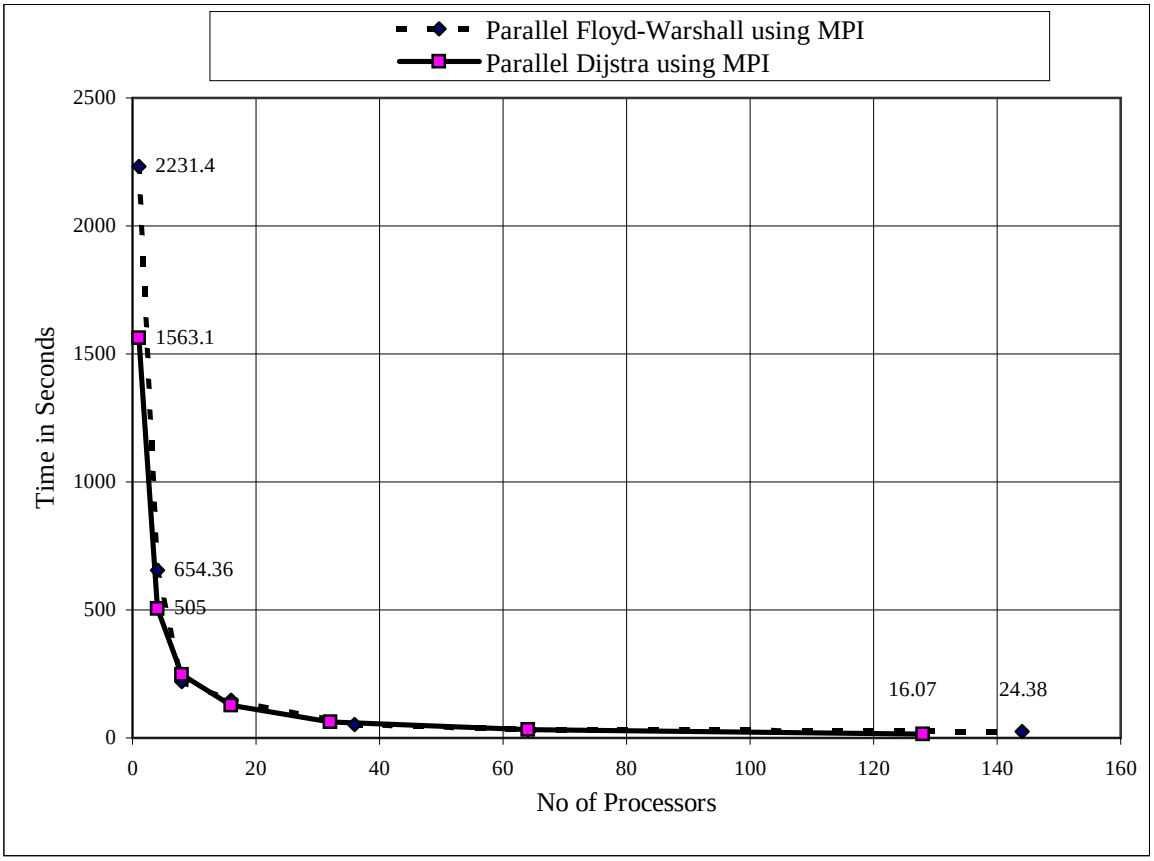


Figure 25 Speedup Graph for Parallel Dijkstra & Parallel Floyd-Warshall Algorithms

DIORAMA's strengths include the following: (a) reliable architecture through use of a relational database (data tier), spatial middleware (business-logic tier), and customized Java MapObject client (client-tier), in contrast to off-the-shelf GIS software, in which all components are coupled as one monolithic system; (b) non-resource intensive Java MapObject client applications, instead of resource intensive, commercial GIS software, (c) straightforward data sharing, since data is stored based on OGC open system standards (Simple Features Implementation Specification); and (d) simplified future upgrading, as business rules are transparent and coded for specific domains, as opposed to proprietary GIS applications, where business rules are coded to encompass many domains of expertise and are treated as black box applications. These features are fundamental to the ultimate use of DIORAMA as a multi-phase, disaster management system with ubiquitous computing capabilities.

DIORAMA's main, current functionalities can be summarized as follows: storage of spatial information in an Open GIS Consortium compliant data format, handling of complex spatial and attribute-based queries with an easy and intuitive GUI, allocation of resource intensive computation in real time, and support of multiple users for simultaneous accessing and updating within a common spatial repository. This combination of off-the-shelf and customized functionalities are enabled by the construction of a three-tier, Enterprise level GIS system. DIORAMA also has interoperability by usage of a multiple

platform compatible language (Java) and data storage in an OGC compliant file format, although no special tools or methodologies supported by OGC were implemented. System validation was based upon four items: (1) level of user friendliness, (2) ease of data entry/storage, (3) intuitive querying, and (4) real time processing speed. The first three were verified by successful undergraduate usage of the system and the fourth by data entry and retrieval without perceived interruption in user interaction. Follow on work includes large quantities of data entry and high-level analysis development for categorization of blast-energy dissipation in a non-free field environment.

7. CONCLUSIONS

An integrated disaster management system based on an enterprise system that combines GIS, DB, and CAD, offers substantial advantages over existing standalone systems in terms of data collection, administration, and retrieval. Current DMSs are presently focused on an overly restrictive definition of disaster management and, thus, do not adequately address the multi-phase nature that is fully reflective of a community's disaster management needs. As such, the architecture of existing systems heavily impedes full applicability and further expansion of these systems, as a direct outgrowth of their hardware and software limitations.

For communities to fully exploit their resources, and minimize duplication of efforts on an administrative level, an DMS is needed. Such a system provides an opportunity for profound advancement in areas of disaster management. By adopting an alternative architecture based on an Enterprise GIS framework, the six disaster management phases (*Identification*, *Prediction*, *Preparation*, *Mitigation*, *Response*, and *Recovery*) can be realized in terms of customization, computing resource distribution, data sharing, and upgrading.

DIORAMA contributes towards the implementation of guidelines through the outlining of a CDF requirement based on four critical aspects: depiction, integration, connection, and evolution. To date, existing disaster management systems are not designed to take into consideration these four aspects, which are essential to adequately address key challenges in terms of (i) the ability to design an efficient data model, (ii) the capability to share data among multiple organizations, (iii) the flexibility to encompass future upgrades, and (iv) the capacity to analyze temporal data evolution. Without addressing these, a cost effective, expandable, and flexible disaster management system cannot be achieved.

An integrated disaster management system based on an Enterprise system that supports GIS, a relational database, and multiple data formats offers substantial advantages over existing standalone systems in terms of data collection, administration, retrieval, distribution, and usage. The prototype architecture demonstrates the constructability of such a system through the employment of a data repository, a middleware system customized for disaster management, and customized client applications for querying blast damage in an urban area. The prototype exploits recent innovations in both hardware and software: enabling tabular and graphical queries for data analysis. From a software architecture perspective, the proposed prototype represents an advance over existing DMSs, because despite widespread industry support for three-tier software architecture, it has yet to be adopted by the disaster management community.

The full advantage of an integrated, spatially adaptable approach is hard to fully comprehend in its potential to revolutionize how disasters are considered. With a proper selection of input data and timely and rigorous updating of the system, large amounts of additional analysis for direct intervention and planning could be incorporated with respect to utility protection and redundancy, assessment of critical infrastructure, structural evaluation and intervention.

The deployed DIORAMA system demonstrates the constructability of such a system through the employment of a data repository system, middleware system, and customized client applications to provide a system that is a robust, reliable, and efficient before, during, and after a natural or man-made disaster. The DIORAMA architecture exploits recent advancements in both hardware and software. Preliminary tabular and visual queries for data analysis have proven possible. When coupled with a fully defined baseline system combined with powerful spatial querying, such a state-of-the-art disaster management system will be equipped to help minimize disaster related life loss and property damage.

8. FUTURE RESEARCH RECOMMENDATIONS

NCSU disaster management research will continue to evolve to incorporate the following capabilities:

- Development of a prototype of Service-oriented Inter-agency Participation Framework using Web Services.
- Accommodation of GPS handheld devices to collect field data and direct synchronization to the enterprise database system.
- Disaster specific analysis capabilities.

8.1 Service-Oriented Inter-Agency Participation Framework

The data maintenance and processing required for disaster response cannot be realistically managed by one organization. Theoretically, the design of a disaster management system could make use of services (both data maintenance and processing) of multiple organizations, instead of incurring the difficulty and expense of attempting to replicate services already available at other agencies. Any analysis or data maintenance task can be outsourced to another organization, if that organization has the proper infrastructure to support such processing. Thus, a network of co-operating agencies could be established to augment existing capabilities to be employed when needed. For instance, a resource intensive, complex analysis job, such as finding the shortest route to reach injured victims, in the aftermath the large magnitude earthquake amidst a dense, complex road network, can be outsourced to a supercomputing center for analysis. Likewise, the National Institute of Building Sciences can offer pre-disaster related services such as loss estimation of buildings caused by an earthquake, prior to a hazard event. An interagency participation network could be created similar to a set of distributed computing nodes, thus leveraging all available resources (Figure 26).

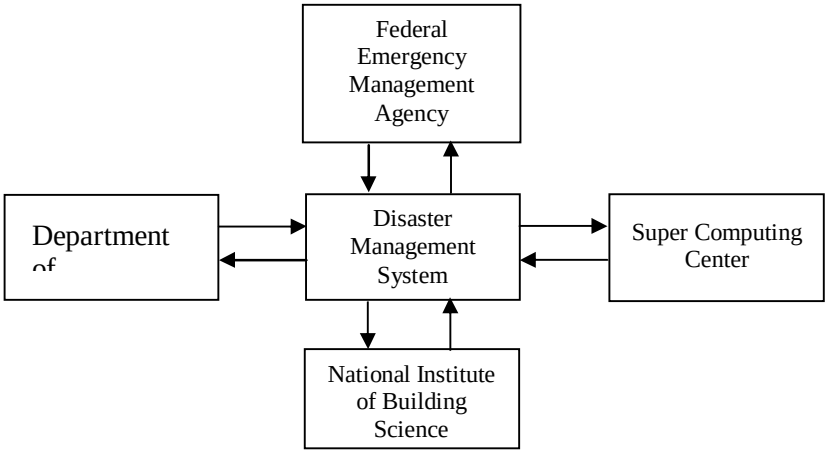


Figure 26 Interagency Participation Network

Building an enterprise scale disaster management system is a complex undertaking and may be considered as a collection of interacting services (Brown et. al 2002), where each service provides access to a well-defined collection of functionalities. In case of a disaster management system, typical services might be (1) finding shortest routes, (2) loss estimation, (3) quantifying debris removal, or (4) estimating temporary housing requirements. These services could be seamlessly integrated to form a service-oriented architecture (SOA). The SOA defines the services that comprise the system, describes the interactions that occur among the services to obtain certain behavior, and maps the services into one or more

implementations in specific technologies (Brown et. al. 2002). There are three main components to a SOA: (1) Service Requestor, (2) Service Registry, and (3) Service Provider (Figure 27).

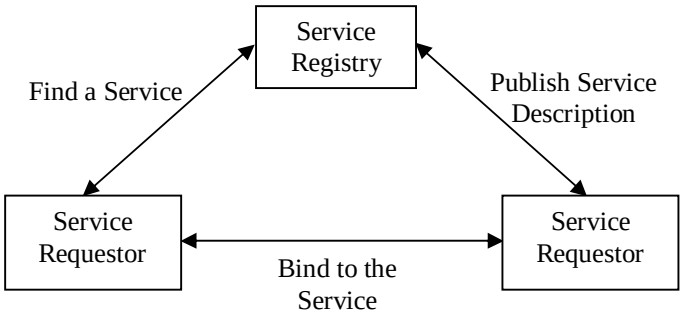


Figure 27 SOA Components

In an SOA, the Service Provider (the owner of the service) publishes its service description to a service registry, which is a central repository that facilitates service discovery by service requestors. The service description contains abstract definitions to invoke operations and exchange messages, and the bindings to a concrete network protocol. The service requestor finds an appropriate service from the service registry and selects the appropriate service(s) from the list of available services. Then it invokes the service directly at runtime using the binding information provided in the service description. An example can be a disaster management system (Service Requestor) invoking a shortest path service provided by a supercomputing center (Service Provider) with the help of a FEMA lookup database (Service Registry) that stores the information about the services made available by the list of supercomputing centers across the nation. Such a robust, reliable, and economical disaster management system can be developed through the integration of different participating agencies with the help of a new Web Service framework, where the participating disaster management agencies can interact and effectively access each other's resources.

Recent developments in XML (Extensible Markup Language) based Web Service architecture provide fertile ground for such a service-based architecture. Web services technology is derived from existing, distributed component technologies such as CORBA and DCOM technology (Caudwell et al. 2001) and offers a new framework for computing technology and standards, as well as the means to connect a network of distributed computing nodes (e.g. servers, workstations, desktop clients, and lightweight handheld clients) in a loosely coupled fashion. The loosely coupled architecture supports otherwise incompatible system technologies and creates composite services on demand. The loose coupling is also important for the implementation of complex collaborative applications needed during interagency participation. This framework is an integration of several key protocols and standards: XML, WSDL (Web Services Definition Language), SOAP (Simple Object Access Protocol), and UDDI (Universal Description, Discovery, and Integration). XML messages compliant with the SOAP specifications are exchanged between the requester and provider (SOAP 2000). The provider publishes a WSDL file that contains a description of the message and endpoint information to allow the requester to generate the SOAP message and send it to the correct destination (WSDL 2001). The service registry uses Universal Description, Discovery, and Integration protocol to enable service requestors to find and use Web services (UDDI 2002). The integration of web services (specific services offered by different agencies) into an Enterprise GIS framework (such as a disaster management system) offers tremendous new opportunities and capabilities for emergency management and hazard mitigation (Tsou and Buttenfield 2002).

8.2 Accommodation of GPS Handheld Devices in Enterprise System

The integrated use of a global positioning system (GPS), a geographic information system (GIS), digital photography, and handheld computing technology is gaining popularity to facilitate qualitative and quantitative data collection in post-disaster analysis system (Deaton and Frost 2001). NCSU system has the capability to incorporate such wireless hand-held devices. One of the main objectives of NCSU enterprise framework is the seamless integration of multiple types of hardware devices to access the central repository system. The following guidelines are recommended to incorporate GPS based wireless devices into DIORAMA:

- Web Service framework and
- Platform independent code.

8.2.1 Web Service Framework

The enterprise system can offer services that are related to information retrieval, update, and querying to wireless handheld devices. Prior to the availability of a Web Service Framework, existing enterprise services lacked a standard, nonproprietary method for communicating with the backend repository system. As a result, developers faced the prospect of having to write, maintain, and debug unique code for every conceivable type of handheld devices (J2ME 2003). With the help of a Web Service Framework, such handheld clients can exchange critical information with the central repository system using the standard protocols such as XML, SOAP, UDDI, and WSDL.

8.2.2 Platform Independent Code

A Web Service framework guarantees only the adoption of standard protocols regarding the invocation of services and the exchanges of messages. Different programming languages such as Java, C, C++ can be used to invoke services hosted by a Web Service Framework. In fact, this ability to accommodate services written in different languages is an advantage of this framework. The NCSU research recommends that platform independent language be used. Java language has currently been used as a language to achieve platform independence. The use of such language ensures that the code written for one brand of mobile device can be easily ported to another device brand. The interoperability helps to lower the cost of developing, maintaining, and debugging programs for diverse mobile devices (J2ME 2003). The NCSU research recommends the use of Java language primarily customized for small handheld devices, also known as Java 2 Platform, Micro Edition (J2ME). The JavaTM 2 Platform, Micro Edition (J2METM) is the Java platform for consumer and embedded devices such as mobile phones, PDAs, TV set-top boxes, in-vehicle telematics systems, and a broad range of embedded devices.

8.3 Disaster specific analysis capabilities

Then integrated nature of the constructed architecture and its interoperability allows it to be an ideal platform to host the development of many important future advances that are highly discipline specific and, thus, outside the scope of this current research.

9. REFERENCES

1. Adams, T.M., Malaikrisanachalee, S., Blazquez, C., Lueck, S. and Vonderohe, A. (2002). "Enterprise-wide data integration and analysis for oversize/overweight permitting," J. Comp. Civil Eng, 16(1), 11-22.

2. AGI GIS Dictionary. <http://www.geo.ed.ac.uk/agidict/welcome.html>(August 17, 2003).

3. ARCIMS (2003). "ArcIMS – GIS for the Internet," http://www.esri.com/software/arcims/index.html, accessed May 15, 2003

4. Beowulf (2003). "Beowulf at NASA/GSFC," http://beowulf.gsfc.nasa.gov/, accessed May 12, 2003

5. Bernstein, P. A., and Goodman, N. (1983). Multiversion Concurrency Control - Theory and Algorithms. ACM Transactions on Database Systems (TODS), v.8 n.4, p.465-483

6. Bernstein, P. (1996). Middleware: A Model for Distributed Systems Services, Communications of ACM, Vol. 39, No. 2, pp. 86 - 98

7. Brown, A., Johnston, S., and Kelly, K. (2003). Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications, A Rational software white paper from IBM

8. Brule, M.R. (2002). Deciphering .NET for GIS. Geospatial Solutions 12(6):32-7

9. Buyya, R. (1999). High Performance Cluster Computing: Architectures and Systems, ISBN 0-13-013784-7, Prentice Hall PTR, NJ, USA, 1999

10. Carroll, R. "Open standards for GIS from an utility perspective". <http://www.gisdevelopment.net/proceedings/gita/2003/sys/sys005.shtml>(October 18, 2003).

11. Cauldwell P, Chawla R, Chopra V, Damschen G, Dix C, Hong T, Norton F, Ogbuji U, Olander G, Rehman M A, Saunders K, and Zaev Z (2001). Professional XML Web Services, Birmingham, Wrox Press

12. Clément, G. and Larouche, C. (2000). "Open Geospatial Datastore Interface (OGDI): A geospatial data infrastructure solution". <http://www.gisdevelopment.net/technology/gis/techgi0057pf.htm> (September 30, 2005).

13. Cormen, T. H., Leiserson C. E., and Rivest, R. L. (1990). Introduction to Algorithms, MIT Press

14. Cutter, S. L. (2003). "GI Science, Disasters, and Emergency Management." Trans. GIS, 7(4), 439-46.

15. Darwin, H. and Date, C.J. Foundation for Object/Relational Databases: The Third Manifesto, 2nd ed., Addison-Wesley, 2000, pp. 15-30.

16. Deaton, S.L. and Frost J.D. (2001). " Integrated Digital Earthquake Reconnaissance," Seventh National Conference Earthquake Engineering (In press)

17. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs, Numerische Matematik 1 (1959) 269-271

18. Dittrich, K.R. Advances in Object-Oriented Database Systems. Lecture Notes in Computer Science, Vol, 334, Springer-Verlag, 1988.

19. DMI-Services (2003). "DMI-Services at a glance", http://www.cmi-services.org/whyCMIS.asp, last accessed July 11, 2003

20. Dorf, W. Personal Communication with New York City Department of Environmental Protection, January, 2002.

21. DTRA (2002). "Consequences Assessment Tool Set -Joint Assessment of Catastrophic Events" <http://www.dtra.mil/press_resources/fact_sheets/display.cfm?fs=cats-jace>(Sept. 30, 2005).

22. Eckerson, Wayne W (1995). "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications." Open Information Systems 10, 1 (January 1995): 3(20)

23. Elmasri and Navathe, S.B. (2000). Fundamentals of Database Systems, 3rd ed., Addison Wesley, Chapter 8, Pages 243-244 & Chapter 19, Pages 329-657

24. ESRI (1999). "GIS for Emergency Management." ESRI White Paper. http://www.esri.com/library/whitepapers/pdfs/homeland_security_wp.pdf(Aug. 30, 2005).

25. ESRI (2003). "Standards and Interoperability - Newsletter", http://www.esri.com/library/newsletters/standards-interoperability/sinews-summer03.pdf, last accessed July 18, 2003

26. Federal Geographic Data Committee (FGDC). (2002). http://www.fgdc.gov/standards.html (September 30, 2005).

27. Flax, L. K., Jackson, R. W., and Stein, D. N. (2002). "Community vulnerability assessment tool methodology." Natural Hazards Rev. 3(4), 163–176.

28. Fowler, M. and Scott, K. (1999). "UML Distilled: A Brief Guide to the Standard Object Modeling Language", 2nd ed., Addison Wesley, Chapter 3.

29. GAO (2003). "Major Management Challenges and Program Risks- Federal Emergency Management Agency," Performance and Accountability Series, United States General Accounting Office, page 6, January 2003

30. Godschalk, D. R. (2003). "Urban hazard mitigation: creating resilient cities." Natural Hazards Rev., 4(3), 101-167.

31. Goodchild, M. F. (2003). "Geospatial data in Emergencies." The geographical dimensions of terrorism. Ed.s Cutter, S. L., Richardson, D. B., and Wilbanks, T. J. Routledge, U.K., 99-104.

32. Gosling J and McGilton H (1996). "The Java Language Environment," http://java.sun.com/docs/white/langenv, accessed May 5, 2003

33. Greene, R. W. (2002). "Confronting Catastrophe: A GIS Handbook", ESRI Press

34. Gropp, W., Lusk, E. and Skjellum A. (1994). Using MPI- Portable Parallel Programming with the Message-Passing Interface, 1st ed., The MIT Press

35. Harris, E. A., Rasdorf, W. J., and Laefer, D. F. (2005). Determining disaster data needs in a multi-disaster context, Nat'l Science Fndn Final Report.

36. Hazen, D., Horrell, J., and Merrill-Oldham, J. (1998). "Selecting Research Collections for Digitization". <http://www.clir.org/pubs/reports/hazen/pub74.html>(August 17, 2003).

37. HAZUS (2005). "HAZUS- Multi Disasters, a loss estimation software," http://www.fema.gov/hazus/hz_index.shtm, accessed September 30, 2005

38. Huyck, C.K. and Adams, B.J. (2003). Engineering and Organizational Issues Related to The World Trade Center Terrorist Attack. MCEER Special Report Series, Memphis, Tennessee.

39. JFC (2003). " Java Foundation Classes (JFC) – Cross Platform GUIs & Graphics," http://java.sun.com/products/jfc/, accessed May 14, 2003

40. J2ME (2003). "Java™ 2 Platform, Micro Edition (J2ME™) Web Services Specification, " http://java.sun.com/j2me/docs/j2me_jsr172.pdf, accessed July 10, 2003

41. Johnson, G. (2003). "Solving disaster management problems using ArcGIS," <http://campus.esri.com> (Nov. 9, 2003).

42. Kevany, M.J. (2003). GIS in the World Trade Center attack – trial by fire. Computers, Environment and Urban Systems Elsevier, Vol. 27 pp. 571–583.

43. LLNL, (2003). "Introduction to Parallel Computing," http://www.llnl.gov/computing/tutorials/parallel_comp, accessed July 10, 2003

44. Laefer, D.F., Elliott, A., and Weller, L. (2002). "The Temporary Use of Drilled Shafts in the Renovation of Carnegie Hall." Deep Foundations 2002: an International Perspective on Theory, Design, Construction, and Performance, ed.s O'Neill, M.W. and Townsend, F.C., ASCE Special Geotechnical Publication No. 116, Vol. 1 p. 320-334: ASCE: New York, NY.

45. Laefer, D.F., Koss, A.L. and Pradhan, A.R. (2005a). "The Need for Baseline Data Characteristics for GIS-based Disaster Management Systems," J. Urban Planning, (in press).

46. Laefer, D.F., Pradhan, A.R., and Koss, A.L. (2005b). "GIS-based Disaster Management Systems: a Cogent Data Framework." Trans. Res. Board (in press).

47. Lim, K., Treitz, P., and St-Onge, B. (2001) "Estimation of Individual Tree Heights using LIDAR Remote Sensing," Proceedings of the Twenty-Third Annual Canadian Symposium on Remote Sensing, Quebec, QC, August 20-24, Vol. 1, pp. 243-250.

48. Liskov, B. and Guttag, J. Program Development in Java: Abstraction, Specification, and Object-Oriented Design", 1st ed., Addison-Wesley, 2000, pp. 167-168.

49. Liu, D., Cheng, J., Law, K. H., Wiederhold G., and Sriram, R. D. (2003). "Engineering information service infrastructure for ubiquitous computing." J. Comp. Civil Eng., 17(4), 219-229.

50. Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W. (2001). "Geographic Information Systems and Science", 1st ed., John Wiley & Sons, Ltd, Chapter11, Page226

51. MAPOBJECTS (2003). "MapObjects – Java Edition", http://www.esri.com/software/mojava/, accessed May 14, 2003

52. Macdonald, A (2001). "Building a Geodatabase", ESRI, Incorporated, Chapter 1, 2001

53. Mansourian, A., Rajabifard, A. Valadan Zoej, M. J., and Williamson, I. (2005). "Using SDI and web-based system to facilitate disaster management." Computers and Geoscience. (In press).

54. Mays, G. C. and Smith, D. Blast Effects on Buildings. Thomas Telford Publications, London, UK, 1995.

55. Merriam-Webster (2003). Merriam-Webster's Eleventh Collegiate Dictionary, Merriam-Webster, Springfield, MA

56. Miller. H. J. and Shaw, S. (2001). Geographic Information Systems for Transportations, 1st ed., Oxford University Press, Oxford, U.K., Chpt. 7, 219.

57. Mineter, M. J., and Dowers, S. (1999). "Parallel processing for geographical applications: A layered approach", *Journal of Geographical Systems* 1:61-74

58. Mineter, M. J., Dowers, S., and Gittings, B. M. (2000). "Towards a HPC Framework for Integrated Processing of Geographical Data: Encapsulating the complexity of Parallel Algorithms", *Transactions in GIS* Volume 4 Issue 3, Pages 245-262

59. Morais, M., 2000: Realizing the Benefits of an N-Tiered Enterprise GIS : 2000 ESRI User Conference taking place in San Diego, California from June 26 through June 30

60. NYC DoB (2005), "New York City Department of Buildings: Revision to Façade Law: Local Law 11 of 1998." <http://www.nyc.gov/html/dob/html/whatsnew2001.html> (May 15, 2003).

61. OGC, (2003). "OpenGIS® Implementation Specifications," http://www.opengis.org/techno/implementation.htm, accessed by July 21, 2003

62. OPENMP, (2003). "OpenMP: Simple, Portable, Scalable SMP Programming," http://www.openmp.org/, accessed May 15, 2003

63. Pena-Mora, F. and Dwivedi, G. H. (2002). "Multiple device collaborative and real time analysis system for project management in civil engineering." J. Comp. Civil Eng., 16(1), 23-28.

64. Pradhan, A. R. (2003). "Infrastructure Management Information System Framework for Disasters," MS thesis, Department of Civil, Construction, and Environmental Engineering, North Carolina State University, Raleigh, North Carolina, USA.

65. Pradhan, A.R., Laefer, D.F., and Rasdorf, W.J. (2005 in press). "Infrastructure Management Information System Framework for Disasters," Journal of Computing in Civil Engineering, ASCE.

66. ProQuest UMI (2003). "Digital Sanborn Maps,"<http://sanborn.umi.com>(September 30, 2005).

67. Radke, J. T., Cova, M. F., Sheridan, A. ,Troy, L. Mu, and R. Johnson (2000). "Challenges for GIS in emergency preparedness and response. ESRI white paper." http://www.esri.com/library/whitepapers/pdfs/challenges.pdf(Aug. 30, 2005).

68. Rasdorf, W. J. (2000). "Spatial Data Quality," Technical Report, Department of Civil Engineering, North Carolina State University, Raleigh, NC (March 2000).

69. Rumbaugh, J., Jacobson, I., and Booch, G. (1999). "The Unified Modeling Language Reference Manual," Addison-Wesley, Inc., Chapter1, 1999

70. SDTS, (1997). "American National Standard for Information Systems – Spatial Data Transfer Standard (SDTS) – Part 1. Logical Specifications," Draft for review, Published by American National Standard Institute, New York, page 5

71. Silberschatz, A. , Korth, H. F. and Sudarshan, S. (2001). "Database Systems Concepts with Oracle CD," McGraw-Hill, Chapter 2, 2001

72. Smith, A. Strategies for Building Digitized Collections. Digital Library Federation Council on Library and Information Resources Washington, D.C, 2001.

73. SOAP, (2000). "Simple Object Access Protocol (SOAP) 1.1 Specifications," http://www.w3.org/TR/SOAP/, accessed May 11, 2003

74. Sunkpho J. and Garrett, J. H. (2003). "Java inspection framework: developing field inspection support systems for civil systems inspection," J. Comp. Civil Eng., 17(4), 209-218.

75. Tsou, M., and Buttenfield, B. P. (2002). "A Dynamic Architecture for Distributing Geographic Information Services," *Transactions in GIS*, Volume 6, Issue 4, Pages 355-381

76. UDDI, (2002). "UDDI Version 3.0 --UDDI Spec Technical Committee Specification," http://uddi.org/pubs/uddi-v3.00-published-20020719.htm, accessed May 12, 2003

77. Uddin, N., and Engi, D. (2002). "Disaster management system for southwestern Indiana." Natural Hazards Rev., 3(1), 19-30.

78. Uddin, N., Engi, D., and Haque, A. (2003). "Community Preparedness and Response Model for Protecting Infrastructures," Proceedings of the Sixth U.S. Conference and Workshop on Lifeline Earthquake Engineering, ASCE, ed., Beaver, J.E., Long Beach, CA, August 10-13, p. 19-28.

79. Umar, A. (1997). Object-Oriented Client/Server Internet Environments, Prentice Hall, Inc

80. University of North Carolina at Chapel Hill. "Digital Library Services: Digitization Guidelines," <http://www.unc.edu/projects/diglib/docs/dig_guidelines01.pdf >(October 19, 2003).

81. USFDA (2003). <http://www.usaid.gov/our_work/humanitarian_assistance/disaster_assistance> (Sept. 30., 2005).

82. Weiser, M. (1993). "Some Computer science problems in ubiquitous computing." Comm. ACM, July. (reprinted as "Ubiquitous computing", Nikkei Electronics, Dec. 6, 1993, 137-143).

83. WSDL, (2001). "Web Services Description Language (WSDL) 1.1 specifications,"

http://www.w3.org/TR/wsdl, accessed May 12, 2003

84. Zerger, A. and Smith, D. I. (2003). “Impediments to using GIS for real-time disaster decision support,” Comp., En-vir. Urban Sys., 27, 123-141.

10. APPENDICES

10.1 Design of Data Model

The data model is the heart of an enterprise GIS. The data model captures the functional and operational requirements of the organization and must be able to integrate other information systems as well as to support the data maintenance process.

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules, which govern operations on the objects. The data model focuses on what data is required and how it should be organized, rather than what operations will be performed on the data. The data model is equivalent to an architect's building plans. Figure 28 shows a simplified description of the data model design process.

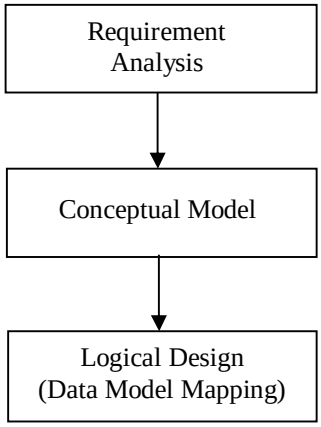


Figure 28 Data Modeling Process

10.1.1 Requirements Analysis

The act of Requirements Analysis is a crucial step during the data modeling process. Since the data model needs to capture the functional and operational requirements of the real world system, this physical system must be thoroughly understood for proper modeling to occur. The Requirements Analysis is a process to develop the overall requirement needs of the system and is often based on information collected in the initial phases. The following aspects were identified:

- Establish the scope of the system to be built and
- Establish a detailed understanding of the desired capabilities for the system.

10.1.1.1 Representation of Requirements Analysis

"Use Case" modeling has become the foundation for the de facto technique for performing software Requirements Analysis and specification. Use Case modeling is the process of describing the system behavior of the target system from an external point of view (Fowler et al. 1999). A Use Case focuses on the external aspects of a system and captures the functional and behavioral requirements of the system that help the users perform their tasks.

Use case diagrams contain actors and use cases. Actors are anyone or anything that may interact with the business. Use cases define a series of actions that benefit the initiating actor or actors. Examining the actors and defining what the actor will be able to do with the system helps to establish how a Use Case

will best be deployed. Since typically all the needs of a system cannot be covered in one Use Case, usually there is a collection of Use Cases. The collection of Use Cases specifies all the ways the system will be used. Use Cases provide a means to:

- Capture system requirements.
- Communicate with the end users and domain experts.
- Test the system.

Use Case diagrams are typically represented in Unified Modeling Language (UML), an industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, which simplifies the complex process of software design (Rumbaugh et al. 1999).

The first Use Case as shown in Figure 29 depicts the process for entering/updating and deleting information from the database from the end user. In this case, the use case actor is either a database administrator or data entry personnel. The user logs into the system, and credentials are verified. The authentication, authorization and accounting functionalities are delegated to the relational database management system (RDMS). As per pre-specified access privileges, the user is allowed insert, update, and delete the information (spatial and attribute) stored in the database management system.

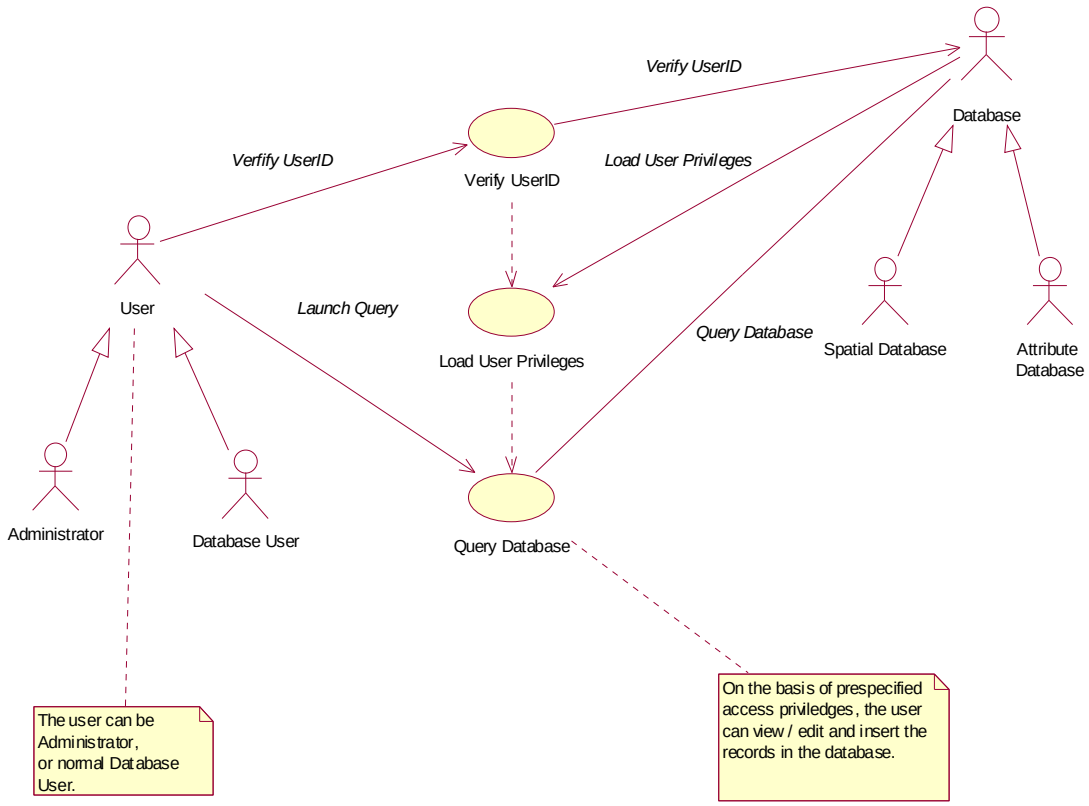


Figure 29 Use Case Modeling for Data Entry/Update/Delete

The second Use Case diagram (Figure 30) depicts the database querying process. To launch a query, a user logs into the system and credentials are verified. The authentication, authorization and accounting functionalities are delegated to the relational database management system (RDMS). Identical to the first

Use Case, as per the pre-assigned access privileges, the user is allowed to query the database. The query results are displayed on the client application in graphical and tabular formats.

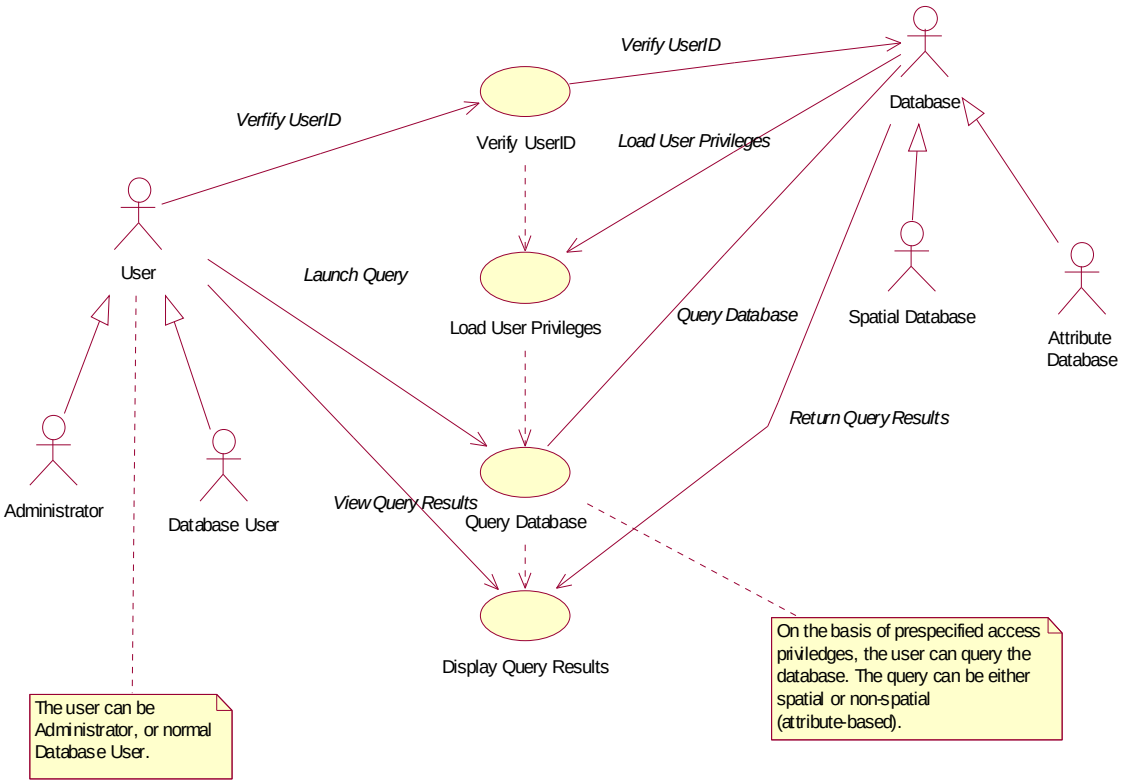


Figure 30 Use Case Modeling for Querying Database

10.1.2 Conceptual Model

Once all the requirements have been collected and analyzed, the next step is called conceptual design, in which a conceptual schema for the database is created using a high-level conceptual data model (Elmasri and Navathe 2000). The high-level conceptual data model captures the details of the Requirements Analysis without being dependent on any particular data model. For instance, when the data model is created as a set of relational tables in a database system, the user has to know the technical details about creating and managing tables. This will hinder the process of frequent updates required during initial modeling, as the environment being modeled changes. Instead, the use of high-level conceptual data model avoids such frequent changes and helps the structure of a database to evolve over time as the environment being modeled changes.

The conceptual schema represents a concise description of the data requirements of the users and consists of detailed descriptions of the entity types, relationships, and constraints. Each is represented using a high-level data model called an Entity-Relationship (ER) model. Since these concepts do not have technical or vendor specific implementation details, they are easier to understand and can be used to communicate with non-technical users. For example, the user does not need to have the technical knowledge of Structured Query Language (SQL) to design a data model. SQL is a declarative relational database language for inserting, updating, querying and protecting data.

In an ER model, the basic object is an entity that represents a "thing" in the real world with an independent existence. In general, an entity can either be an object with physical existence such as a particular building, window, façade as in case of IMIS project, or it may be an object with a conceptual existence such as boring log. A boring log is something physical- it is like a visual report that describes a field investigation. Each entity has attributes that describe the particular properties of that particular entity. For instance, a building entity can have attributes of maximum height, address, width, and date of construction. Every particular entity will have a value for each of its attributes. The attribute values that describe each entity become the bulk part of the data stored in the database.

10.1.2.1 Different Entity Attribute Types

Each entity has attributes that describe the particular properties of the entity. For instance, a building entity has a maximum height, maximum width, and street address attributes. The attributes are classified under different types. There are four main entity types:

- Simple versus composite,
- Single-valued versus multi-valued,
- Stored versus derived, and
- Complex attributes.

10.1.2.1.1 Simple Versus Composite

Attributes that cannot be further divided are called simple or atomic attributes, while composite attributes are those capable of further division into smaller subparts representing finer basic and independent attributes. For example, the maximum height of a building is a simple attribute, but an address attribute of the same building can be further divided into its street address, city, county, and zip code.

10.1.2.1.2 Single-Valued Versus Multi-Valued Attributes

Attributes that have a single value for a particular entity are called single-valued attributes. The maximum height of a building is a single-valued attribute. In contrast, the boring log attribute of a building can be multi-valued, if the building has more than one boring log affiliated with it.

10.1.2.1.3 Stored Versus Derived Attributes

In some cases, two or more attribute values are related – for example, the zip code and state. A US state can be easily determined from the zip code value as every zip code is affiliated with only one state. Hence, the state attribute is the derived attribute, and the zip code is the stored attribute.

10.1.2.1.4 Complex Attributes

Composite and multi-valued attributes can be nested in an arbitrary way to form a complex attribute. For example, a building address (which is a composite attribute) that has two street addresses because it has entrances on two different streets.

10.1.2.2 Representation of ER Model

An ER model is typically represented by an ER diagram. The notation for ER diagrams is depicted in Figure 31. The benefits of ER diagram are its relative simplicity and pictorial clarity. There is a pre-established convention for notion. This diagramming technique is used as a de facto standard for representing logical structure of the database. Other diagramming techniques, such as Unified Modeling Language (UML), are gaining popularity as well. UML is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. In ER diagram, entities are usually represented a rectangle boxes. As previously defined, an entity can be any physical thing, such as

building, or it can be conceptual existence such as boring log. Attributes are presented by ovals attached to their entity type by straight lines. Composite attributes are attached to their component attributes by straight lines, while multi-valued attributes are displayed in double ovals. An entity type that does not have key attributes of its own is called weak entity type and is represented by a double box.

An ER diagram also supports modeling relationships between various entity types. A relationship is an association among several entities (Silberschatz et al. 2001). Whenever an attribute of one entity type refers to another entity type, some relationship exists. For example, the attribute BUILDING_ID of Storey entity refers to the building for which the storey is a part of it. Such references should be captured as relationships, not as attributes. In an ER diagrams, relationships are displayed as diamond-shaped boxes, which are connected by straight lines to the rectangular boxes representing entity types.

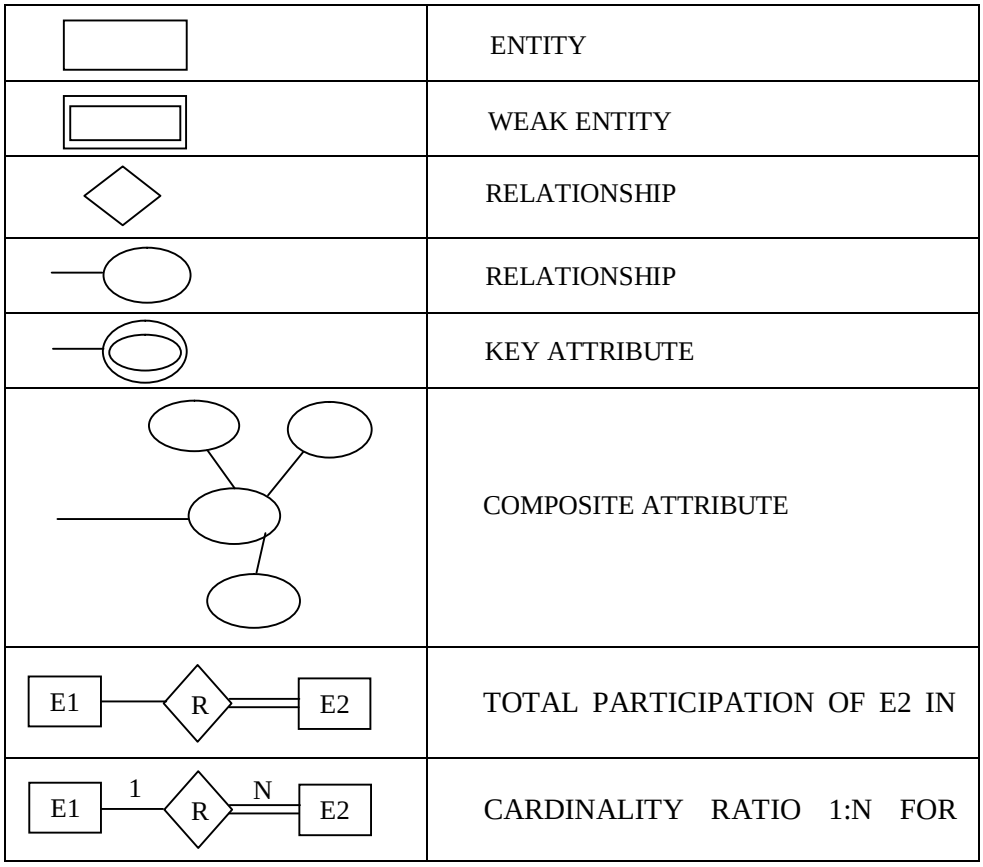


Figure 31 Notations for ER diagrams (adapted from Elmasri and Navathe 2000)

10.1.3 ER – to – Relational Mapping

An ER model gives a conceptual model of the world that is to be represented in a database. To implement the database, the Relational Model needs to be used. Mapping the ER model to the Relational model provides well-designed, relational tables. The following guidelines will result in such a structure by creating specific, identifiable mappings between various entities.

10.1.3.1 Regular Entity Types

- Create a new Relation State that includes all the simple attributes. A Relation State, also denoted as r(R), is a set of n-tuples r = $\{t_1, t_2, \ldots t_m\}$. Each n-tuple is an ordered list of n values t = $<v_1, v_2, v_3, \ldots, v_n>$.

- For a composite attribute, include only the simple component attributes and
- Omit multi-valued attributes (see multi-valued attributes).

- Pick a Primary Key (PK). PK of a relational table uniquely identifies each record in the table (such as the unique building_id for each building in IMIS system). Consider the building entity of ER diagram shown in Figure 32. A building Relation State is created with all of its simple attributes such as maximum height or data of built. Then the BUILDING_ID is assigned as the primary key for the building Relation State (Figure 33).

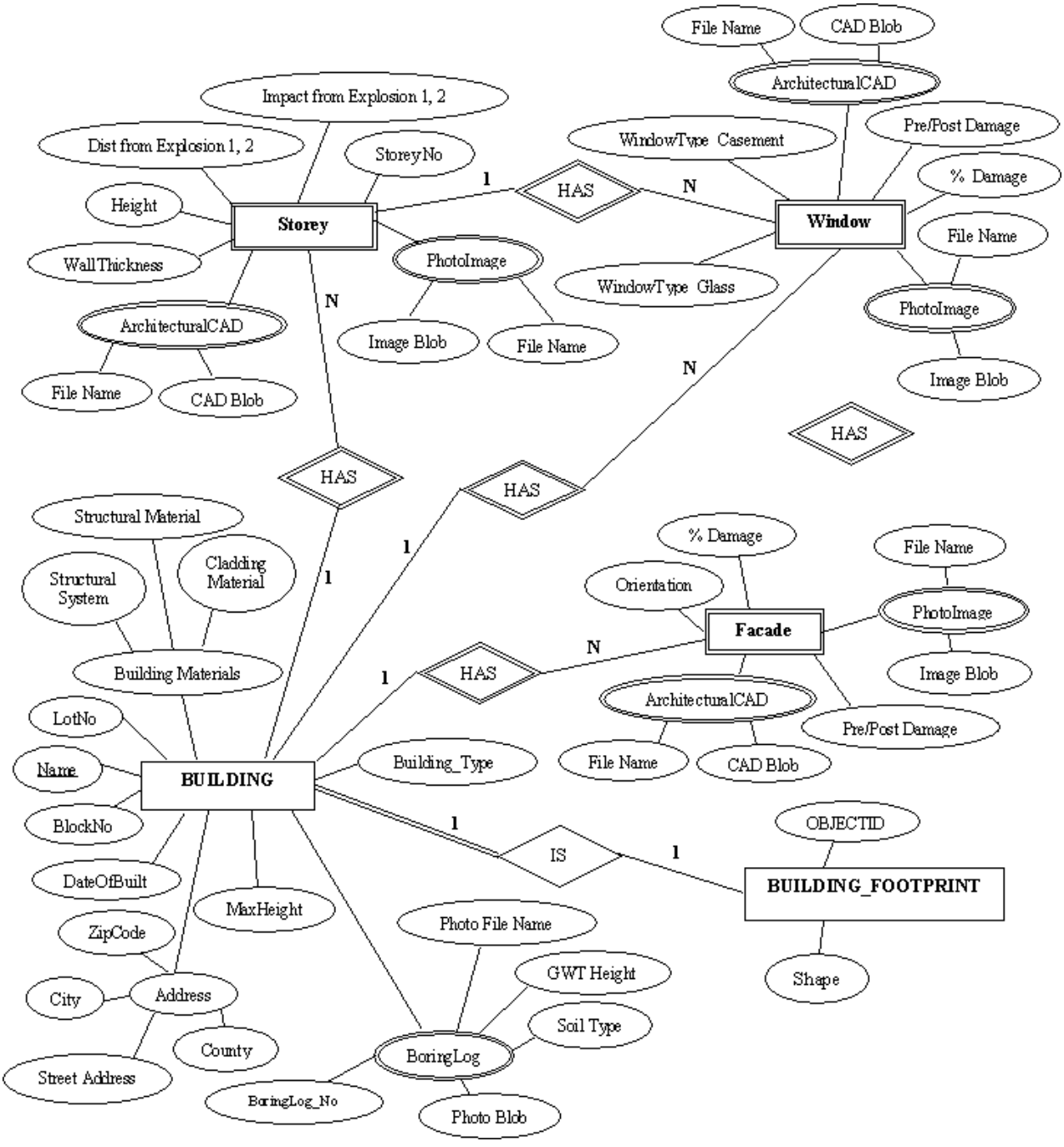


Figure 32 ER Diagram for DIORAMA

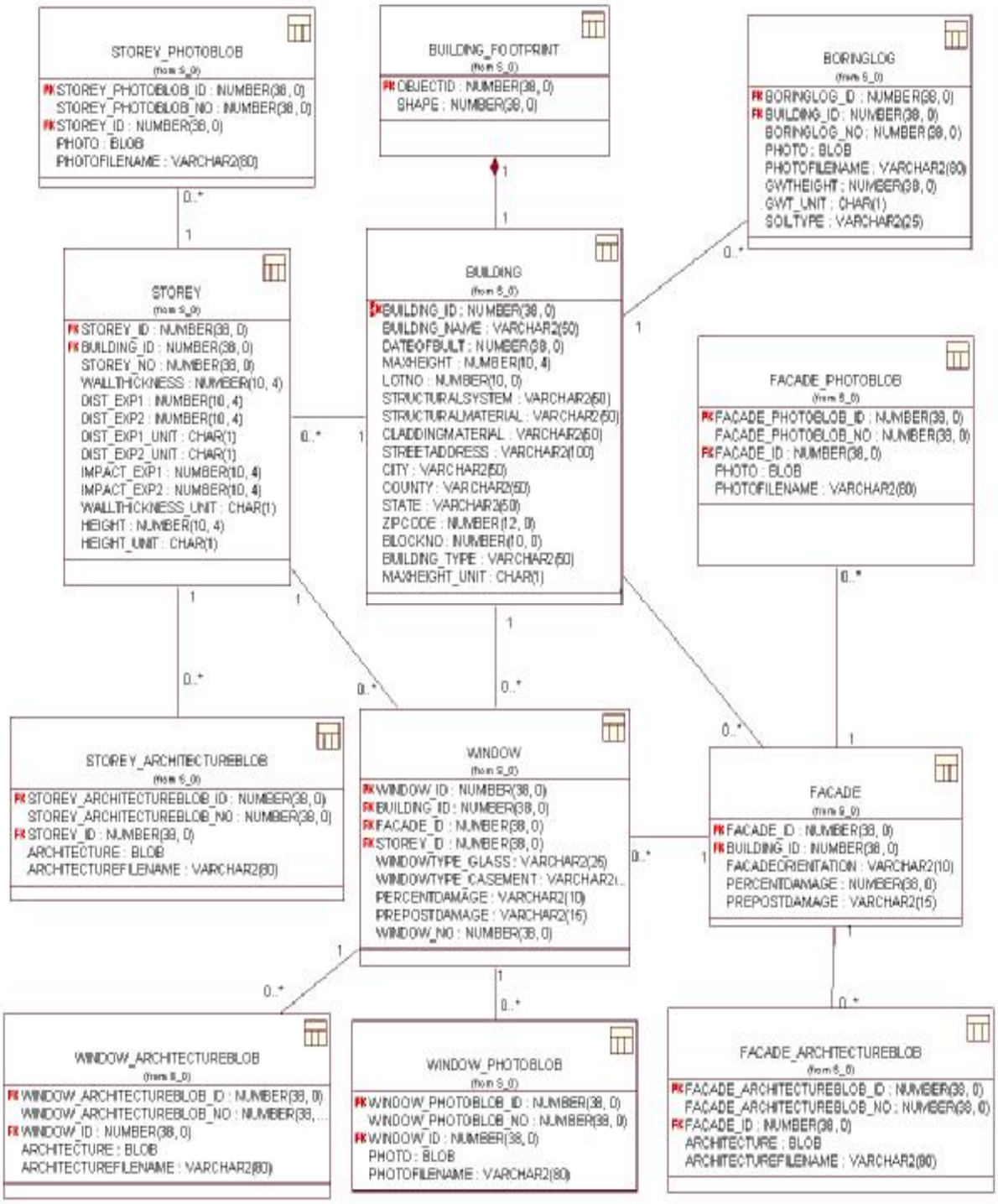


Figure 33 UML Conceptual Schema for DIORAMA Database

10.1.3.2 Weak Entity Types

- Create a new Relation State.
- Include all simple attributes (or simple components of composite attributes).
- Include a foreign key (FK) to the owner entity's relationship. A FK is defined as a field in a relational table that matches the primary key column of another table. The foreign key is also used to cross-reference tables. A PK for a weak entity type is the combination of the FK and a partial key. The partial key is defined as the part of the PK. In the case of a façade entity (weak entity), a new Relation State Façade is created with all of its simple attributes, which may include the number of windows and the building's cladding material. BUILDING_ID (PK of Building relationship) is included as a foreign key in Façade Relation State. The primary key for the Relation State is the combination of FACADE_ID and BUILDING_ID. Since FAÇADE_ID is a part of PK in Façade Relation State, it is also called as a partial key.

10.1.3.3 1:1 Relationships

- Add to one of the participating Relation States an FK to the other Relation State. Identify the Relation States (suppose two Relation States S and T) that are participating in a relationship (R). Choose one of the relationships (Select S) and include the PK of T as a FK in S. Building and Building_Footprint entities have a 1:1 relationship with each other. BUILDING_ID attribute of Building_Footprint Relation State is added as an FK to Building Relation State.

10.1.3.4 1:N Relationships

- Add to the Relation State on the N-side an FK of the other Relation State. As building has many stories, the Storey Relation State belongs to N-side of 1:N relationship between Storey and Building Relation States. To model this relationship between Building and Storey Relation States, add a Building_ID (PK of building Relation State) as an FK to Storey Relation State.

10.1.3.5 Multi-Valued Attribute

- For each multi-valued attribute, A:
  - Create a new Relation State.
  - Include an attribute or attributes corresponding to A.
  - Include the PK of the Relation State that represents the entity, which has A as an attribute. For example, add façade_id (the PK of Façade Relation State) to the Façade_Architectureblob Relation State (multi-valued attribute).

Figure 33 shows the UML schema for DIORAMA. Appendix 10.2 is the SQL code for generating the IMIS schema in any standardized relational database management system. The schema has been normalized to the Third Normal Form (3NF). Normalization is a process of analyzing the given relation to achieve the desirable properties of (1) minimizing redundancy, and (2) minimizing data insertion, deletion, and update anomalies.

Table 4 is an example of a table that is not normalized. The Building_ID, BuildingName, Zip code and BlockNo columns have repeated values in Table 4 that would have to be manually entered with each new entry. In contrast, Table 5 has the same set of data in a non-redundant format. In this case, only one unique identifier related to the building needs to be entered. Thus, the database community has developed a series of guidelines for ensuring that databases are normalized. These guidelines are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). The higher levels of normalization make the

relational data model less prone to redundancy and update anomalies. In practical applications, 1NF, 2NF, and 3NF along with the occasional 4NF are used.

Table 4 Non-Normalized Table

Building_ID	BuildingName	Zipcode	BlockNo	Storey_NO	Storey_Height
1	A1	27607	31	1	10
1	A1	27607	31	2	12
1	A1	27607	31	3	11
2	A2	27617	2	1	9
2	A2	27609	14	2	10

Table 5 Normalized Tables

Building_ID	BuildingName	Zipcode	BlockNo
1	A1	27607	31
2	A2	27617	2

Building_ID	Storey_NO	Storey_Height
1	2	12
1	3	11
2	1	9
2	2	10

10.2 SQL Code

8.2.1 SQL Code for Creating Tables

```
CREATE TABLE BUILDING_FOOTPRINT
        (OBJECTID INTEGER NOT NULL,
         SHAPE INTEGER NOT NULL,
         CONSTRAINT BF_PK PRIMARY KEY (OBJECTID));

CREATE TABLE BUILDING
   (BUILDING_ID INTEGER NOT NULL,
        BUILDING_NAME VARCHAR(50) NOT NULL,
        DATEOFBUILT INTEGER ,
        MAXHEIGHT NUMBER(10, 4),
        LOTNO          NUMBER(10),
        STRUCTURALSYSTEM VARCHAR(50),
        STRUCTURALMATERIAL VARCHAR(50),
        CLADDINGMATERIAL VARCHAR(50),
        STREETADDRESS VARCHAR(100),
        CITY VARCHAR(50),
        COUNTY VARCHAR(50),
        STATE VARCHAR(50),
        ZIPCODE NUMBER(12),
        BLOCKNO NUMBER(10),
        BUILDING_TYPE VARCHAR(50),
        MAXHEIGHT_UNIT CHAR(1),
        CONSTRAINT BUILDING_PK PRIMARY KEY (BUILDING_ID),
        CONSTRAINT BUILDINGNAME_UNQ UNIQUE (BUILDING_NAME),
```

```
  CONSTRAINT BUILDING_FK FOREIGN KEY (BUILDING_ID) REFERENCES BUILDING_FOOTPRINT(OBJECTID) ON DELETE CASCADE);

CREATE TABLE BORINGLOG
        (BORINGLOG_ID INTEGER NOT NULL,
         BUILDING_ID INTEGER NOT NULL,
         BORINGLOG_NO INTEGER NOT NULL,
         PHOTO BLOB,
         PHOTOFILENAME VARCHAR(80),
         GWTHEIGHT INTEGER,
         GWT_UNIT CHAR(1),
         SOILTYPE VARCHAR(25),
         CONSTRAINT BORINGLOG_PKS PRIMARY KEY (BORINGLOG_ID),
         CONSTRAINT BORINGLOG_UNQ UNIQUE (BUILDING_ID, BORINGLOG_NO),
         CONSTRAINT BORINGLOG_FK FOREIGN KEY (BUILDING_ID) REFERENCES BUILDING(BUILDING_ID) ON DELETE CASCADE);

CREATE TABLE FACADE
        (FACADE_ID INTEGER NOT NULL,
        BUILDING_ID INTEGER NOT NULL,
        FACADEORIENTATION VARCHAR(10) NOT NULL,
        PERCENTDAMAGE INTEGER,
        PREPOSTDAMAGE VARCHAR(15),
        CONSTRAINT FACADE_PKS PRIMARY KEY (FACADE_ID),
        CONSTRAINT FACADE_UNQ UNIQUE(BUILDING_ID, FACADEORIENTATION),
        CONSTRAINT FACADE_FK FOREIGN KEY (BUILDING_ID) REFERENCES BUILDING(BUILDING_ID) ON DELETE CASCADE);

CREATE TABLE FACADE_PHOTOBLOB
        (FACADE_PHOTOBLOB_ID INTEGER NOT NULL,
         FACADE_PHOTOBLOB_NO INTEGER NOT NULL,
         FACADE_ID INTEGER NOT NULL,
         PHOTO BLOB,
         PHOTOFILENAME VARCHAR2(80),
         CONSTRAINT FACADE_PHOTOBLOB_PK PRIMARY KEY (FACADE_PHOTOBLOB_ID),
         CONSTRAINT FACADE_PHOTOBLOB_UNQ UNIQUE (FACADE_PHOTOBLOB_NO, FACADE_ID),
         CONSTRAINT FACADE_PHOTOBLOB_FK FOREIGN KEY (FACADE_ID) REFERENCES FACADE (FACADE_ID) ON DELETE CASCADE);

CREATE TABLE FACADE_ARCHITECTUREBLOB
        (FACADE_ARCHITECTUREBLOB_ID INTEGER NOT NULL,
         FACADE_ARCHITECTUREBLOB_NO INTEGER NOT NULL,
         FACADE_ID INTEGER NOT NULL,
         ARCHITECTURE BLOB,
         ARCHITECTUREFILENAME VARCHAR2(80),
         CONSTRAINT FACADE_ARCHITECTUREBLOB_PK PRIMARY KEY (FACADE_ARCHITECTUREBLOB_ID),
         CONSTRAINT FACADE_ARCHITECTUREBLOB_UNQ UNIQUE (FACADE_ARCHITECTUREBLOB_NO, FACADE_ID),
```

```
        CONSTRAINT FACADE_ARCHITECTUREBLOB_FK FOREIGN KEY (FACADE_ID)
REFERENCES FACADE (FACADE_ID) ON DELETE CASCADE);

CREATE TABLE STOREY
        (STOREY_ID INTEGER NOT NULL,
        BUILDING_ID INTEGER NOT NULL,
        STOREY_NO INTEGER NOT NULL,
        WALLTHICKNESS NUMBER(10, 4),
        DIST_EXP1 NUMBER(10, 4),
        DIST_EXP2 NUMBER(10, 4),
        DIST_EXP1_UNIT CHAR(1),
        DIST_EXP2_UNIT CHAR(1),
        IMPACT_EXP1 NUMBER(10, 4),
        IMPACT_EXP2 NUMBER(10, 4),
        WALLTHICKNESS_UNIT CHAR(1),
        HEIGHT NUMBER(10,4),
        HEIGHT_UNIT CHAR(1),
        CONSTRAINT STOREY_PKS PRIMARY KEY (STOREY_ID),
        CONSTRAINT STOREY_UNQ UNIQUE (BUILDING_ID, STOREY_NO),
        CONSTRAINT STOREY_FK FOREIGN KEY (BUILDING_ID) REFERENCES
BUILDING(BUILDING_ID) ON DELETE CASCADE
        );

CREATE TABLE STOREY_PHOTOBLOB
        (STOREY_PHOTOBLOB_ID INTEGER NOT NULL,
         STOREY_PHOTOBLOB_NO INTEGER NOT NULL,
         STOREY_ID INTEGER NOT NULL,
         PHOTO BLOB,
         PHOTOFILENAME VARCHAR2(80),
         CONSTRAINT STOREY_PHOTOBLOB_PK PRIMARY KEY (STOREY_PHOTOBLOB_ID),
         CONSTRAINT STOREY_PHOTOBLOB_UNQ UNIQUE (STOREY_PHOTOBLOB_NO,
STOREY_ID),
         CONSTRAINT STOREY_PHOTOBLOB_FK FOREIGN KEY (STOREY_ID) REFERENCES
STOREY (STOREY_ID) ON DELETE CASCADE);

CREATE TABLE STOREY_ARCHITECTUREBLOB
        (STOREY_ARCHITECTUREBLOB_ID INTEGER NOT NULL,
         STOREY_ARCHITECTUREBLOB_NO INTEGER NOT NULL,
         STOREY_ID INTEGER NOT NULL,
         ARCHITECTURE BLOB,
         ARCHITECTUREFILENAME VARCHAR2(80),
         CONSTRAINT STOREY_ARCHITECTUREBLOB_PK PRIMARY KEY
(STOREY_ARCHITECTUREBLOB_ID),
         CONSTRAINT STOREY_ARCHITECTUREBLOB_UNQ UNIQUE
(STOREY_ARCHITECTUREBLOB_NO, STOREY_ID),
         CONSTRAINT STOREY_ARCHITECTUREBLOB_FK FOREIGN KEY (STOREY_ID)
REFERENCES STOREY (STOREY_ID) ON DELETE CASCADE);

CREATE TABLE WINDOW
        (WINDOW_ID INTEGER NOT NULL,
        BUILDING_ID INTEGER NOT NULL,
```

```
	FACADE_ID INTEGER  NOT NULL,
	STOREY_ID INTEGER NOT NULL,
	WINDOWTYPE_GLASS VARCHAR(25),
	WINDOWTYPE_CASEMENT VARCHAR(25),
	PERCENTDAMAGE VARCHAR(10),
	PREPOSTDAMAGE VARCHAR(15),
	WINDOW_NO NUMBER(38) NOT NULL,
	CONSTRAINT WINDOW_PKS PRIMARY KEY (WINDOW_ID),
	CONSTRAINT WINDOW_UNQ UNIQUE (BUILDING_ID, FACADE_ID, STOREY_ID),
	CONSTRAINT WINDOW_FK1 FOREIGN KEY (BUILDING_ID) REFERENCES
BUILDING(BUILDING_ID) ON DELETE CASCADE,
	CONSTRAINT WINDOW_FK2 FOREIGN KEY (FACADE_ID) REFERENCES
FACADE(FACADE_ID) ON DELETE CASCADE,
	CONSTRAINT WINDOW_FK3 FOREIGN KEY (STOREY_ID) REFERENCES
STOREY(STOREY_ID) ON DELETE CASCADE);

CREATE TABLE WINDOW_PHOTOBLOB
	(WINDOW_PHOTOBLOB_ID INTEGER NOT NULL,
	 WINDOW_PHOTOBLOB_NO INTEGER NOT NULL,
	 WINDOW_ID INTEGER NOT NULL,
	 PHOTO BLOB,
	 PHOTOFILENAME VARCHAR2(80),
	 CONSTRAINT WINDOW_PHOTOBLOB_PK PRIMARY KEY
(WINDOW_PHOTOBLOB_ID),
	 CONSTRAINT WINDOW_PHOTOBLOB_UNQ UNIQUE (WINDOW_PHOTOBLOB_NO,
WINDOW_ID),
	 CONSTRAINT WINDOW_PHOTOBLOB_FK FOREIGN KEY (WINDOW_ID)
REFERENCES WINDOW (WINDOW_ID) ON DELETE CASCADE);

CREATE TABLE WINDOW_ARCHITECTUREBLOB
	(WINDOW_ARCHITECTUREBLOB_ID INTEGER NOT NULL,
	 WINDOW_ARCHITECTUREBLOB_NO INTEGER NOT NULL,
	 WINDOW_ID INTEGER NOT NULL,
	 ARCHITECTURE BLOB,
	 ARCHITECTUREFILENAME VARCHAR2(80),
	 CONSTRAINT WINDOW_ARCHITECTUREBLOB_PK PRIMARY KEY
(WINDOW_ARCHITECTUREBLOB_ID),
	 CONSTRAINT WINDOW_ARCHITECTUREBLOB_UNQ UNIQUE
(WINDOW_ARCHITECTUREBLOB_NO, WINDOW_ID),
	 CONSTRAINT WINDOW_ARCHITECTUREBLOB_FK FOREIGN KEY (WINDOW_ID)
REFERENCES WINDOW (WINDOW_ID) ON DELETE CASCADE);
```

8.2.2 SQL Code for Creating Sequences

```
create sequence buildingseq increment by 1 start with 1;
create sequence facadeseq increment by 1 start with 1;
create sequence storeyseq increment by 1 start with 1;
create sequence windowseq increment by 1 start with 1;
create sequence seqstudent increment by 1 start with 1;
create sequence seqdataprovider increment by 1 start with 1;
create sequence seqdataformat increment by 1 start with 1;
```

```
create sequence FACADE_PHOTO_SEQ increment by 1 start with 1;
create sequence FACADE_CAD_SEQ increment by 1 start with 1;
create sequence STOREY_PHOTO_SEQ increment by 1 start with 1;
create sequence STOREY_CAD_SEQ increment by 1 start with 1;
create sequence WINDOW_PHOTO_SEQ increment by 1 start with 1;
create sequence WINDOW_CAD_SEQ increment by 1 start with 1;
create sequence BORINGLOG_SEQ increment by 1 start with 1;
```