12-2010

A GIS-Based Building Data Model Storing, Visualizing, and Analyzing Building Information for United States Coast Guard and Environment Systems Research Institute

Eric William Wittner
University of Redlands

University of Redlands

A GIS-Based Building Data Model

Storing, Visualizing, and Analyzing Building Information for
United States Coast Guard
and
Environment Systems Research Institute

A Major Individual Project submitted in partial satisfaction of the requirements
for the degree of Master of Science in Geographic Information Systems

by
Eric William Wittner

Committee in charge:
Karen Kemp, Ph.D., Chair
Douglas Flewelling, Ph.D.

A GIS-Based Building Data Model
Storing, Visualizing, and Analyzing Building Information for
United States Coast Guard
and
Environment Systems Research Institute

The report of Eric William Wittner is approved.

_____

Douglas Flewelling, Ph. D., chair

_____

Karen Kemp, Ph.D.

December 2010

Acknowledgements

Without the support of the following individuals the successful completion of this Major Individual Project would not have been possible.

James J. Dempsey, Lieutenant Commander, United States Coast Guard. James provided the impetus for this entire Major Individual Project. His passionate pursuit of better methods to manage information at the Coast Guard, was what motivated me to pursue using GIS in a domain that I have never worked in before.

Dr. Mike Flaxman, Assistant Professor, Urban Information Systems Group, Department of Planning, Massachusetts Institute Technology. Mike served as my primary mentor in the early stages of my project. He provided direct technical support and assistance in developing methods of converting the data I received from the United States Coast Guard into usable GIS information.

Nathan Shephard and Bill Miller, Esri. Bill and Nathan are my managers at Esri, and they provided me with all manner of support, such as flexible work hours, technical expertise, and constant encouragement. And for this I am grateful.

Karen Kemp, Ph.D. for her unmatched patience, support, and wisdom without which this project would not have been possible.

ABSTRACT

A GIS-Based Building Data Model
Storing, Visualizing, and Analyzing Building Information for
the United States Coast Guard
and
Environmental Systems Research Institute

By Eric William Wittner

As the world's population moves from rural areas to urban centers, managing information in the built environment effectively and efficiently will become critical to sound decision making. The ability to represent, visualize and analyze these urban centers will be at the heart of city and regional planning processes. GIS has traditionally been used as a tool for supporting these processes, but has focused on the exterior two dimensional features of the environment. Increasingly planners are being asked to consider all aspects of the built environment, such as air traffic corridors above the ground, utilities networks below, and the interior and exterior of buildings on the ground , in a comprehensive decision making system. New data models, analysis methods, and visualization techniques will need to be developed to meet these emerging needs.

The purpose of this Major Individual Project (MIP) was to develop a GIS-based building data model for the United States Coast Guard and Environmental Systems Research Institute, Inc. This data model represents not only the exterior of a building, but also its interior features. The features in the data model are designed in such a way as to support spatial analysis, and the project will demonstrate how such analysis can be conducted on building information. This analysis includes both vector and raster methods, and is scalable to work not only within in one building, but between buildings in a site, campus, or city. The data model also supports the simple management, modification, and maintenance of building features, while still providing the complex geometries necessary to visualize the building information in both two and three dimensions.

Table of Contents

Section **Page**

1.0 Introduction 5

1.1 About this document 5

2.0 Project Background 7

2.1 Description of Clients 7
2.1.1 United States Coast Guard 8
2.1.2 Environmental Systems Research Institute, Inc. 8
2.2 Client Needs Assessment 9
2.2.1 USCG 9
2.2.2 Esri 10
2.3 Summary 10

3.0 Building Data Model 11

3.1 Study of Existing Building Data Models 11
3.2 Data Model Requirements and Data Needs 13
3.3 Data Inventory 15
3.4 Conceptual Database Design 17
3.5 Populating the Source Data Model 18
3.6 Generating the In-Building Transportation Network 21
3.6.1 Study of Existing Methodologies 21
3.6.2 Proposed Solutions 23
3.6.3 Selected Solution 28
3.6.4 Network Generation Method 28
3.7 Summary 31

4.0 Editing and Analyzing Building Data 33

4.1 Editing the Data Model 33
4.2 Querying Non-GIS Data Source 34
4.3 Support for analysis 35
4.3.1 Visualizing Building Information in three dimensions 40
4.4 Route Generation on the Network Data Model 42
4.5 Summary 43

Section **Page**

5.0 Conclusion 45

5.1 Faults and Flaws 45
5.2 Next Steps 47
5.2.1 Multipatch Building Exterior 47
5.2.2 Annotation and Labeling 47
5.2.3 Serving Building Data as a Globe Service 47
5.2.4 Exterior Landscape Features 47
5.2.5 New Derivative Analysis Models 47
5.2.6 Revised Network Generation 48
5.3 Summary 50

References 51

Appendix A—Building Data Model 55

Appendix B—ArchiCAD Extension 67

Appendix C—Network Generation Scripts 77

Appendix D—Vulnerability model 85

Appendix E—Hazard model 87

Appendix F—Accessibility model 89

List of Figures

Figure	Page
Figure 3-1	15
Figure 3-2	16
Figure 3-3	17
Figure 3-4	18
Figure 3-5	19
Figure 3-6	20
Figure 3-7	22
Figure 3-8	23
Figure 3-9	25
Figure 3-10	26
Figure 3-11	28
Figure 3-12	31
Figure 4-1	34
Figure 4-2	35
Figure 4-3	36
Figure 4-4	37
Figure 4-5	38
Figure 4-6	38
Figure 4-7	39
Figure 4-8	40
Figure 4-9	41
Figure 4-10	42
Figure 4-11	43
Figure 5-1	46
Figure 5-2	49
Figure 5-3	50

1.0 Introduction

How can geographic information systems (GIS) be used to represent built features in an urban environment? As the world's population moves from rural areas into larger metropolitan regions, the need for analysis on the built environment is going to grow. Today, organizations that are responsible for managing a large number of buildings, such as multi-national corporations, government agencies, and various branches of the armed forces, are demanding an increased ability from GIS to help manage their facilities. They want to be able to represent the basic components of their buildings, track their assets (equipment, furniture, and personnel) as they move through them, and conduct spatial analysis for code compliance, emergency response, way finding (routing), and space planning. The purpose of this project is to provide an example of how buildings can be represented in a GIS and how analysis on those buildings can be conducted. The goal is to provide a prototype building data model design and to demonstrate how to do analysis on a building.

1.1 About this document

The document details the research, design, data collection, analysis, and assessment involved in modeling buildings in a GIS. It is laid out in seven sections, an introduction to the project, the background of the project, the process involved in building and collecting the necessary data, and then the analysis conducted to verify the usefulness of the data, the conclusion from this process, then supporting material and references, followed by the appendix of material related to the project.

- **Section 1—Introduction**
  This section details what this project is about, and what each section in the project covers.

- **Section 2—Project Background**
  This section details the clients that provided the impetus and drove the needs for this project. It provides background about each client, the needs specific to each client, and a summary of the requirements synthesized from the user needs assessment.

- **Section 3—Building Data Model**
  This section details the research on existing building data models and standards conducted to develop a comprehensive understanding of how to represent a building. It then presents the conceptual data model, an inventory of the data collected for the project, and the methods used to combine them into a working data model. It also highlights the process of generating derivative data from source data, such as analysis, visualization, and transportation modeling data sets.

- **Section 4—Editing and Analyzing Building Data**
  This portion of the project details the editing and analysis tasks conducted with the

data model to demonstrate and prove its capabilities, and insure that it met the needs of the clients.

- **Section 5—Conclusion**
  The summary of what the project succeeded in accomplishing, any flaws exposed during the process, and what the next steps are.

2.0 Project Background

The world of building construction and management is in the midst of a revolution. New technological innovations are pushing architects and engineers to embrace three-dimensional design, using emerging Building Information Modeling (BIM) techniques. BIMs allow object oriented design of buildings, where each object has a set of attributes and parameters which dictate how it behaves and relates to other objects within the BIM. Manufacturers are beginning to produce BIM objects of the components they produce so architects and designers can drag these objects into their BIM model and use them directly. The BIM can be used to manage the construction of a building by attaching a fourth dimension to objects that shows the order of installation/construction of each object. Once a building is completed, tabular data can be attached to the objects that compose the model, allowing the BIM to be harnessed for use in facilities management; the BIM then tracks the changes to the building over time.

It is this rich collection of objects that makes the BIM well suited for use as a data source in GIS. GIS has the capability to conduct analysis functions that do not exist in most Computer-Aided Drafting (CAD) software packages or BIMs. By taking selective sets of objects out of the BIM and bringing them into a GIS, we enable facility and building managers to ask questions of their data that they could never pose before. Additionally, bringing the building information into a GIS can provide managers an easy and simple way to distribute their data to clients and decision makers across their organization because GIS is inherently scalable and easily delivered over the World-Wide Web.

The United States Coast Guard is interested in the use of GIS technology to help manage their existing building information across their entire organization. The United States Coast Guard, Civil Engineering Unit (CEU), 11th District has been on the cutting edge of this effort and has asked the Environmental Systems Research Institute, Inc. (Esri) to assist in developing a demonstration of their technology. Esri has lacked the time, staff, and expertise to effectively pursue this research. It was decided to ask an academic institution for assistance in conducting this research and development.

2.1 Description of Clients

The focus of this project is on developing a generic data model and analysis methods usable on any building. This was done in order to take into account the needs of building and facilities managers across the country, who have been searching for the right tools and methods to help manage a dizzying variety of facilities. This project addresses the needs of two primary clients: United States Coast Guard and Environmental Systems Research Institute, Inc.

2.1.1 United States Coast Guard

The United States Coast Guard (USCG) is the smallest of the seven uniformed services of the United States. USCG has a broad and important role in homeland security, law enforcement, search and rescue, marine environmental protection, and the maintenance of river, intra-coastal and offshore aids to navigation. It also lays claim to being the oldest continuous seagoing service of the nation. The Coast Guard's motto is *Semper Paratus*, meaning "Always Ready." Its stated mission is to protect the public, the environment, and the United States' economic and security interests in any maritime region including international waters and America's coasts, ports, and inland waterways. USCG has about 40,150 men and women on active duty, who help manage and maintain a variety of facilities allover the United States, and has an extensive inventory of boats, ships, helicopters, and airplanes. The USCG Eleventh District covers the states of California, Arizona, Nevada, and Utah, the coastal and offshore waters including the offshore waters of Mexico and Central America down to South America. Its units are located throughout the state of California with its headquarters located on Coast Guard Island in Alameda, California.

In 2006, USCG Eleventh District units responded to 3,048 search and rescue cases, saving over 525 lives and 9.3 million dollars in property. It conducted 5,962 commercial vessel inspections. USCG and U.S. Navy assets under the tactical control of the district prosecuted 35 major counter-drug cases in the eastern Pacific Ocean, interdicting almost 187,907 pounds of cocaine representing 61.5 percent of the USCG total for that year. The USCG Eleventh District alone seized 149,491 pounds of cocaine in 2006.

The USCG Eleventh District Civil Engineering Unit (CEU) is responsible for the construction, maintenance, and dismantling of all shore-based facilities within the district. These facilities include buildings, security structures, roads, docks, runways, helicopter landing pads, and a variety of utilities such as potable water, grey water, storm water, sewage, power, gas, aviation fuel, telecom, and communication services. These facilities are critical to the USCG's mission, and must be maintained in top working order. Additionally, the CEU must facilitate the evolution of these facilities to meet the logistical needs of new and emerging mission requirements, especially in the post-911 world.

2.1.2 Environmental Systems Research Institute, Inc.

Environmental Systems Research Institute, Inc. (Esri) was founded in 1969 as a consulting firm that specialized in land-use analysis projects. The early mission of Esri focused on the principles of organizing geographic information in support of planning and decision making. Esri planning projects included redevelopment plans for the City of Baltimore, Maryland, and assisting Mobil Oil in site selection for the town of Reston, Virginia. During the 1980s, Esri devoted its resource to developing a core set of application tools that could be applied in a computer environment to create a GIS. This is what is known today as GIS technology. In 1982, Esri launched its first commercial GIS software called Arc/INFO, which combined the display of geographic features with a database management tool.

Esri evolved from a consulting firm to a large organization dedicated to GIS research and GIS software development. Esri's global presence grow with the release of ArcView, an affordable desktop GIS application, which shipped an unprecedented 10,000 copies in the first six months of 1992. Additionally, Esri expanded into the data distribution and publishing business, eventually creating the Geography Network (a collaborative system for publishing, sharing, and using geographic information), and geographic analysis methods over the internet. In the late 1990s, Esri embarked on an ambitious research project to reengineer all of its GIS software as a series of COM objects; and in 1999, ArcInfo 8 was released. It was the beginning of a new family of software, built on industry standards providing intuitive and easy to use tools right out of the box that were scalable across the enterprise. Today, Esri's GIS products are poised for even greater growth with continued expansion in computing power and the wide-spread adoption of high-speed internet technology.

Esri employs more than 4,000 staff, more than 1,600 of whom are based at the company's world headquarters in Redlands, California, and the rest at 10 regional offices in the United States. The Redlands campus has been expanding continuously since the mid-1990s as Esri grows to meet the needs of its clients. Additionally, Esri works with more than 80 international distributors who support customers in more than 200 counties. Esri is staffed by a diverse work force of experts with valuable experience and technical knowledge dedicated to the communication of geographic information and the success of their users.

2.2 Client Needs Assessment

2.2.1 USCG

USCG owns over 30 million square feet of building space, and leases an additional 3 million. It owns and operates 7,930 buildings at 656 locations, and controls 65,000 acres of land. With over 40,000 personnel and an annual operating budget of over 7 billion dollars, managing the Coast Guard's shore facilities is comparable to managing a large multi-national company or a small government (Dempsey, J.J. & Hammond, D., 2002). The complexity of these management tasks elevates the importance of sound strategic planning and decision making because of the aggregated cost and the impacts mismanagement can have on operational capabilities. These challenges have motivated the Coast Guard Civil Engineer Program to re-evaluate shore facilities management techniques, institute changes to these practices, and adopt enabling technology to better meet their mission requirements.

One of the technological enablers that the Coast Guard has turned to is Geographic Information Systems. GIS is already utilized extensively in managing search and rescue, security, and environmental protection operations, as well as in managing the landscape of Coast Guard facilities. However the management of building construction, space planning, move management, space utilization analysis, mission dependency/logistical readiness assessment, is conducted using CAD programs and custom-built third-party applications. The limitations of this current system are two-fold: (1) CAD systems are generally restricted to

storing information on a building-by-building basis, prohibiting assessment across a site, region, or district; and (2) the building information cannot be easily integrated with existing GIS information for the rest of the facilities. The CEU desires an integrated system where they can easily do analysis across their entire region, including the interior and exterior of buildings and all aspects of the built environment including utility networks. Their goals are threefold: (1) to represent the components of a building necessary to support analysis within an enterprise GIS system; (2) to find a way to simply and effectively keep those components up to date as buildings and other facilities go through rehabilitation and redesign; and (3) to automate costly manual assessment techniques using existing models and scripting frameworks in GIS.

2.2.2 Esri

Esri's present focus is on developing software applications and analysis methods to meet the needs of its users. By 2007, over half the world's population will be in urban areas. This means that Esri customers will need to manage their information in the built environment. This environment will be three-dimensional, requiring analysis across, above, and below the earth's surface. Esri's users will need to be able to integrate large-scale data, such as detailed drawing plans, with small-scale city information in a seamless database that supports analysis. Esri is currently researching the needs and requirements of their users to identify what published best practices and advancements in software development they will need to make in order to meet these new and emerging challenges. Part of that research effort is a series of demonstrations or prototype projects focused on the types of analysis that building and facility managers need to conduct.

Esri is in a constant state of expansion, as they add new personnel to meet the needs of their users. It was decided that one of the pilot projects should focus on meeting the internal needs of Esri managers to help plan moves between new and old buildings on campus, as well as demonstrate some of the way finding capabilities of GIS inside of a building.

2.3 Summary

The goal of this project is to create a data model that is capable of storing building information from CAD and BIM sources and will support building and facilities management business processes. These processes as defined by the user needs assessment of the USCG and Esri are asset management, space planning, space utilization assessment, and two- and three-dimensional visualization of building attributes. In addition, the data model needs to support spatial analysis in general, as required by the clients, to support needs not yet identified by those organizations. The data model needs to be scalable, working not only at the building level but also across campuses, regions, nations, and the globe. However, this project will focus on modeling, editing, and analysis at the building scale.The next step in the process is to research existing building data modeling efforts and standards related to buildings in general.

3.0 Building Data Model

In order to work with buildings in GIS the system must be able to represent them. This includes being able to store the basic geometries representing their components, store attributes about those geometries, and store relationships between those geometries. The challenge is in the semantic translation of building features into the GIS context. This involves breaking the hierarchical nature of the BIM data into collection of features, generalizing them as necessary to reduce overhead, as well as maintaining a link back to their properties (Karimi, H.A. & Akinci B., 2009). The stored representation must be suitable for display in two dimensions, for use in editing and for map or plan generation. These representations must also be viewable and navigable in three dimensions, for use in simulation modeling and three dimensional mapping. In addition these representations must be designed, stored, attributed, and related in such a way as to support various analysis methods related to the business processes of a building.

The purpose of a data model is to provide a practical template for working with data, within a GIS, for a specific domain (Esri, 2007, Data Model Introduction). Having a common frame work supports the implementation of GIS projects, simplifies data loading and extraction, and helps support integration with existing standards. These models differ from standards in that they provide a rough framework that can be customized (simplified or expanded) to meet a specific need.

This data model focuses on representing the most basic components of a building, both its interior and exterior. These components must support the basic business process involved in facilities and building management. This includes asset inventory, such as tracking the position and movement of equipment or personnel between different rooms within the building. The components must support thematic mapping in two and three dimensions, representing a variety of attributes. For example, the United States Coast Guard's mission dependency index (MDI), facilities condition index (FCI), and space utilization index (SUI), describe various characteristics of rooms within a United States Coast Guard facility (Dempsey, J.J., 2003). In addition these components must be usable by existing GIS based spatial analysis tools, so that new attributes, features, and processes can be modeled as needed.

3.1 Study of Existing Building Data Models

Three potential examples of building data models were identified: (1) the International Alliance for Interoperability's (IAI) Industry Foundation Classes (IFCs), (2) Penobscot Bay Media's GIS Data Model for Interior Spaces, and (3) CityGML, an open XML standard for the representation of cities, all of which are discussed in more detail later. Additionally, a variety of standards were found that apply to buildings such as standards for the calculation of the interior spaces of buildings or parametric design of steel structures. These represented methods for designing buildings, managing information about buildings, or managing the construction of buildings, but did not prove comprehensive enough for consideration as data models. Two of the space standards stood out for consideration as a guideline for how to represent rooms or space within a building. The American National Standards Institute and

Building Owners and Managers Association International Z65.1-1996 floor area measurement standard (BOMA) and the Working Group on Postsecondary Physical Facilities' Postsecondary Education Facilities Inventory and Classification Manual (FCIM).

Industry Foundation Classes (IFCs) are definitions for each component in a building data model designed specifically to support the transfer of building information between building project participants. (International Alliance of Interoperability, 2007, Industry Foundation Classes FAQ) IFC's are currently under development by the International Alliance for Interoperability (IAI). The IAI's focus is on insuring that whole, complete, thorough, and accurate building data is transferred from participant to participant through the whole lifecycle of the building. IFCs are data elements that represent parts of the building or elements of the process relevant to those parts. The focus is on rigorous hierarchy of these data elements and their relationship to (and interactions on) each other. It requires the sequential addition and maintenance of these objects as the construction, maintenance, and re-purposing of a building commences. However, IFCs are new to the scene of building management and in their infancy when it comes to adoption by large CAD companies. They are being used primarily as a data transfer standard for one-time snapshots of existing information systems, rather than as a home for tracking the workflow of facilities management. Ultimately, the hierarchical nature of the data and the lack of specifications on the geometry needed to represent base components rendered this data model unusable in this project. As IFCs progress and tighter definitions of IFC standards are achieved through adoption by manufacturers and builders, it will become easier to convert IFC objects into GIS features. As the GIS community moves to support IFCs as an import format for creating geographic features, IFCs may prove useful in GIS-based building data modeling.

Penobscot Bay Media (PenBay), a small professional services firm that specializes in information visualization solutions using GIS, has been developing a GIS data model for the interior spaces of buildings. This data model is being designed to support their clients' needs to asses and visualize existing built spaces. They must be able to answer the simple space planning questions such as how much space exists, who occupies each space, what departments are assigned to a space, who pays for a given space, and how efficiently are the spaces used. These questions cannot be answered easily from existing building data models in CAD or BIM software packages. PenBay's data model focuses on a collection of complex geometries representing simple components of a building that can answer space-related questions, according to a variety of existing space calculation standards. These standards include the FICM, BOMA, and any combination of geometries required by the client. Although these complex geometries can be simplified for purposes of spatial analysis and visualization, they do not easily support editing and scenario building on the interior spaces of buildings. For this reason, it was decided to develop a separate building data model that supports these editing requirements.

CityGML is an XML derived visualization standard for urban areas, focused on being able to model whole cities at multiple levels of detail, while still providing the frame work for analytical queries through a structured data hierarchy. The goal is to develop a common information model representing all 3D urban objects, from building to street furniture such as

lamp posts, benches, and signs (Kolbe T. & Bacharach S., 2006). CityGML defines the objects and relations for most objects in a city. This includes the semantic structure of a city, the topographic surface of the city, the relationships between objects (hierarchy and ownership), and the properties that govern their appearance. CityGML is an open data model, implemented as an application schema of Geographic Markup Language developed by the Open Geospatial Consortium. Although promising as a method for supporting three dimensional visualization and storage of buildings for city visualization purposes, it lacks the necessary detail for use as a building data model.

FICM and BOMA represent two different standards for calculating how much space is in a building. It establishes what portions of a room should be measured in order to get its square footage. How that is determined varies depending on the standard, and where the room is positioned in the building. Sometimes the measurements are conducted from the center lines of the room's walls, sometimes from the interior surface of the wall. The door can be included or excluded from this calculation, as well as the window sill. Ultimately what these standards dictate is that any data model will need the capacity to dynamically calculate the square footage based on different combinations of geometry, or different representations of the various components of a building.

3.2 Data Model Requirements and Data Needs

The data model must provide a means of relating data from external sources, such as tables or relational databases, to the various components of a building. It must support the storage of basic information about each component that is the attributes required in almost all types of analysis. The geometry representing each component must be simple enough to be easily edited, maintained, and updated. However the model must support the generation of more complex features from these basic geometries, as needed. Through the review of existing data models, space standards, CAD packages, pre-existing building datasets, and various periodicals, a core list of building features was identified as necessary to represent a building. They include the building's footprint, floors, rooms, walls, doors, hallways, and ceilings.

Building footprints are needed to provide the basic geometry to represent the building. In two dimensions, this represents the extent of the building at its base, and can be extruded from its base height to form a simple three-dimensional representation of the building. These polygons provide the foundation on which to attach information about the building such as the owner, operator, date of construction, total square footage, and fields necessary for data management (e.g., a unique building identifier or an elevation identifier) to relate it to a table of separate elevation measurements.

The floors of a building represent another basic component of a building. Since the majority of spatial analysis in GIS can only be conducted in two dimensions, floors represent the analysis surface on which most of the spatial analysis will take place. They form the extent or boundary of any analysis conducted. They need to store attributes as to which floor they are, be that with a unique floor identification field or with a generic floor number. Like footprints,

floors need an elevation identifier to relate them to a separate table of elevation data for use in three-dimensional visualization.

Rooms represent the most important geometry in the interior of a building. They are the feature to which the most important information is associated, thus a unique room identification number is essential. Additionally, they may store a room type, a square footage, and other information such as a series of identification codes based on the equipment stored in the room. Square footage may not be based upon the actual geometry of the room but some other calculation method such as FICM or BOMA. Rooms need an elevation identifier to relate them to a separate table of elevation data for use in three-dimensional visualization

Walls are the most critical feature when it comes to conducting way finding within the confines of a building. They define the primary barrier to movement through a building. Additionally, different types of analysis may require different types of information about a wall. For instance an internal fire model for a building may require information about the type and material of a wall. For this project, walls can be represented as either the centerlines of walls with an attribute representing their thickness, or as polygons representing the actual thickness of walls. They need an attribute indicating what type of wall they are, interior or exterior, for vulnerability analysis purposes. Additionally, walls need an elevation identifier to relate them to the common table of elevation data for use in three-dimensional visualization.

Doors can be critical to certain way finding applications, especially if evacuation time estimates are required. It may be necessary to add certain amounts of travel time or restrict the passage of evacuees through doors as represented as nodes on a transportation network. They can be most easily represented as lines, with widths as an attribute. A unique identifier for each door allows the assignment of custom impedance values for each door along the route, or if the door has a type field impedance could be calculated based on the door type. Additionally to these other attributes, each door needs an elevation identifier to relate them to the common table of elevation data for use in three-dimensional visualization.

Hallways may be necessary for way finding and raster analysis processes. Hallways may be treated differently than rooms within a transportation network, since they are edges to travel along rather than destinations to reach. However, it is easy to generate hallways on the fly so they do not need to be persisted in the data model. Simply subtract the rooms, walls, and doors from the floor polygon and one is left with hallways. Use the elevation identifier from the source floor.

Ceiling data may be desirable for three-dimensional visualization purposes when one is viewing the interior of the building. Some rooms have false ceilings, so representing the next floor as the top of the room may be inaccurate. The ceilings of rooms and hallways can be generated on the fly as needed by combining rooms and hallways. Set the height of each features based on the attribute representing its base elevation add to the attribute representing its extrusion height. The combination of the two should reflect the height of the false ceiling.

For visualization purposes it may be desirable to include all kinds of building furniture such as vents, light fixtures, light switches, fire extinguishers, etc. These features can be easily represented as points, with attached images, and be added as desired at a later date. However each feature will also require an attribute representing its elevation if they are to be displayed in 3D.

3.3 Data Inventory

USCG delivered 35 buildings in Graphisoft's plan data format for their BIM software produced by Graphisoft called ArchiCAD (See Figure 3-1). These building plans were highly detailed showing not only the general components of the building such as windows, doors, and walls, but also detailed structural components such as beams within walls, the components of a window's frame, and detailed heating and air conditioning components (See Figure 3-2). The plans were constructed from pre-existing two-dimensional CAD software such as AutoDesk's AutoCAD, and Bentley's Microstation, along with the digitization of hardcopy architectural plans by contractors in support of the Coast Guard's facility management efforts. All features in each building plan were referenced to their own Cartesian coordinate system, and were not geo-referenced. The plans provided had the sensitive information about specific rooms scrambled or removed as a security precaution.

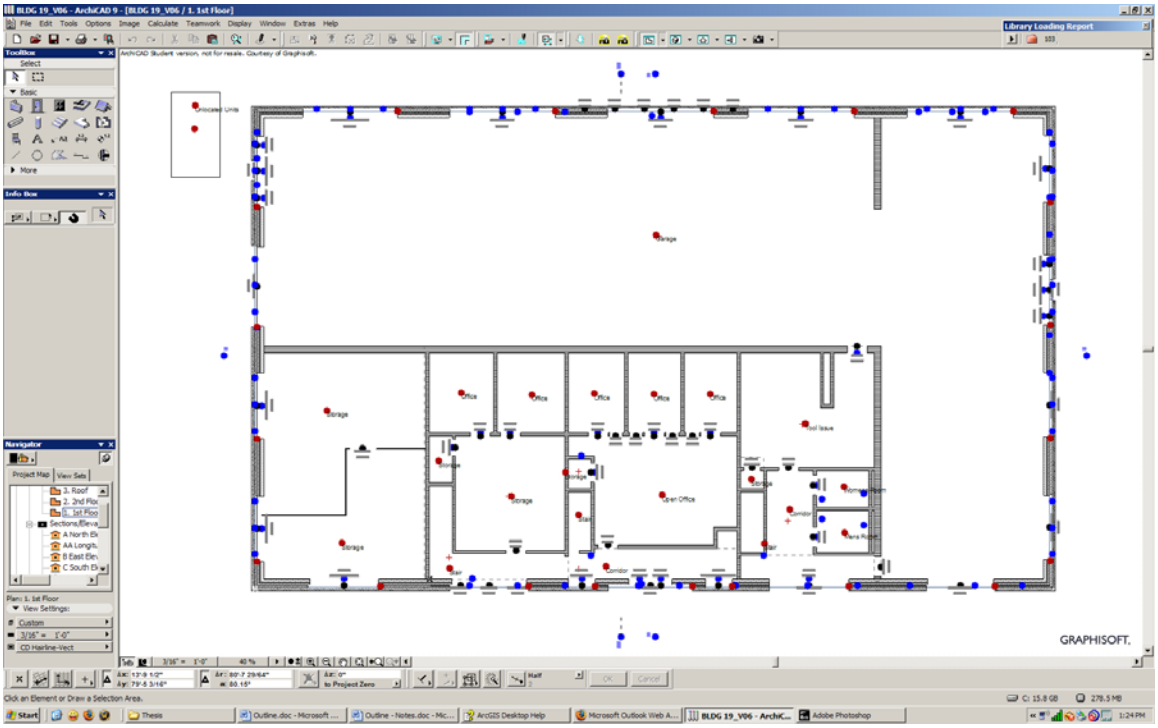


Figure 3-1: *ArchiCAD plan view of building 19*

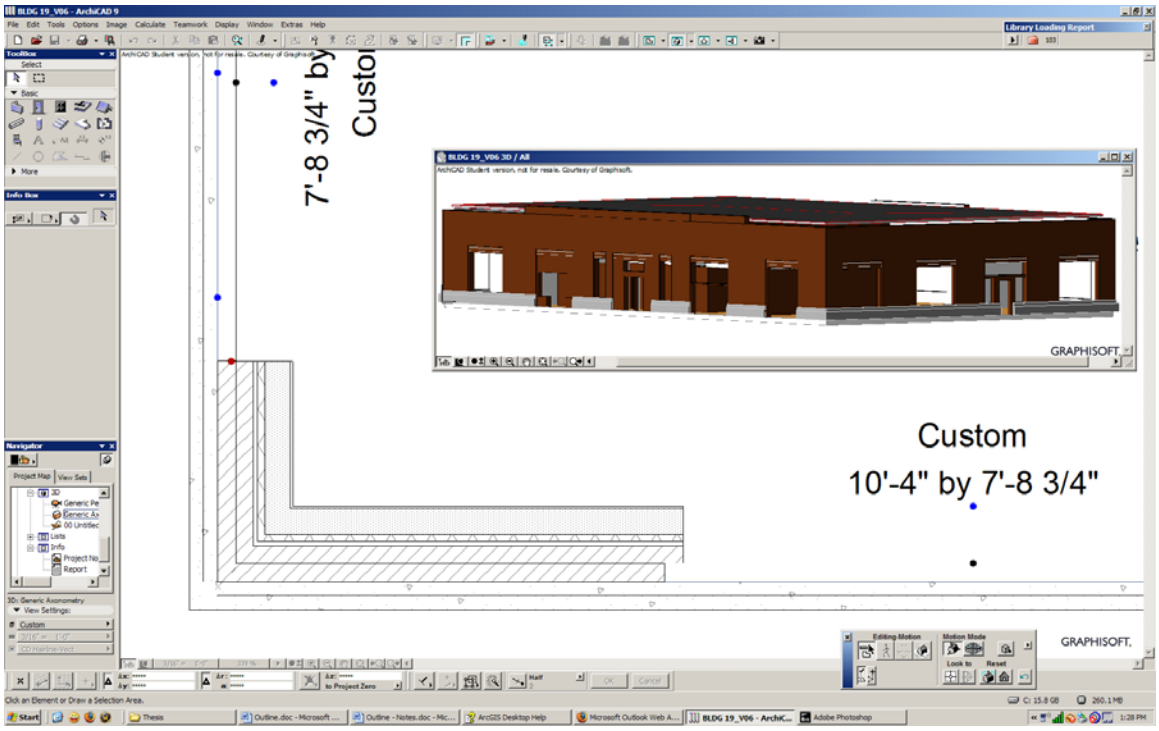


Figure 3-2: *ArchiCAD view of wall detail in building 19.*

Esri provided all the components of their building on campus as feature classes containing the rooms and walls of each building. See Figure 3-3 for an example of what the source data looked like. The various floors of each building were offset geographically and were not perfectly aligned.

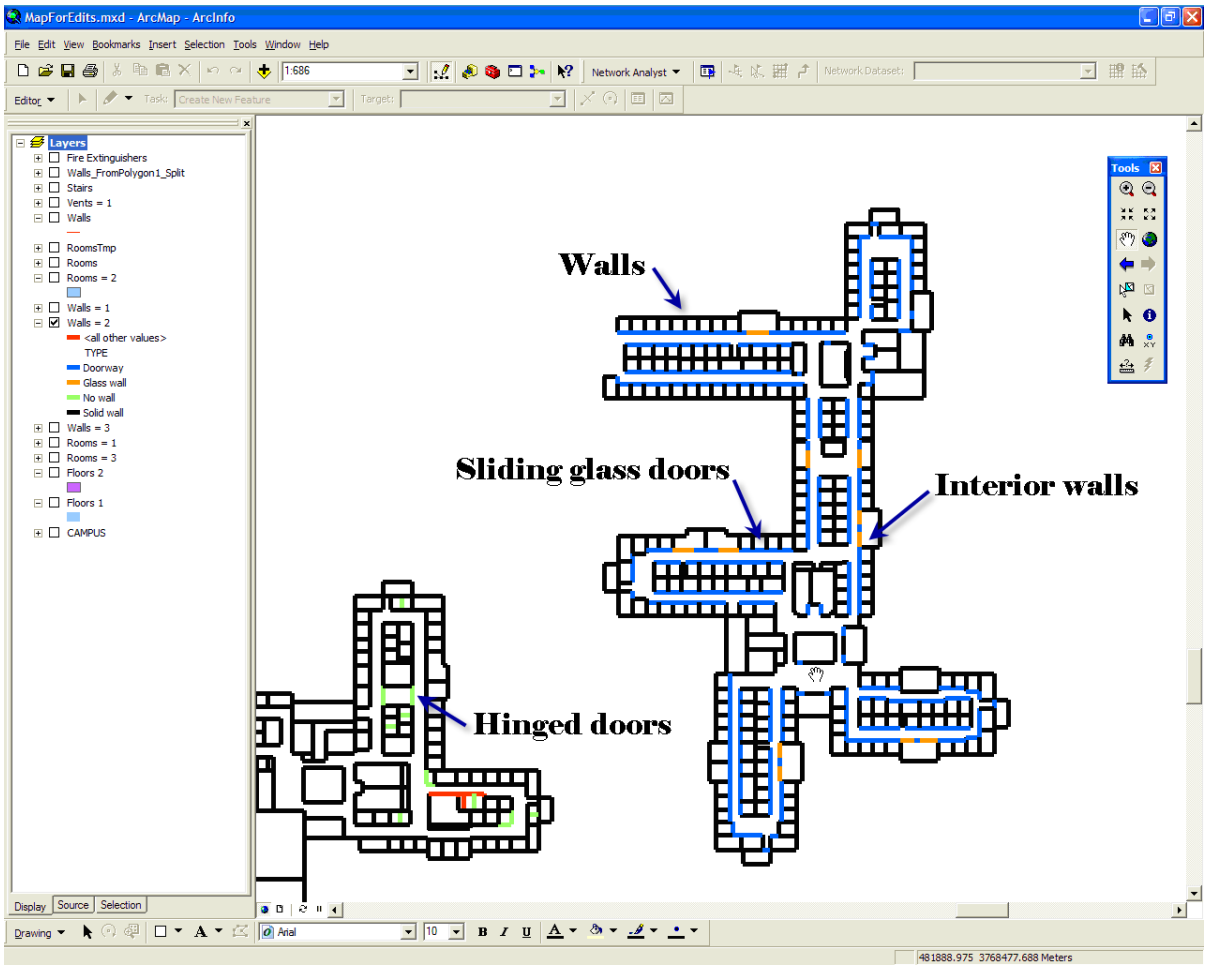


Figure 3-3: *ArcMap view of buildings M and N*

3.4 Conceptual Database Design

The data model was composed of three primary components, the source data model, the derivative visualization and analysis data model, and the transportation data model. The source data model provides the basic features that represent the building. These basic features can be used to generate different types of derivative features for specific purposes. A set of geoprocessing scripts was developed to automatically generate a visualization and analysis data model. The source data model however is flexible enough to allow the development of custom derivative data models, for use in a variety of applications or in support of specific data standards. The transportation data model stores the necessary data layers to conduct way finding across a building. Figure 3-4 shows how the different features in the data models relate to each other. See Appendix A for more details on the data model.

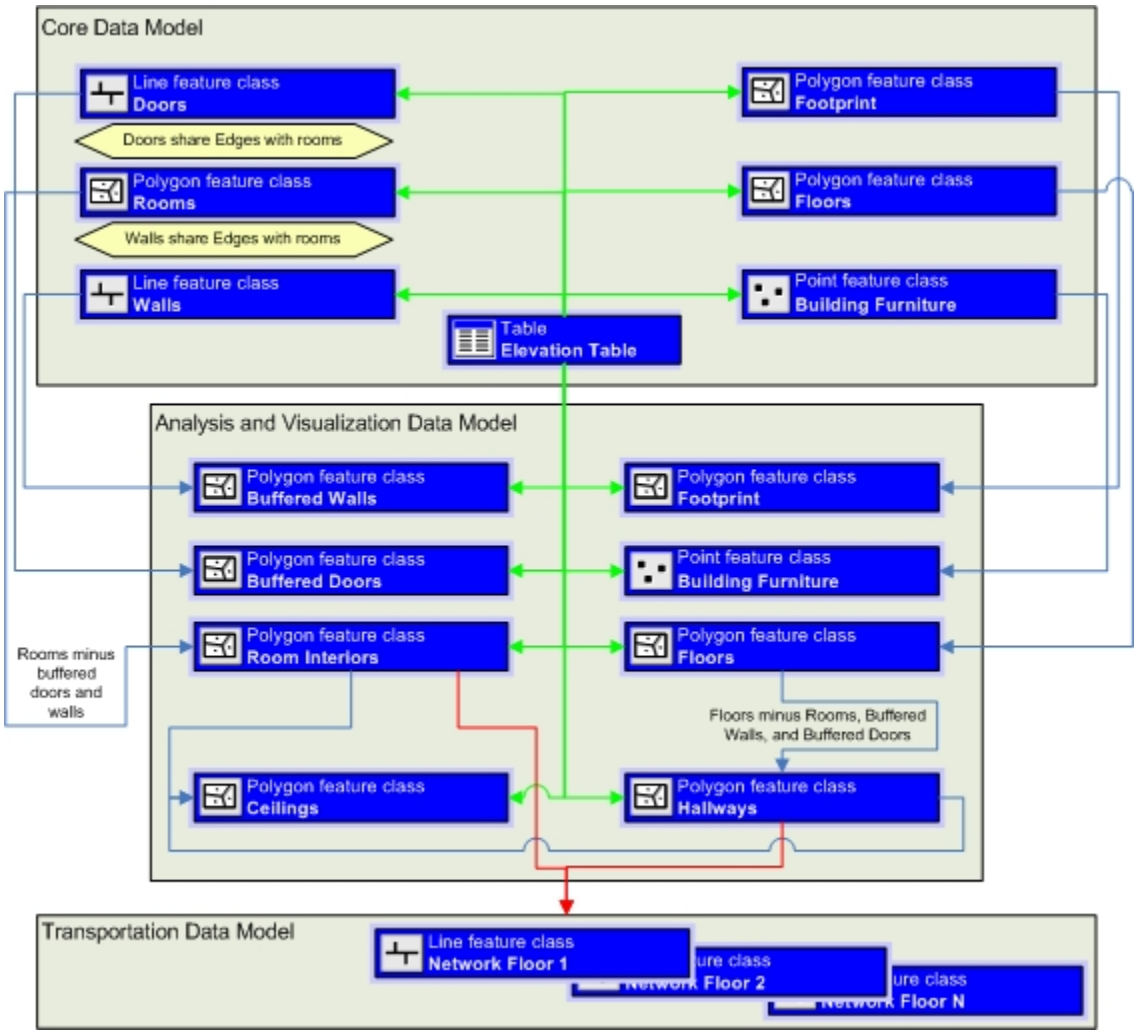


Figure 3-4: *Entity relationships in the conceptual data model.*

3.5 Populating the Source Data Model

Graphisoft provided a student version of ArchiCAD to assist in the conversion of the UCSG buildings into a data format usable by Esri products. An extension to ArchiCAD was developed (with Esri's assistance) to export the basic features of a building out of plan format as three-dimensional multipatches in an Esri shapefile. See Appendix B for more information about the extension. This method still suffered from a loss of feature properties, represented in the BIM, during importation. (Johansson M., & Roupe M., 2010) More extensive development would be required to support a loss-less importation of BIM features to GIS. To learn more about this custom ArchiCAD extension, see Appendix B. Two buildings were selected for conversion, buildings 1 and 19. One can see a three dimensional view of the resulting multipatch features for building 19 in Figure 3-5.

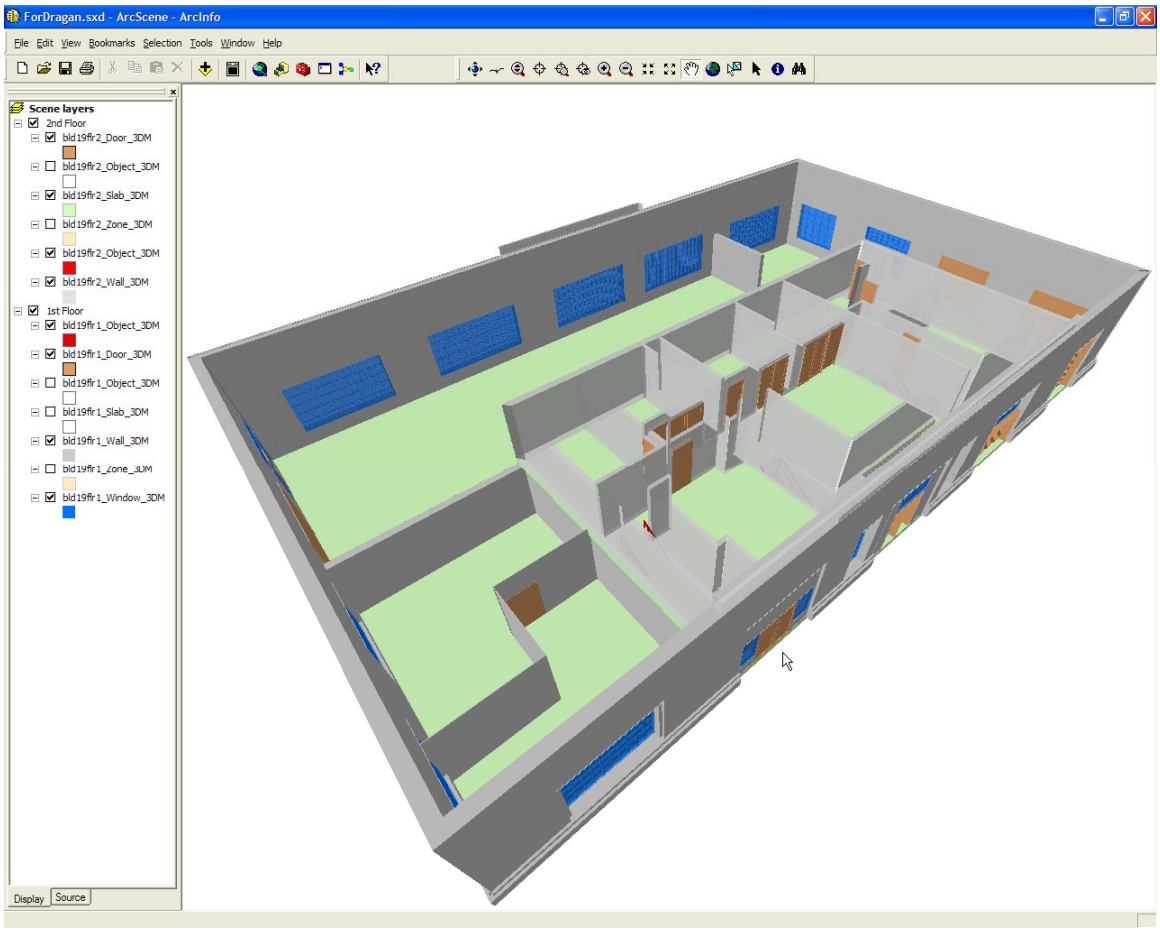


Figure 3-5: *ArcScene view of building 19*

The multipatches for these buildings were imported into an ArcGIS personal geodatabase, and then generalized into polygon feature classes using the Multipatch Footprint geoprocessing tool in the 3D Analyst extension. Figure 3-6 represents a view of those two dimensional features in ArcMap.

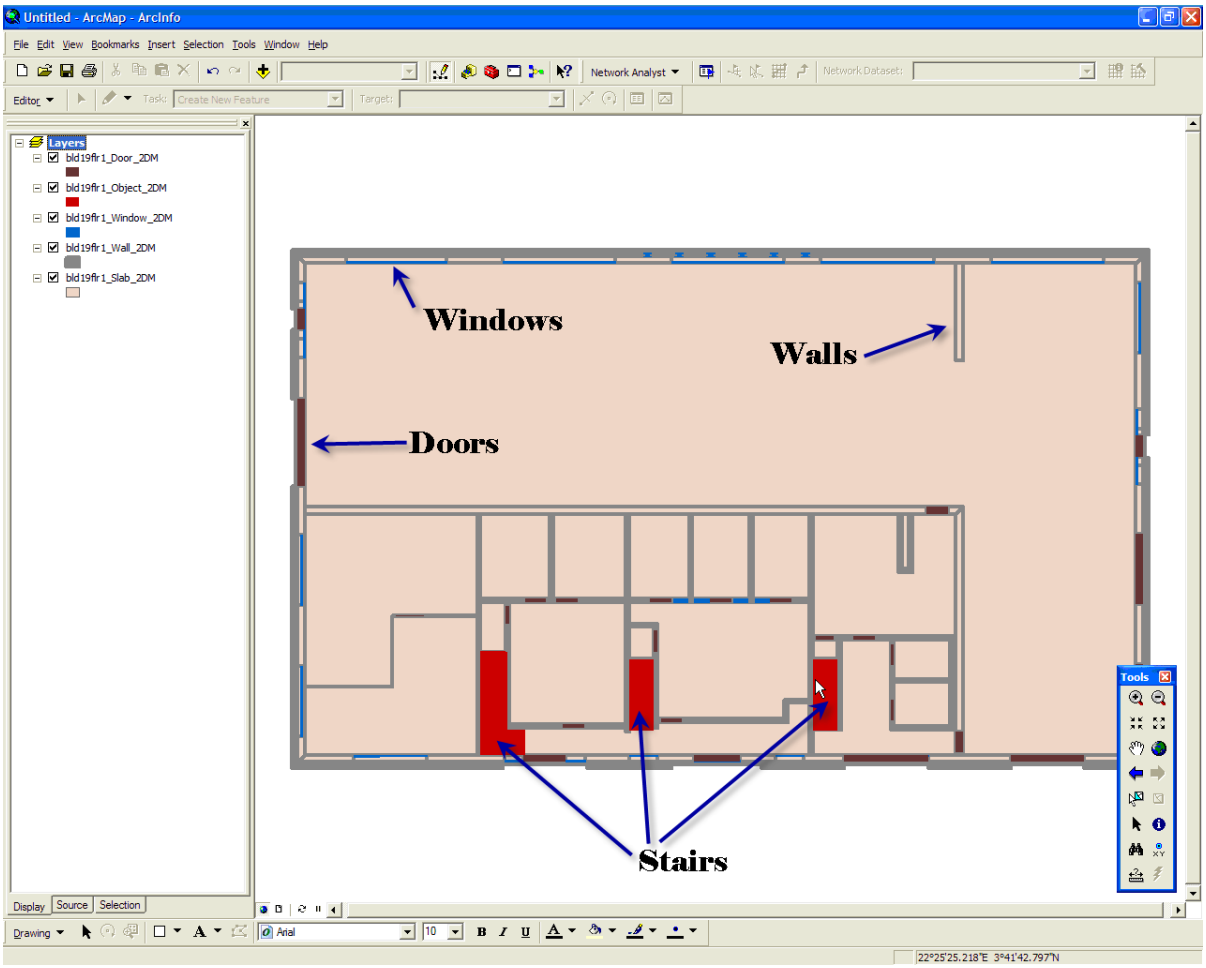


Figure 3-6: *ArcMap view of building 19*

A set of control points were developed to project the two-dimensional representation of the buildings into geographic space. No easy method was found to convert the polygons representing the walls of the building to wall center lines, for the United States Coast Guard data it was decided to leave them as polygons. As polygons they would be readily usable in analysis and visualization, but harder to edit. A routine to skeletonize polygons would need to be developed to handle importation of BIM data into the data model, since most BIM files output walls as multipatch features which can most easily be converted to polygons. CAD data however commonly uses multiple lines, representing the exterior of the walls. These lines would have to be converted to polygons, and then a skeletonizing routine run on the result. It should be noted that this is not the only way to populate a GIS based building data model. Many companies are developing new technology to generate building interior geometry from Light Detect and Ranging (LIDAR) scans. This method alleviates the need to collect, scan, or transfer CAD files all together, and works best for buildings built in the past that do not have CAD or BIM information available. (Lynch, 2010)

3.6 Generating the In-Building Transportation Network

There were no pre-established methods for creating a transportation network in the interior of a building. That means as part of this project a method for populating the transportation data model would need to be developed. Existing methods were researched, a solution was selected, and then implemented.

3.6.1 Study of Existing Methodologies

There are a variety of existing examples of how to conduct building visualization and way finding, but very few that combine the two. Only one reference (Kojo 2006) was found regarding conducting way finding from the interior to the exterior of buildings that was comprehensive enough for consideration. Kyushu University of Japan has been developing an exterior/interior way finding application utilizing both raster and vector analysis. This will be discussed in more detail later. However the efforts of other organizations to represent buildings, and generate routes through those buildings, can still provide useful insight and guidance.

Esri has developed an internal tool for manual way finding called the Esri Campus Viewer. It allows Esri staff to browse a series of web enabled two-dimensional maps of the Esri campus. These maps display some of the basic features of a building, specifically rooms and walls. Users can find the location of specific people, floors, conference rooms, or offices within a building, and display them graphically. However in order to find their way from one location to another they have to manually pan through the map and look at their path. There is no automated route generation capability built into the website. Additionally way finding is limited by the fact that the map system does not handle multistory buildings well. The method chosen for representing buildings with multiple floors is to offset additional floors in geographic space. That means you can pan to a place on the map and see all the levels of a building side by side. This makes it challenging and difficult to find the correct stairway or elevator in complex buildings, and precludes the use of these internal building maps in way finding between buildings since they are not positioned in the correct geographic location.

Sears, Roebuck and Company manages a large fleet of vehicles, over 1,000 delivery trucks and 12,500 repair service vehicles across the United States. (Weigel, D. & Cao, B., 1999) Historically they have relied on a tabular database system to generate delivery and service routes on a daily basis. In 1998 they recognized that by utilizing GIS based network technology they could generate more efficient routes. These routes would result in significant cost savings and more accurate estimation of service times for their customers. Sears worked with Esri to develop this new GIS based routing system. The new system automated the geo-coding of addresses to more quickly determine the location of deliveries and service calls. It then used route optimization based on the number of available delivery trucks and service vans to find the best routes for each day. It calculated large numbers of routes and alternate routes for each van on the fly in real time, allowing resources to be redirected as needed. The result was a drastic increase in efficiency, saving Sears a significant amount of money, and the ability to more reliably predict delivery times and service calls for Sears' customers.

The Graduate School of Design at Kyushu University in Japan has been researching how to utilize GIS in the management of information across their campus. (Kojo 2006) The University is involved in a large expansion project, growing to meet the needs of its students. This growth involves the addition of a new buildings and the remodeling of existing structures. Given the tight schedule of construction and the rapid pace at which this development will take place, they have an immediate need to do move management and planning across their campus. This involves assessing the location and space utilization of various departments and groups within the campus; determining the optimum configuration of new space as it comes online and of temporary space as existing structures are rehabilitated; and providing way finding technology to help students find faculty and resources that have moved across the campus during this process. One of their applications was a way finding pilot project that blended raster least cost path analysis and transportation networks. They utilized raster least cost path analysis to generate transportation paths between buildings on campus on an individual basis for each user. These paths would avoid going through buildings and used a cost surface that encouraged students to walk on sidewalks and through squares. Figure 3-7 shows the derivative of the various steps in the raster least cost path analysis process.

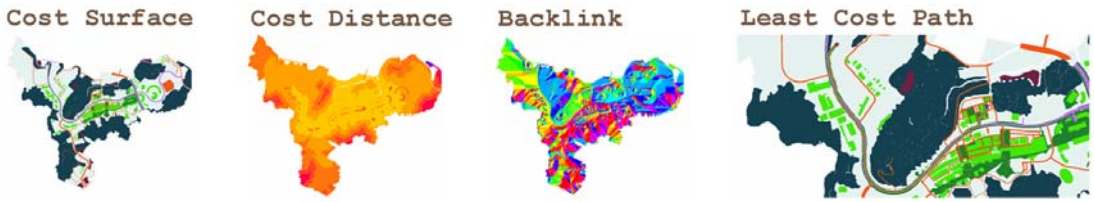


Figure 3-7: *Least cost path analysis across Kyushu University (Kojo, 2006)*

They then connected these paths to the individual building transportation networks at their ends, and did routing inside the buildings. They wrote custom scripts to do routing across different levels of the building. If the destination the user had selected from a drop-down list was on the second floor of a building, the script generated a route on the network of the first floor to the nearest set of stairs or elevator, and then ran a separate route analysis on the second floor from those stairs or elevator to the destination. Figure 3-8 shows the custom application developed to do the routing within a building, the two dimensional view of the networks, and the route generated by that application.

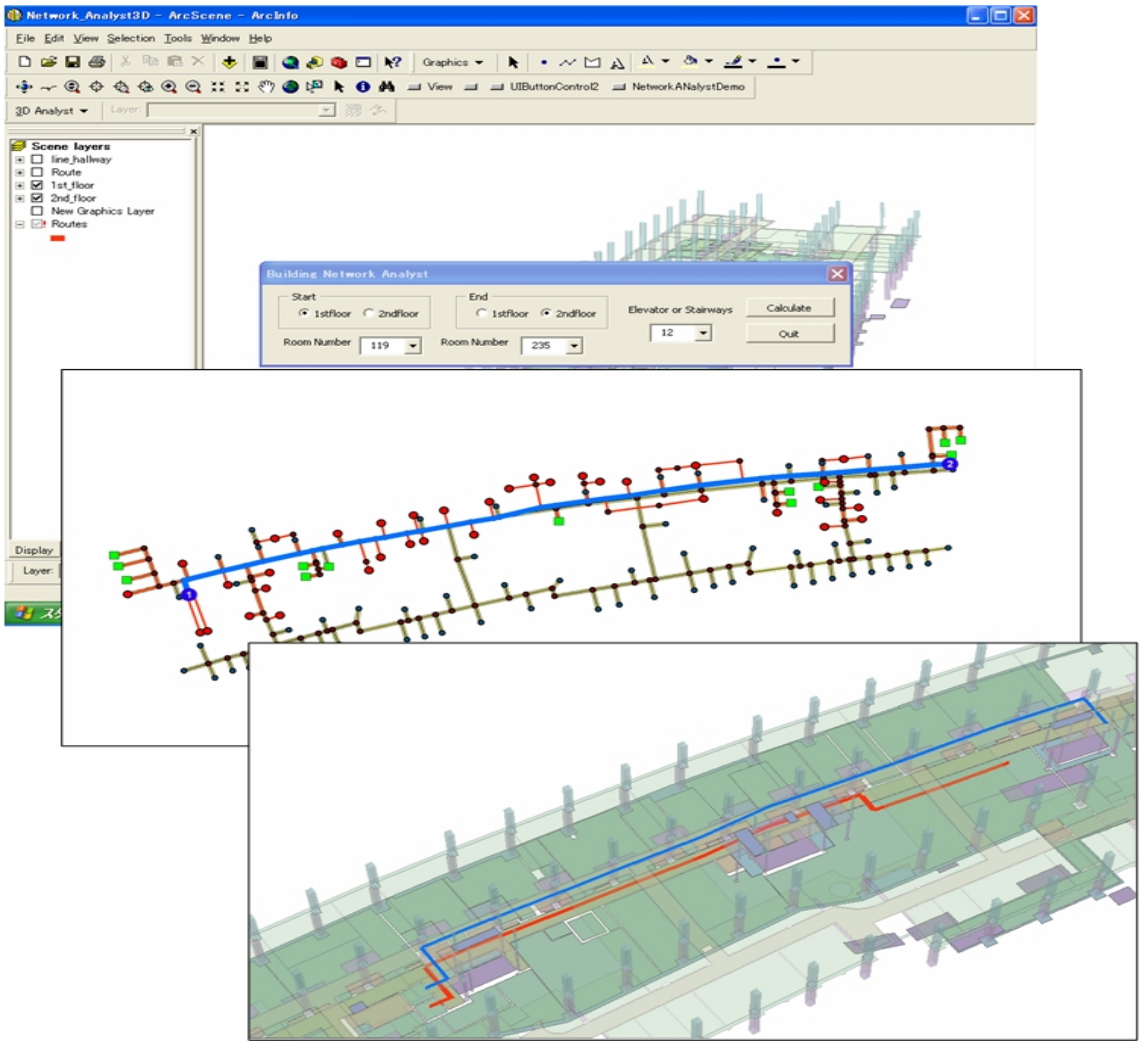


Figure 3-8: *Network way finding across Kyushu University (Kojo, 2006)*

3.6.2 Proposed Solutions

There are multiple ways of modeling way finding in a GIS. It was decided to explore three different methods of conducting way finding, and see how they could be applied to the interior spaces of buildings. The first is based on GIS's ability to model the connectivity between features, the second is using raster analysis to do path generation between two points, and the last is to utilize the customized transportation network tools available in GIS.

Topologic Way Finding

One of the major advantages of GIS is the ability to model the relationship between spatial features. This modeling is commonly called topology and can be persisted as part of the data or established on the fly as part of the software. Topology can model the relationships between spatial features such as adjacency, intersection, and containment. These

relationships may be harnessed to conduct way finding through a set of features based upon their locations and adjacency. Using the topologic relationship between features, each room or hallway can be attributed with the correct path to take in order to get to a specific room or hallway. Finding a path between the features is as simple as selecting the correct attribute from the feature and following it to the next feature in the chain. By jumping from feature to feature, pulling the same attribute, the user can trace a path to the destination.

In order to generate the attributes necessary for this type of way finding, a simple GIS model is necessary. This model consists of several steps that are completed at the time of the model's execution.

1. Start from the destination feature (e.g., MeetingRoom).
2. Add a new attribute to all the features that will hold the value indicating which direction to travel (e.g., Directions to MeetingRoom).

   The identifier for the destination feature should be defined at this point (e.g., Destination).
3. Select all features that are adjacent/accessible to the destination feature.
4. Deselect all features that have an identifier for the new attribute.
5. Set the new attribute to the remaining selected features.
6. Select an adjacent/accessible feature and repeat the process from step 3 until all features have definitions for the attribute created in step 2.

For example, in Figure 3-9, our destination feature that we begin with is called Room Feature A (Room FA). The model adds a new attribute to all features called Directions to FA. The identifier for the Directions to FA attribute is defined as Null for Room FA. Once this is complete, the iterative process begins by selecting all features adjacent/accessible to Room FA. In this case, there is one adjacent/accessible feature called Room Feature B (Room FB). The Directions to FA attribute associated with Room FB is then given the identifier FA. The process then repeats as all features adjacent/accessible to Room FB are selected. Those that already have an identifier in the Directions to FA attribute are deselected, and then all remaining features are given the identifier FB for the Directions to FA attribute. In this case, there are two adjacent features that have undefined identifiers for the Directions to FA attribute: Room Feature C (Room FC) and Room Feature D (Room FD). The Directions to FA attribute that is associated with all room features are now defined, and the model is finished. You can then trace the path back to FA by pulling the value on Directions to FA from each consecutive feature.

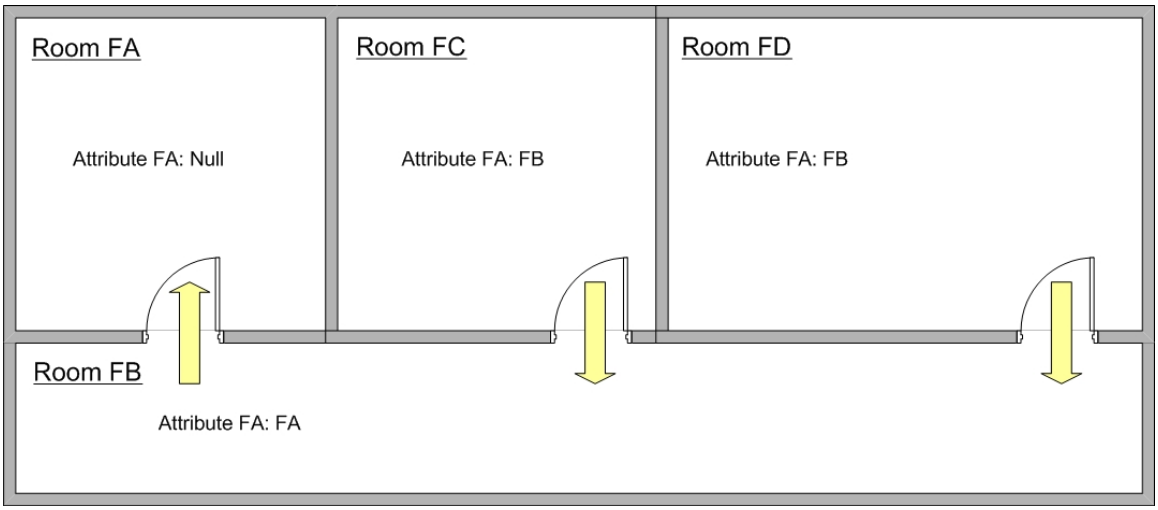


Figure 3-9

One of the advantages of this system is quickness. If this method is used to generate paths on the fly, it's based on simple feature relationship queries which the GIS can perform easily and quickly. If the chose is made to persist solution for each room as attributes, the resolution is even fast because the way finding becomes a simple collection of table queries. Also, since the model is based on a simple attribute system, it easily supports three-dimensional way finding. As long as the 3D features share a face, this proximity based solution will still generate valid results. The analysis required to populate these attributes is simple and easily understood.

The disadvantage of this model design is that the analysis is very sensitive to the quality of the underlying data. Rooms must only share adjacent edges at the doors where they are joined, and feature in 3D need space between them to separate floors. If these rules are not adhered to then you get false paths through the geometry. Furthermore, there is no way to calculate travel times using this way finding method, since it selects whole rooms instead of actually calculating a path through them. If the routes are persisted as attributes the model needs to be re-run in order to re-calculate these attributes each time the source data is change. Additionally in order to persisting each solutions requires one attribute per existing room feature which means that this recalculation can be time consuming at the time of update.

Raster Analysis

Raster analysis can be used to generate a path between two points. The first step in this process is developing a cost surface that represents the impedance or restriction of movement across the study area. Then cost distance analysis, from the starting point, is performed on this surface. This analysis calculates the travel time to every cell on the raster from that starting point, and the back link raster which shows the path back to the starting point. Then utilizing the cost distance and back link rasters a least-cost path analysis can generate paths to any location or set of locations within the study area. These paths are single cell paths that travel from the destination back to the source using the back link raster.

Raster analysis can be used for way finding in interior spaces of buildings. The walls become restricted areas so that any path calculated is forced to go around them. Doors are represented as a single line of cells having the impedance associated with the time it takes to open them. Each cell in a hallway or room has a value equal to the amount of time required to traverse the length of that cell. Then it is a simple matter of running the cost distance analysis and conducting the least cost path analysis to the destination. Figure 3-10 provides an example of what the resulting raster cost path might look like. The result is a path through rooms and hallways with the travel time to the destination. This analysis method provides other opportunities such as increasing the impedance of cells within narrow corridors demonstrating that they are harder to traverse, or buffering walls to increase the restricted area to force the walking path more towards the center of the hallway or room.

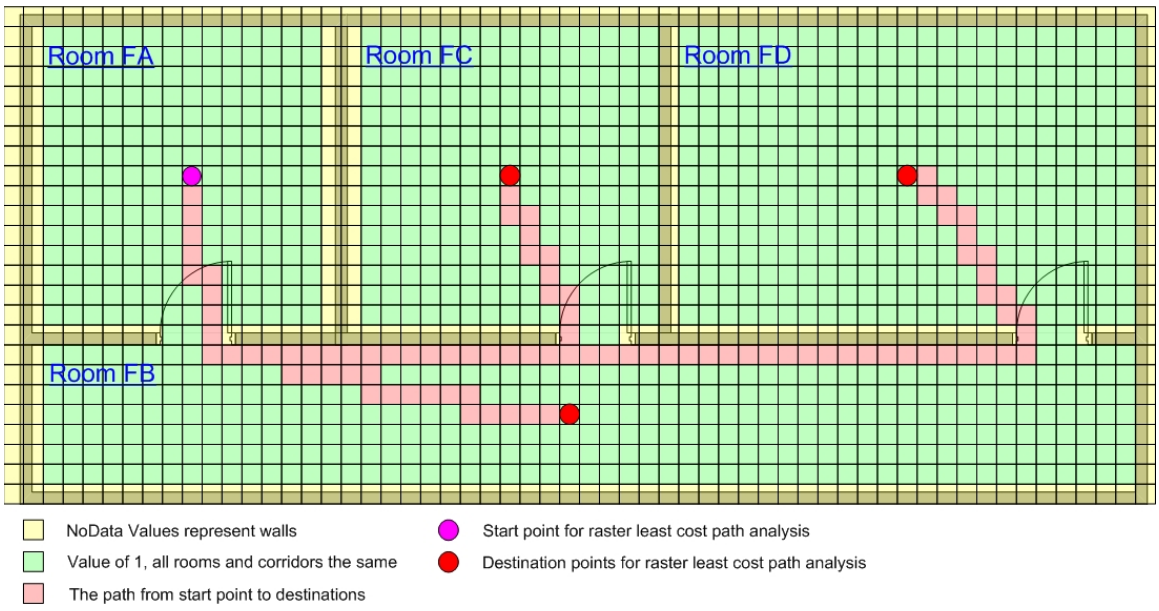


Figure 3-10

One of the advantages of this method is that it is least sensitive to data quality. Walls as single lines or polygons can be used, as long as they roughly define the boundaries of rooms for the raster analysis. The least cost path process can almost always produce a route.

The disadvantage of this model design is that this analysis process is computationally intensive, so this process is not well suited to on the fly generation of paths for way finding. Additionally this form of raster surface analysis is limited to two dimensions, meaning that it would require large amounts of custom scripting to support way finding in three dimensions.

Network Analysis

Network analysis uses a specialized data set, called a geometric network, and a set of analysis methods to determine connectivity and/or paths to different parts of the network. Geometric networks can model the infrastructure of various types of utilities such as electrical, telecom, water, waste water, storm water, and gas. It allows the user to trace connectivity through the network to model impacts of service interruptions on different parts of the network. It can also be used to model transportation networks, such as roads, buses, light rail, and air travel.

Using what is called a multimodal network, one can actually combine different types of transportation together into one comprehensive network. The advantage is that one can trace a path from point to point while utilizing different types of transportation. For example, one could trace an optimal multimodal path to a workplace: by car to a depot, by bus to a stop, on foot to the destination. More importantly multimodal networks allow you to trace the shortest path across the network based on impedance and delay built into the geometric features of the network. This allows you to use a network to identify the shortest path between two points, the amount of time it will take to traverse that path, and the directions along that path someone should take. Figure 3-11 shows a very simple transportation network for a small building, note the network is a total separate collection of geometries from the building. At present transportation routing technology does not support three-dimensional transportation networks, meaning that vertical lines cannot be represented within the network, and the route generation algorithms do not take into account the elevation of the start and destination points.

Modeling movement through a building is inherently a three-dimensional problem unless the building happens to have only one floor. You need a tool capable of tracing a path from the starting point, to a stairway or elevator where the path can jump to another floor and continue on to its destination. Additionally, not all points of vertical transition can access all floors, so you must have a way of limiting which floors a specific transition point can let you jump to. For example, express elevators do not provide connectivity to each floor. Since current network technology does not support this type of analysis, an alternative method must be used. One can conceptualize movement through a building as a multimodal network, with each floor as its own discrete network and each staircase and/or elevator as the points of transition between these networks.

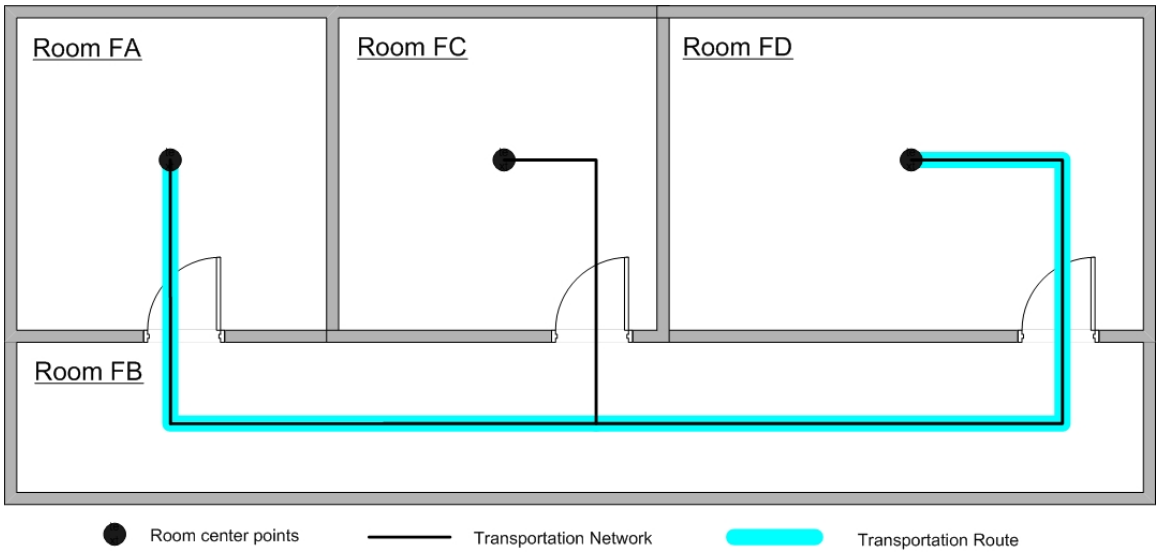


Figure 3-11

The primary advantage of this method is that it can support way finding across many floors efficiently. The path generation analysis methods are core to the software, and run very quickly. The multimodal transportation network is well documented in the software. The downside is that the generation of these complex geometric networks can be very time intensive. Past efforts at generating internal transportation networks for buildings have all been done manually, and at great expense in time and effort. The scope of the project did not lend itself to this exhaustive data creation.

3.6.3 Selected Solution

The ideal solution would be to harness the path generation abilities of least cost path analysis without losing the advantages that a multimodal transportation network gives in terms of speed and ability to represent movement across different floors of a building. The selected solution was to blend least coast path analysis and multimodal transportation networks together. Use raster based least cost path analysis to create a route between two rooms on the same floor of a building. Then conduct that analysis iteratively for every room on that floor. Then combine all those paths and convert them to a vector transportation network. This would be conducted for each floor of the building, resulting in one network per floor, which then could be connected through transportation nodes representing the stairs and elevators to form a multimodal transportation network. This method couples the ease of route generation from least-cost path analysis with the capabilities of a multimodal network. Automating this process would allow the building manager to re-run the network generation any time changes were made to the building's layout.

3.6.4 Network Generation Method

It was decided to implement this solution using Esri ArcCatalog platform using a custom developed Python script. Initial prototyping of the cost surface, source and destination points generation, cost distance analysis, cost path generation, and raster to vector conversion would

be conducted in ModelBuilder. Once a functioning model was produced, it was exported to Python, and modified to run cost-distance and cost-path portions iteratively on every room. For more information on this python script, view Appendix C.

Data Preparation

A Model Builder model was developed in Esri ArcCatalog to generate a cost surface for the floor of buildings. The model converted the floor of a building to a raster data set with a cell size of one tenth of a meter. On conversion each cell adopted the unique identifier of its source polygon as its raster value, however all values for the floor were reclassified to 1. The reason all parts of the floor were treated as the same was to represent that it was just as easy to walk on one side of the corridor as it was to walk on the other. For the purposes of this model it was assumed that walking through doorways, or down narrow hallways did not decrease travel time significantly. The walls for the selected floor were then buffered by their width, and converted to raster using the exact same spatial reference as the floor. This was done to prevent interpolation when the two rasters were combined. All cells representing walls were reclassified with a value of NODATA. NODATA was used for walls because the Esri Cost Path function treats NODATA areas as impassable. Map algebra was used to combine the two rasters, with the NODATA cells from the walls raster overwriting the cells on the floors raster.

The same ModelBuilder model converted all rooms for the floor from polygons to points using the Esri Feature to Point tool. This created a point for each polygon as close to the geometric center of the polygon as possible but still within the original geometry's extent. These points retain all the attributes associated with each room. All interior non-emergency stairways and elevators were converted from polygons to points using the Esri feature to point function. This model was then converted to a Python Script.

Path Generation

A second Model Builder model was developed to generate a path from one room to all other destinations on a floor. The model selects a point from the room points feature class and copies it into a new feature class. This point will be used as the start point for cost path generation. All the other room points are copied to a new feature class and combined with the staircase and elevator points to form the destination points feature class. Esri's Cost Distance GeoProcessing tool was used to generate cost distance and backlink rasters from the start point to all destination points. The Cost Path GeoProcessing tool was used to generate raster paths from the start point to all destination points.

This model was converted to a Python Script, and appended to the previous Python script created during data preparation. It was modified to loop through a list of buildings to generate raster cost paths between all room points for an entire floor of a building.

Network Creation

These cost path rasters were merged together into a single raster, using the Mosaic command in the Python script. For more information on the script, see Appendix C. The combined cost path raster was reclassified with anything having a value other than NODATA getting a value of 1. The raster was then converted to a vector feature class, with lines representing the paths between all rooms on the floor. This new line feature class represents the transportation network edges across the floor of a building between all rooms.

One transportation line feature class was generated for each floor of the building, and then added to the building transportation network manually. Non-emergency stairways and elevators were used to connect the transportation networks of each floor together in a multimodal network. These features were represented by points in a geodatabase, the performed the role of transition between the different modes, in this case floors, of transportation. An example of what the resultant transportation network looks like can been seen in Figure 3-12.

Note that this method has some advantages and disadvantages. For instance the shortest path between two rooms as represented by this model place the path of the network directly adjacent to a wall. In a room with only a single exit, there are potentially three exit paths as the model tries to cut the corner to the left and right of the door, and potentially go straight across the hallway to the opposite room. These problems and limitations will be discussed later on in the “Faults and Flaws” portion of the Conclusion.

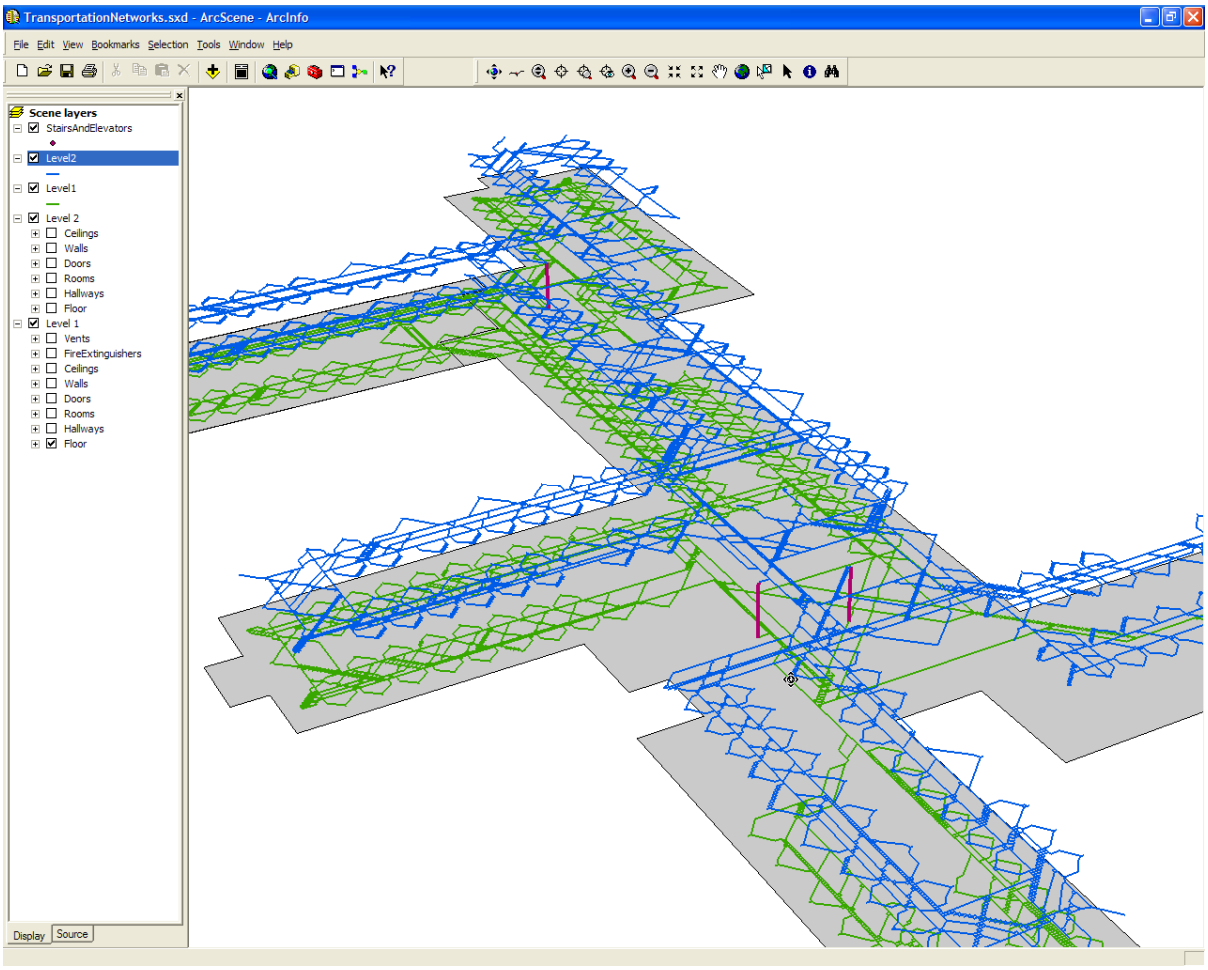


Figure 3-12: *Multimodal transportation network for buildings M and N*

3.7 Summary

The result of the data modeling effort was a geodatabase that contains the basic features of a building. The feature classes within the geodatabase are organized into three feature datasets. One feature dataset contains all the source data model information necessary to represent the basic components of a building. The second feature dataset contains the derivative analysis and visualization data component. The third contains the feature classes necessary to model a transportation network across the building. Once a working data model had been developed, the next step was to try and conduct analysis on the building features to see if it meets the need and requirements of the project's clients.

4.0 Editing and Analyzing Building Data

The next step in the major individual project was to test the data model that had been designed and developed to insure that it met the needs of the clients. Items tested included:

(1) the ease of editing and creating building components in the data model,
(2) the ability to link to external tabular data, and query through to that information from GIS applications,
(3) various types of spatial analysis on the data, displaying the results in thematic two and three dimensional maps, and
(4) the ability of the multimodal transportation network to map routes between rooms within the building.

These demonstrations were conducted using Esri's ArcGIS suite of products, including ArcMap, ArcCatalog, and ArcScene.

4.1 Editing the Data Model

The simple nature of the source data model allows the user to easily edit a building's rooms or walls. As the building's configuration changes, derivative data used in analysis and visualization is updated through the use of geoprocessing scripts. Each time the user finishes editing the building, all the user needs to do is rerun the script and the analysis, visualization, and network data models are built automatically. Additionally, by separating out *z*-values in a separate table and relating them to simple features in the source data model using elevation identifiers, it is easy to rapidly change the base elevation and/or extrusion height of features. Figure 4-1 demonstrates adding a new wall to subdivide a room in Esri's building M.

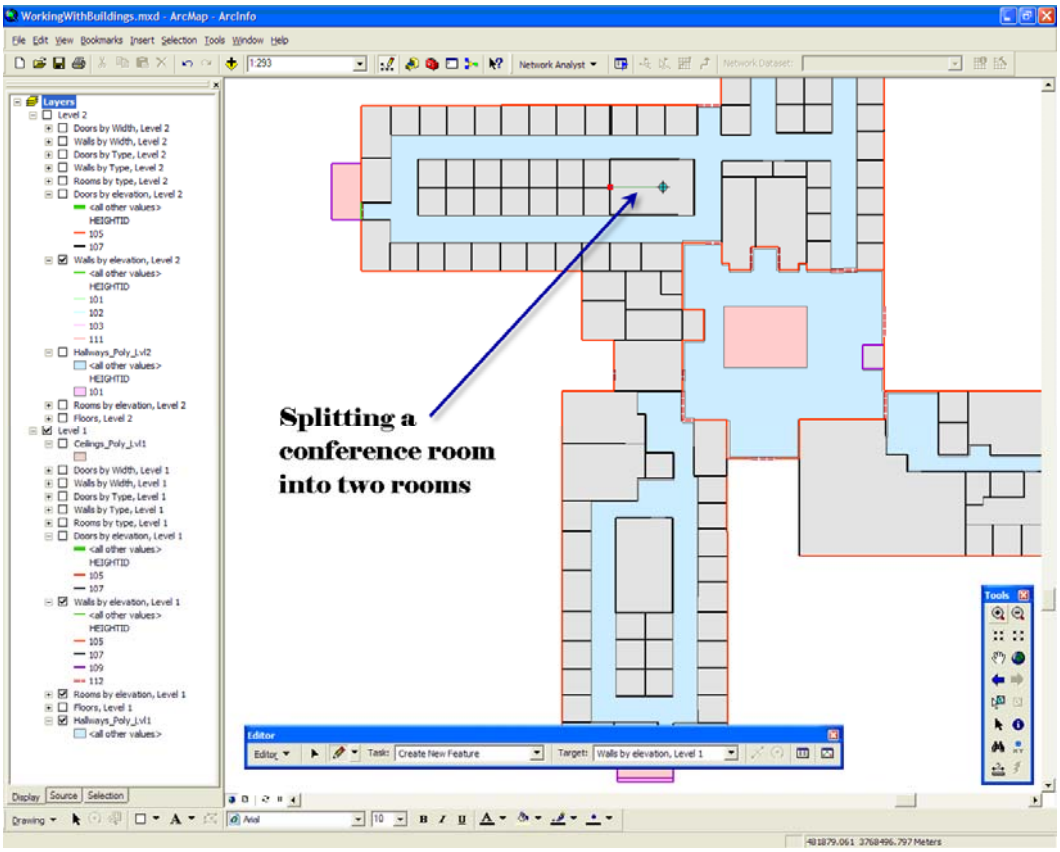


Figure 4-1: *Editing of rooms in ArcMap*

Additionally, one of the main advantages of GIS is the ability to select sets of features by their spatial relationships to other features in the database. For example, with the simple design of this database it is easy to conduct a spatial query to select all the rooms within a building, and then based upon a shared base elevation select all rooms, walls, doors, and windows for a specific floor. This ability to query spatial relationships in real time, without having to model those relationships in the data itself, is one of the primary advantages of storing building information in a GIS environment.

4.2 Querying Non-GIS Data Source

All features within the source data model have unique identifiers associated with them. This allows the user to relate data from existing tables or from external databases to features within the building. This means the GIS data can be light-weight and simple to manage. More complex databases e.g. for managing personnel, equipment inventories, or maintenance schedules can be maintained in third-party applications. Figure 4-2 shows the results of an identification of a room in building M on Esri's campus. The attributes displayed are pulled from an external table containing personnel information about every Esri employee.

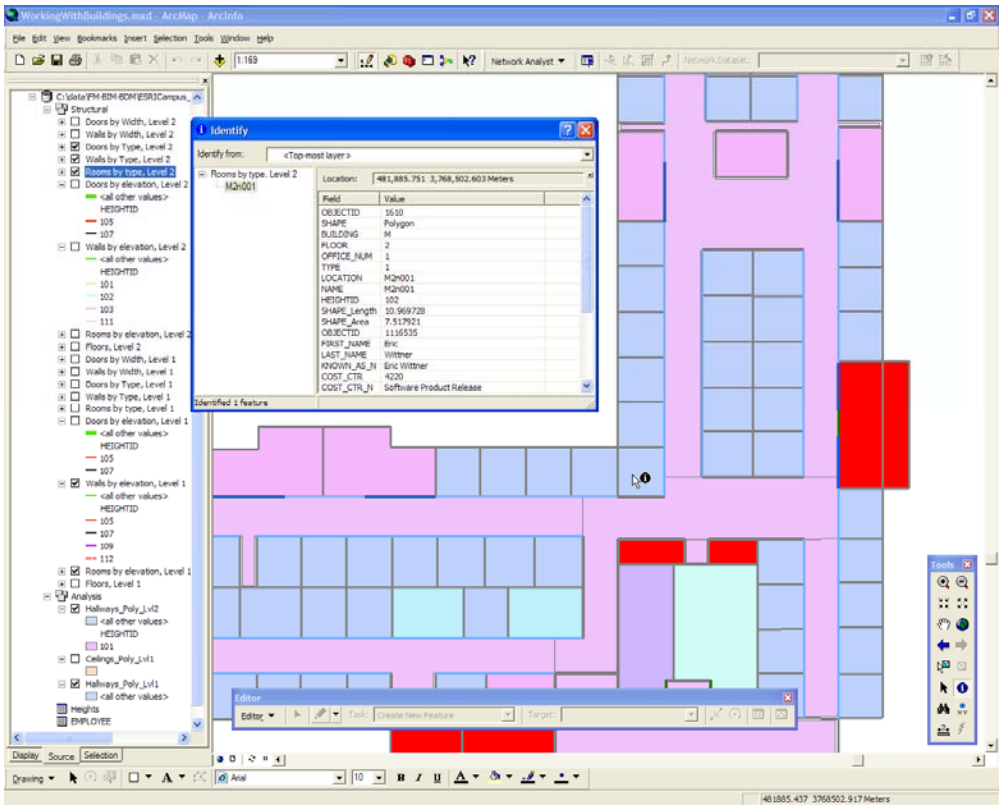


Figure 4-2: *Querying room attributes in ArcMap*

4.3 Support for analysis

Based on the attributes stored on each feature in the source data model, derivative data models can be constructed on the fly to support analysis needs. For example, a geo-processing model could be developed to support the construction of building features suitable for use in FICM or BOMA area calculations. More commonly the attributes are used to convert all features in the source data model to polygons, or to generate new data layers from existing ones by sub-setting, subtracting, or combining existing features. Once the necessary analysis components are generated from the source data model, a variety of analysis processes, both vector and raster, can be conducted.

Three analysis examples were conducted to test the data model and see if it met the client's needs and requirements. The first leveraged the ability of GIS to calculate proximity and adjacency of spatial features in order to assess how vulnerable rooms in a building are to penetration from an external threat. The second tested the ability of GIS to convert vector building components into a raster data set, and conduct cell based analysis and generalization on the structure to assess the hazard an explosive component stored in a building represented to all the rooms in the building. The third process tested the ability of the model to support

analysis not only within a building, but across a campus between buildings while integrating GIS data representing features external to the building.

Assessment of Vulnerability through Topology

For the vector based proximity analysis it was decided to use GIS to select all rooms that had an exterior wall, window, or door and give them a vulnerability value based on how easy it was to enter the rooms through their weakest external feature. Obviously doors were considered easier to breach than windows, and windows were considered easier to breach than walls. Then all the remaining rooms in the building were assessed and given vulnerability values based on their proximity to the nearest room with an external wall, window or door. Model Builder was utilized to automate the analysis process, as shown in Figure 4-3.

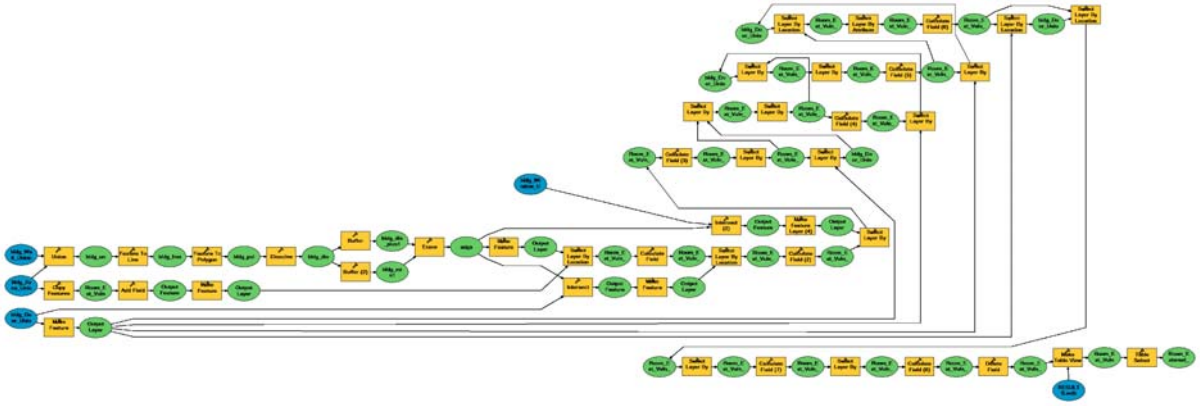


Figure 4-3: *Vulnerability model*
See Appendix D for a larger version

The resultant attribute value for each room showed its vulnerability to an external threat, could easily be visualized in GIS, re-associated with source data through unique room identifiers, and could easily be reproduced or modified using the existing Esri geoprocessing framework. Figure 4-4 shows the result of the GIS analysis, with red representing the most vulnerable rooms, and the cooler colors representing rooms that had lesser vulnerability. Note that the model exposed some flaws in the input data. White rooms had no door connecting them to adjacent rooms, indicating a problem with the source data.

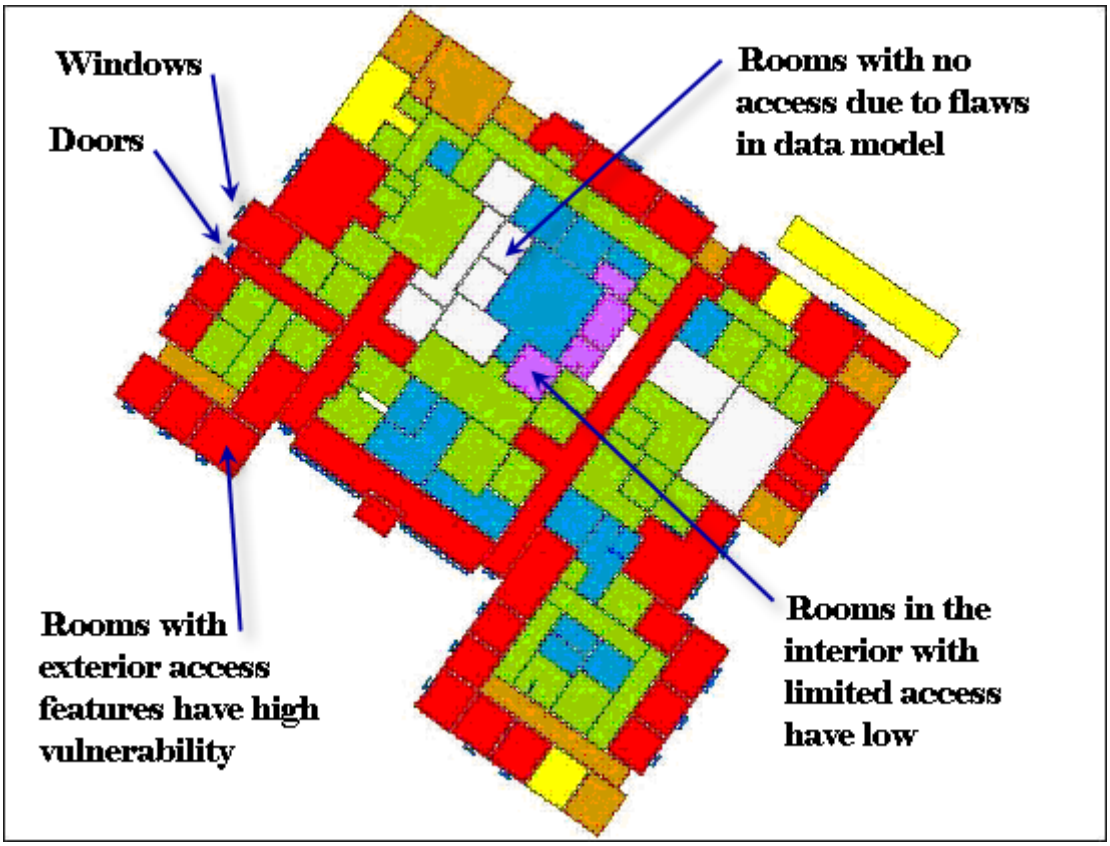


Figure 4-4: *Vulnerability assessment of building 1*

Hazard Assessment Using Raster Cost Distance Analysis

For the raster analysis demonstration, it was decided to attempt the use of cost distance analysis to assess how much an explosive device in one room threatened all the surrounding rooms. The rooms were converted to a raster, with a cell value equal to their cell size representing that the explosion only had to contend with distance to diminish its force. Walls were given varying raster values depending on their composition and size, to represent how different types and thickness of walls would dampen the effect of an explosion. These two rasters were combined to create a cost surface raster on which to conduct cost distance analysis. This process was automated by implementing the model in Model Builder, as shown in Figure 4-5, so that it could be repeated on any floor.

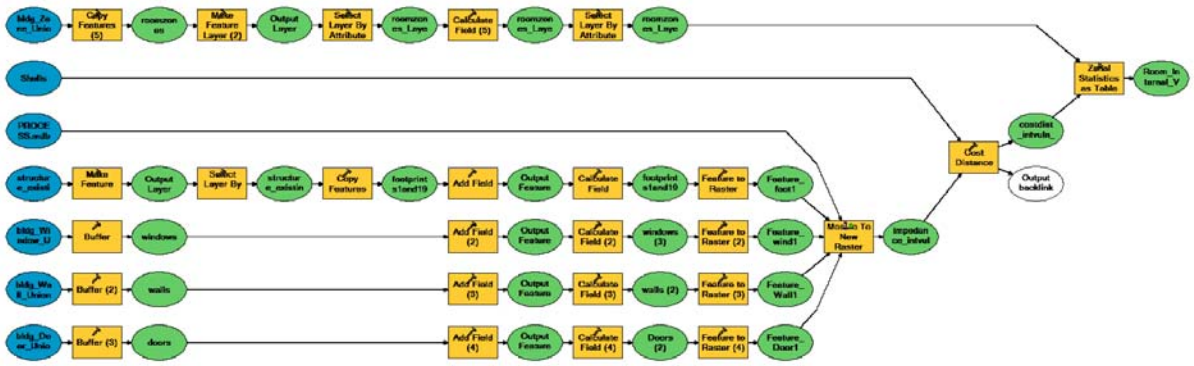


Figure 4-5: *Hazard model*
See Appendix E for a larger version of this Figure

The result is a cost distance raster that provides a rough estimate of how badly each cell within the raster would be affected by the blast. An example of this raster is shown below in Figure 4-6, with areas in great risk in shown in red.

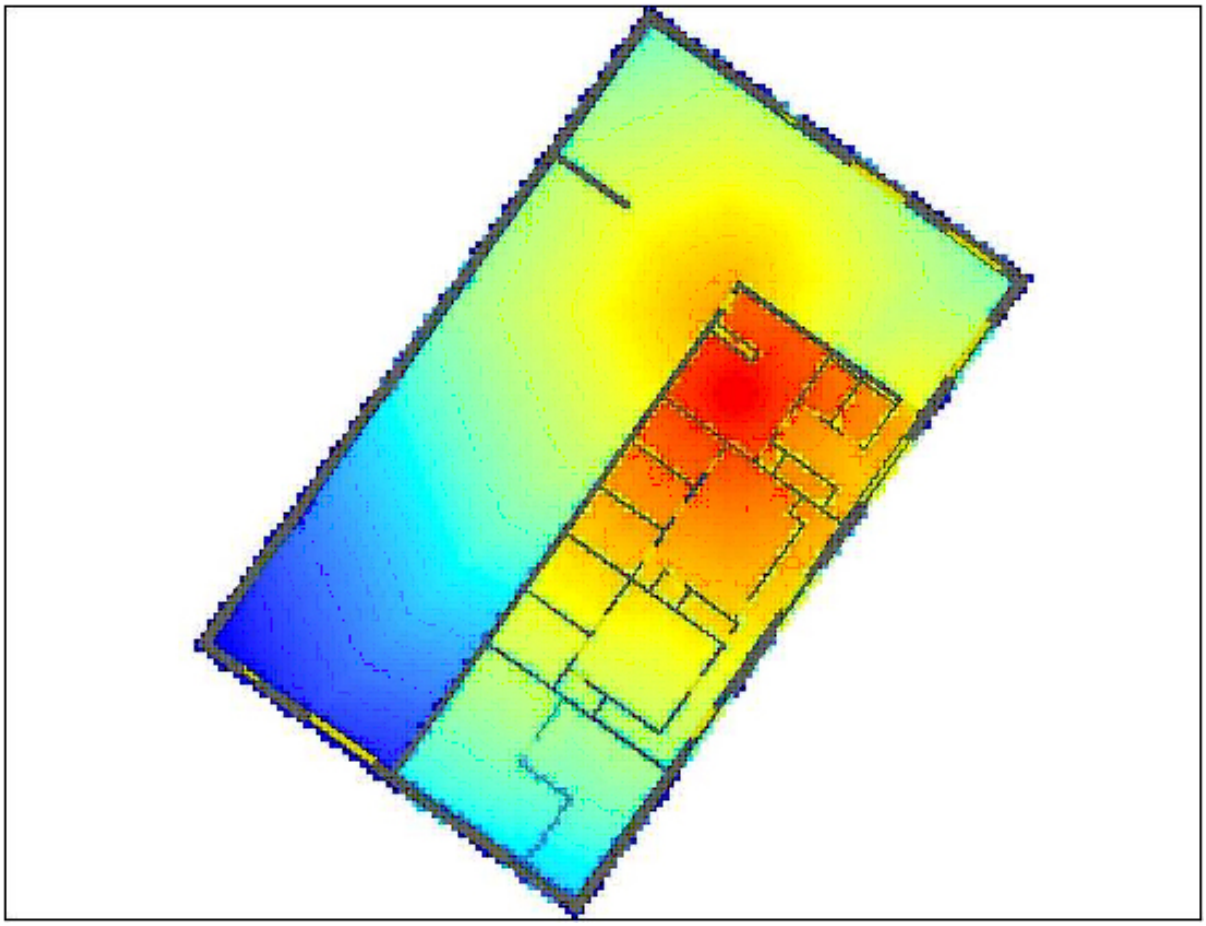


Figure 4-6: *Raster hazard assessment*

The resulting raster is then generalized for each room, using zonal statistics, to produce a value for each room indicating in how much danger it is from a potential blast, based on the highest value cell within each room.

Proximity Analysis Across the Interior and Exterior of a Building

The final analysis test was to conduct analysis across a campus from the interior of one building to the interior of another building. It was decided to assess how accessible each room in two buildings on different sides of Coast Guard island were, to a room in one of the buildings, based on the travel distance between them. A cost surface was generated using Model Builder that included external features on the campus such as walkways, roadways, parking lots, athletic fields and courts, vegetation, and internal building features such as walls and doors. The data was automated using Model Builder, as shown in Figure 4-7, so that it could be conducted again as more buildings were added to the data model.

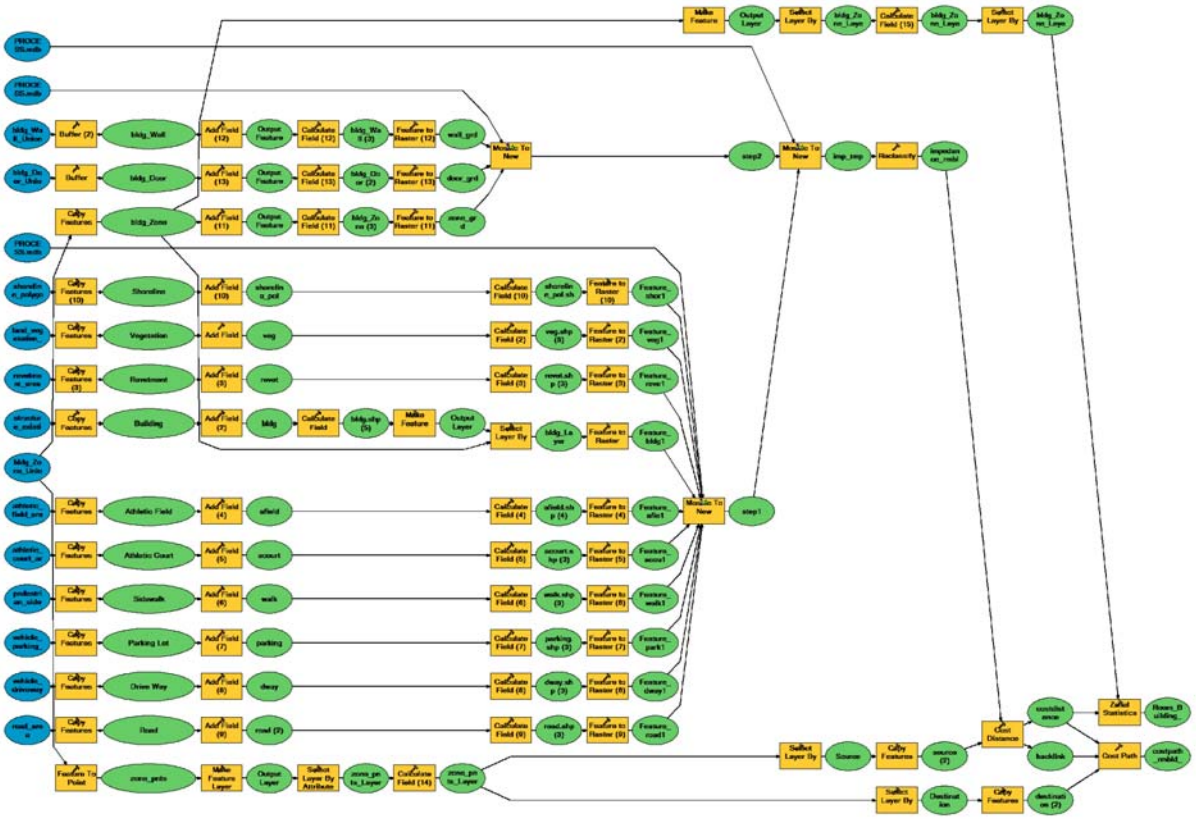


Figure 4-7: *Accessibility model*
See Appendix F for a larger version

Cost distance analysis was conducted on this surface, then the cost surface was generalized based upon the rooms in each building using zonal statistics. The result was an attribute attached to each room in both buildings that showed how accessible each room was from one

location. Areas in the darker grey were highly accessible, and lighter areas were farther away and hence less accessible.

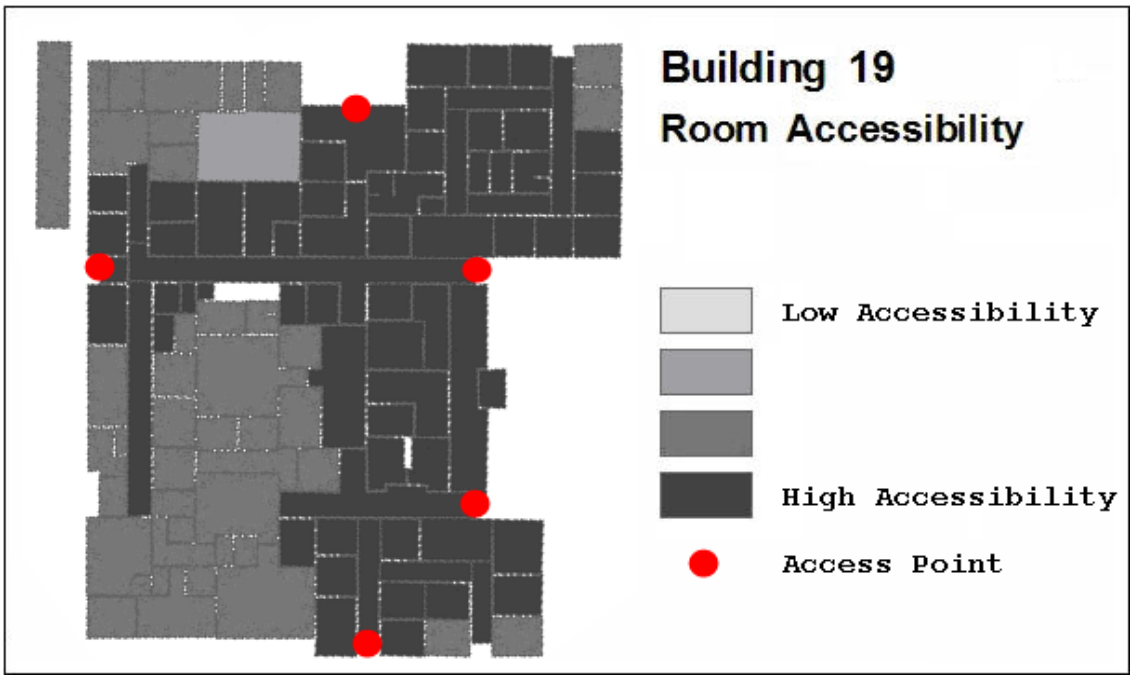


Figure 4-8: *Accessibility of rooms in building 1 and 19*

These three analysis examples demonstrated that the data model was robust enough to support a wide variety of analysis methods, not only in the interior of buildings, but across a campus.

4.3.1 Visualizing Building Information in three dimensions

The data model also proved capable of representing building information, and thematic data in three dimensions. Figure 4-9 shows building M and N in three dimensions, with each room colored according to its type. The purple rooms are offices, the green rooms are balconies, the light blue rooms are conference rooms, etc. Any attribute can be displayed thematically in three dimensions.

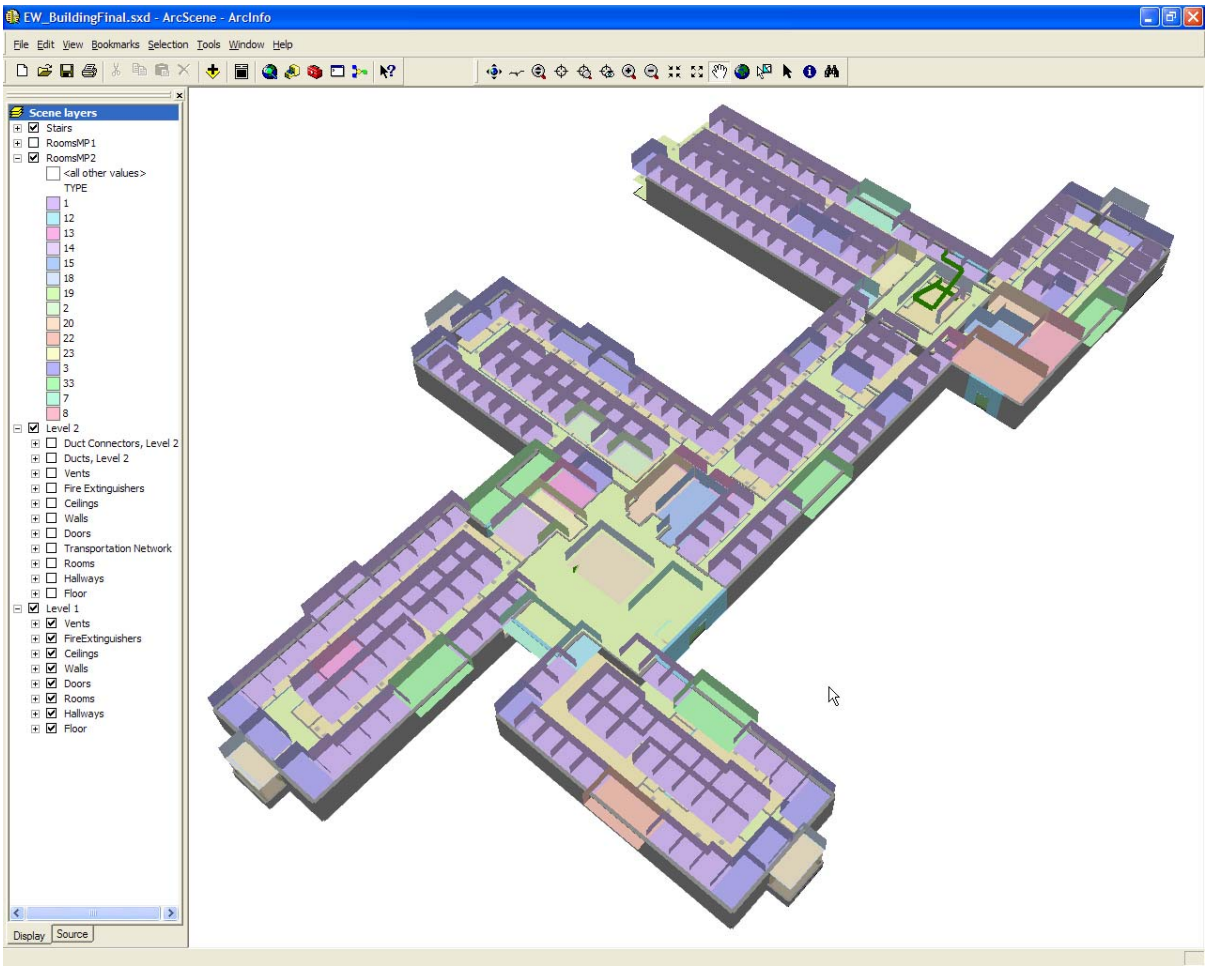


Figure 4-9: *Thematic map of room types in Building M and N*

As well as supporting the query of attributes from certain types of derivative data in three dimensions. Figure 4-10 shows information about the resident of an office queried in three dimensions.

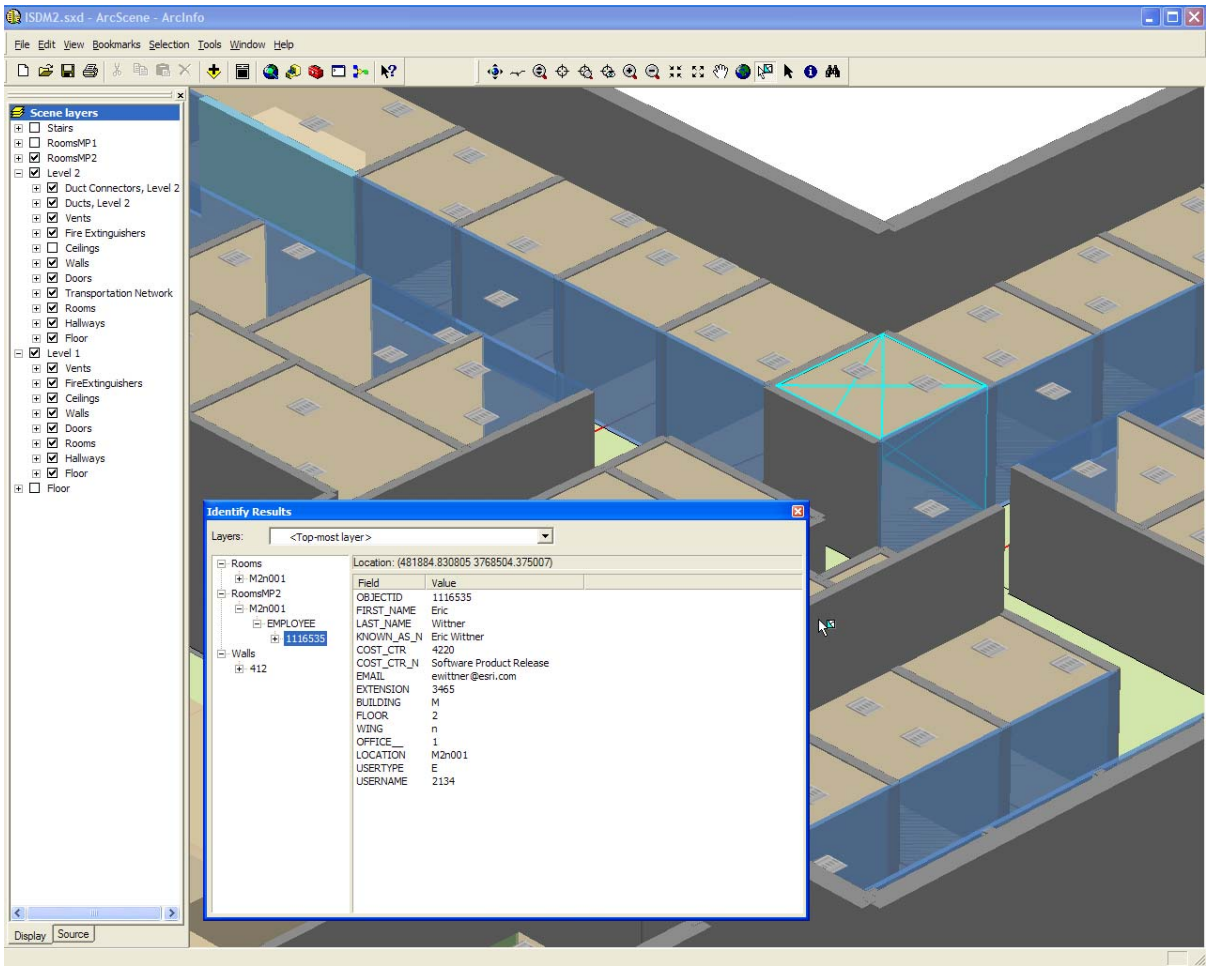


Figure 4-10: *Querying feature attributes in ArcScene*

4.4 Route Generation on the Network Data Model

The network data model proved capable of representing movement across multiple floors of a building. However route generation is limited to two dimensions in GIS. Figure 4-11 shows a route generated from a room on level 1 to a room on level 2, through use of one of the elevators. The result was a route that could be used to guide a person to a specific office or conference room in a building.

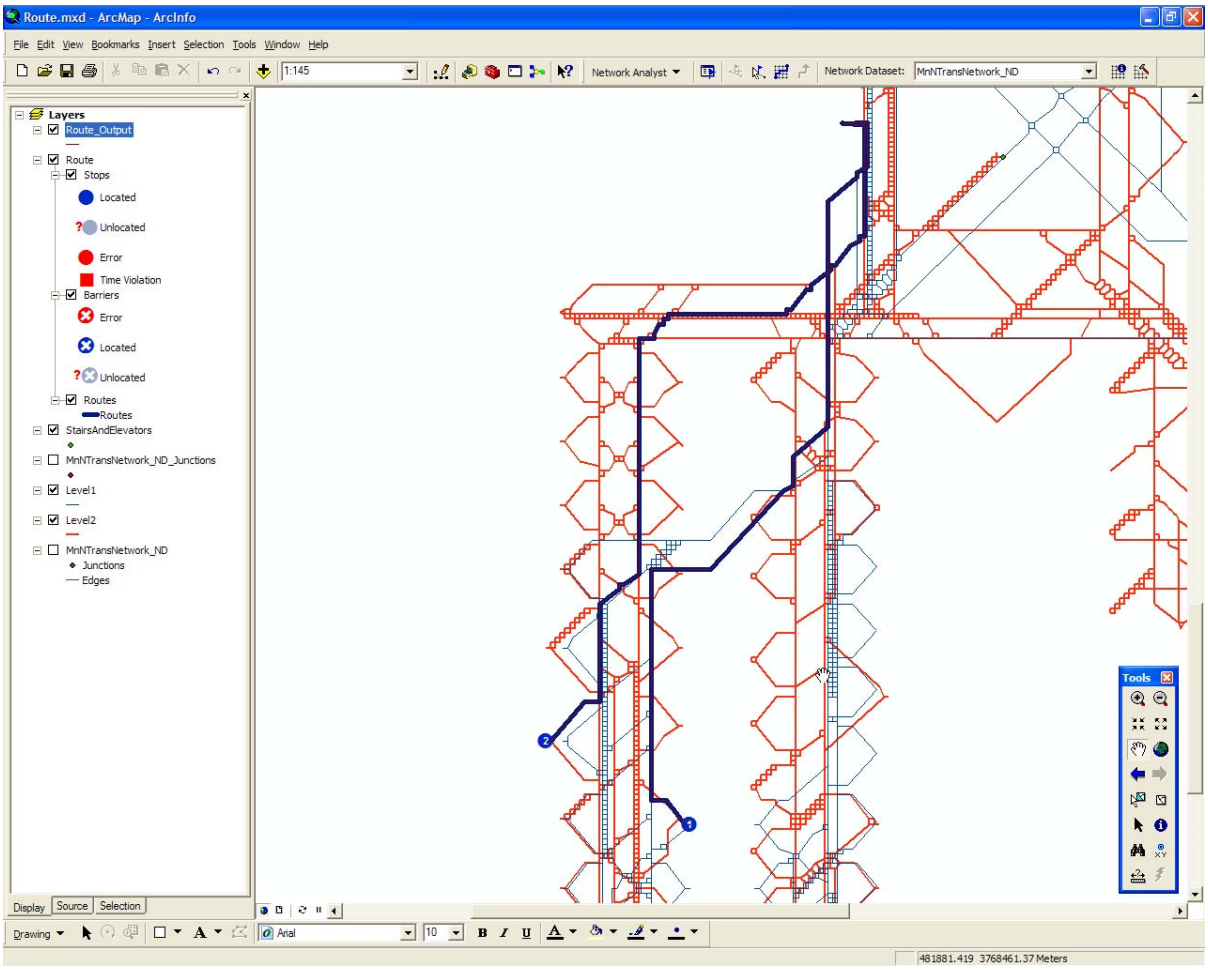


Figure 4-11: *Route from floor 1 to floor 2*

4.5 Summary

The data model proved that with its existing features and relationships it could support editing, attribute querying, and a variety of spatial analysis methods. Although the types of analysis conducted here are by no means exhaustive, they did prove that the model supports the most basic types of analysis commonly done in GIS, and those most likely to be used by building and facilities managers.

5.0 Conclusion

This project focused on developing a data model for a building. The concept began with researching the needs of clients who are working with building information, then synthesizing those needs into a list of requirements. Existing data models and standards were assessed to guide the development of a data model that supported the clients' needs. A series of strategies were developed to populate the data model with the clients' data. Then analysis was done to test the effectiveness of the data model in meeting the clients' needs. This testing revealed some faults and flaws in the data model, and the methods used to populate it.

5.1 Faults and Flaws

Querying on features in three dimensions proved unreliable. The identify and selection tools in ArcScene use ray tracing to detect which features the user has clicked on. These ray tracing routines fail to intersect extruded features, such as polygons and lines. [1]

The primary flaw in the results of the transportation network is the spider-web effect of combining multiple transportation paths generated from least-cost path analysis. Each path mapped the shortest route between two rooms, following as closely to the walls as possible in order to avoid added and unnecessary distance. However, this result does not accurately represent how people move down a corridor. When people walk, their preference is to use the middle of the hallway rather than hug the walls. Additionally, when combining these paths, every corridor in the building has multiple paths or lines running down it, and the paths between rooms that are close together form a spider-web of cost paths lines. The effect can be seen in Figure 5-1. It clearly shows how confusing the transportation network generated through this method can be. Although this method proves effective for generating paths between rooms in a building, it is not visually intuitive to the user and not suitable for a way finding application.

[1] The 3D selection issue has been resolved in ArcGIS 10.

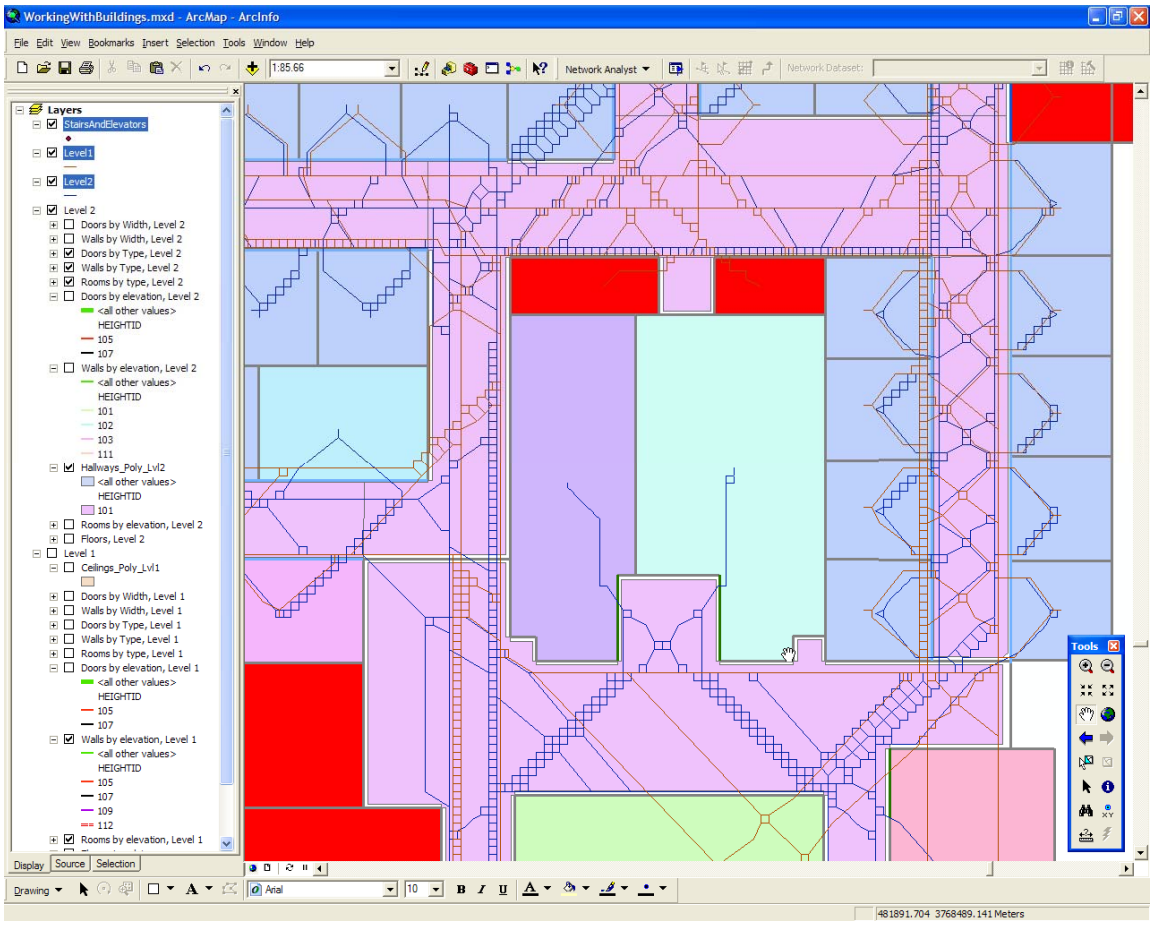


Figure 5-1: *Spider-web effect on least cost path generated transportation network*

The consumers of a building transportation network expect it to be a regular network of straight lines with lines up the center of corridors and single-connecting lines from the corridors to the center of each room (see Figure 3-11).

Another issue is that the route generation routines in Esri's Network Analyst were designed for use in two dimensions. This limitation is two-fold: (1) there is no way to easily place the start and end points for the network generation routine on different levels of the building based upon elevation values; and (2) the resulting route generated by the routine is inherently two-dimensional, dropping all *z*-values from the features it was generated from. In order to effectively display the route in three-dimensions, the user must manually attach elevation values to the route. [2]

[2] ArcGIS 10 now supports 3D network routing, with all results produced Z enabled if the input features were Z enabled.

5.2 Next Steps

Although this data model succeeds in representing the basic components of a building, there are a number of enhancements that could improve it. Some of these would improve visualization, others provide new methods of accessing the information, and the most important ones address the limitations in the existing data model or methods.

5.2.1 Multipatch Building Exterior

One enhancement that would assist in the visualization of the building is an exterior multipatch shell for the building. A multipatch shell could have geo-typical images, or actual building facades mapped to it in detail. This would provide a more realistic external view of the building than is currently available.

5.2.2 Annotation and Labeling

The current data model does not provide any foundation for adding annotation and labeling. Clients will want to be able to display the attributes of features in three dimensions as annotations and labels. Some basic annotation layers should be added to the data model to demonstrate to the user how to do annotation and labeling in three dimensions.

5.2.3 Serving Building Data as a Globe Service

The current data model has been consumed in two and three dimensions using exclusively desktop GIS software. One of the next steps is to serve the data over the internet as a globe service for consumption in a light-weight data viewing client like Arc Explorer. This would test the ability of the dada model to support data delivery over the net. It would also demonstrate that GIS is capable of serving multiple buildings over the web, while supporting much of the functionality on the desktop to multiple users at the same time.

5.2.4 Exterior Landscape Features

The addition of landscape features in the data model such as street furniture, terrain models, and imagery would expand the data model from a building-specific data model to a facility-wide data model. This would require assessing what the kind of features landscape architects and city planners need to use in their planning processes, as well as integrating existing data models for utilities, hydrology, and road transportation. New models and methods would have to be developed to integrate the building and landscape information together into a seamless data model.

5.2.5 New Derivative Analysis Models

Two new derivative data models should be developed in order to test the data model and to see how well it can support data standards. A data model that supported space calculation using the FICM definition of the interior of buildings, and another for the BOMA definition

would demonstrate the ability of the data model to support these standards. It also might reveal gaps in the basic geometries used to represent features in the data model. At the bare minimum it would drive the development of a new process to split derivative features into the component parts necessary to assemble into different types of data models, and in support of different analysis methods.

5.2.6 Revised Network Generation

The spider-web network effect can be addressed by simply changing the cost surface on which the least-cost path analysis is conducted to generate the transportation network. By reducing the cost of moving down the center of hallways significantly, we can force the least-cost path analysis to use a common set of paths to get from room to room. By making it more expensive to move along a corridor or room close to a wall, we can force paths to trace a route to the center of rooms and to the center line of corridors using the shortest path possible.

This cost surface can be generated by conducting a Euclidian distance analysis from all the walls of the corridors. This produces a raster with high distance values at the center of each hallway. We can treat this raster like a digital elevation model, and conduct hydrologic analysis to generate a flow accumulation raster. The cells in the flow accumulation raster with a value of zero are pulled out and these ridges represent the center lines of the corridors. The distance raster for the corridors is inverted and then multiplied by 100. The center lines of the corridors are given a value of 1. This means that any cost path analysis will run along the center line of the corridor till it has to "climb out" of the pit on the surface to reach a room. The resulting raster provides a much better representation of a transportation network inside of a building, as shown below in Figure 5-2.

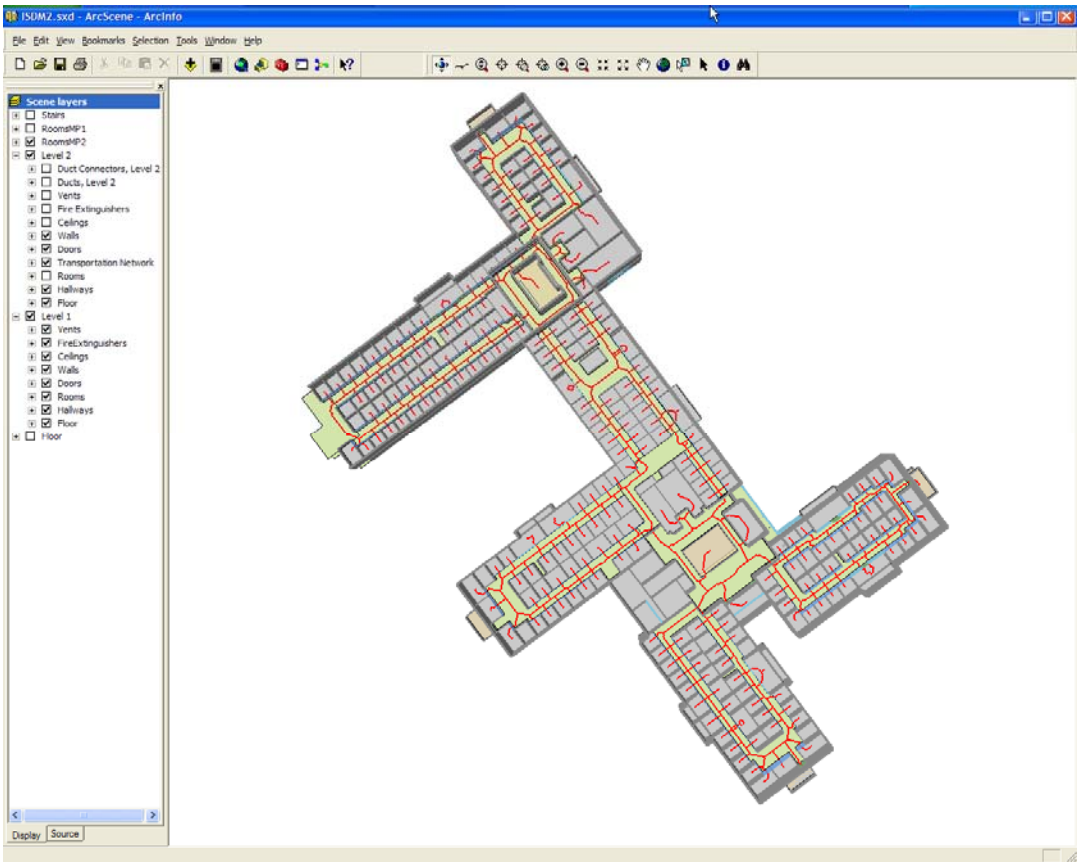


Figure 5-2: *Revised transportation network*

However, this analysis is raster based, meaning it can still produce a zigzagged effect. "Conventional cost-simulation algorithms consider 4 or 8 adjacent cells as the only possible direction in which a phenomenon can move. The constraint results in path traces that may be artificially zigzagged" (Xu J. & Lathrop, Jr R.G., 1994) There are methods available for conducting the analysis based on a wider selection of cells, to expand the number of potential directions the path can chose from. For example the analysis could be expanded to consider 16 cells, both those adjacent and those nearly adjacent, show in the central graphic in Figure 5-3. Another option is a conditional system, where the most likely path of direction is selected using 8 surrounding cells, and then the search is expanded in that direction to include more cells as shown in the right most graphic in Figure 5-3.

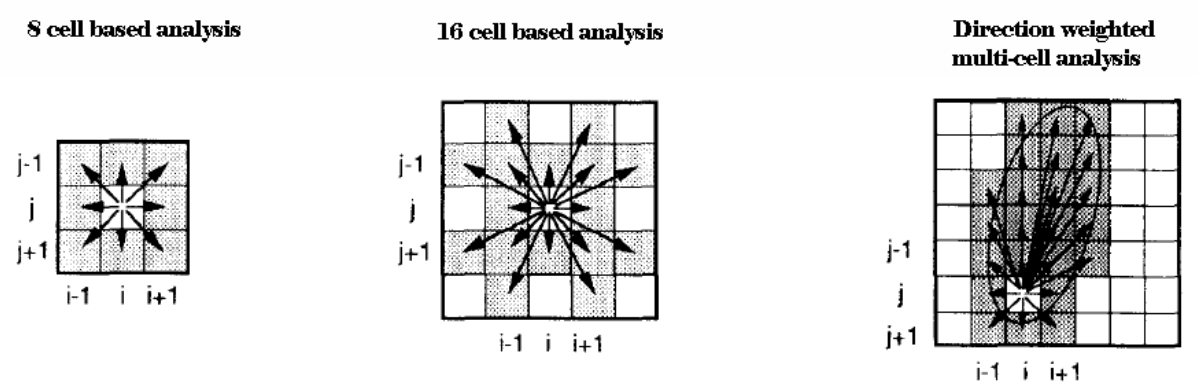


Figure 5-3: *Alternate cost path analysis cell groupings (Xu, & Lathrop, 1994)*

There methods require the generation and synthesis of multiple back link rasters in order to determine which cell the path should jump to next, since it can now move to none adjacent cells. Then a custom path allocation algorithm needs to be constructed to read multiple back link raster. The result is a much smoother line than normal least-cost path analysis would generate.

5.3 Summary

This data model provides the USCG and Esri with a starting point on which to develop data models specific to their needs. It succeeds in meeting the minimum requirements of the clients, but is still only a working prototype. Tools need to be developed to support population and visualization of the data in the model before users without a very strong technical background could work with the data model. GIS is just starting to be applied to this domain, and this data model provides the foundation to explore the capabilities of GIS in the building and facilities management world. As the ability of GIS to represent and analyze three dimensional features expands, so will the ability of this data model to represent and analyze the built environment.

References

American National Standards Institute, BOMA (1996) *Standard Method for Measuring Floor Area in Office Buildings.* Washington, D.C.: Building Owners and Manager Association International.

Arctur, D. & Zeiler, M. (2004). *Designing Geodatabases.* Redlands, CA: Esri Press.

Building Owners and Managers Association. (1997). Answers to 26 key questions about the ANSI/BOMA standard method of measuring floor area in office buildings. Retrieved November 15th, 2007, from http://www.boma.ca/FloorMeasurement/ANSWERS%20TO%2026%20KEY%20QUESTIONS%20ABOUT%20THE%20ANSI.pdf.

Dempsey, J. J. & Voellar, J. (2003). Internal Document, U.S. Coast Guard Shore Facility Capital Asset Management, Concepts Strategies and Metrics.

Dempsey, J.J. (2003). Internal Document, U.S. Coast Guard Shore Facility Capital Asset Management, SFCAM Metrics - Executive Overview.

Dempsey, J.J. & Hammond, D. (2002). Internal Document, U.S. Coast Guard Regional Strategic Planning, Integrated Decision Making Using Objected-Oriented Information Technology.

Department of Education, Working Group on Postsecondary Physical Facilities. (1992). Postsecondary Education Facilties Inventory and Classification Manual. Retrieved, May 2nd, 2010, from http://nces.ed.gov/pubs92/92165.pdf.

Esri. (2007) Data Model Introduction. Retrieved November 1st, 2007, http://support.Esri.com/index.cfm?fa=downloads.dataModels.intro.

International Alliance of Interoperability. (2007). Industry Foundation Classes FAQ. Retrieved May 5th, 2007, from http://www.iai-na.org/technical/faqs.php.

International Alliance of Interoperability. (2007). IFC2x Edition 3. Retrieved August 1st, 2007, from http://www.iai-international.org/Model/R2x3_final/index.htm.

Johansson M., & Roupe M. (2010). How can GIS and BIM be Integrated?, Poster, CADRIA Conference, 2010, retrieved May 2nd, 2010.

Karimi, H.A. & Akinci B. (2009). *CAD and GIS Integration*. Florence, KY: CRC Press.

Kojo, K. (2006). PowerPoint Slideshow, Routing solutions for Kyushu University, Internship Final Report at ESRI 2006.

Kolbe, T. & Bacjarach S. (2006) CityGML: An Open Standard for 3D City Models. *Directions Magazine*, November 15th 2010, http://www.directionsmag.com/article.php?article_id=2209&trv=1.

Lutz, M. & Ascher D. (2004) *Learning Python.* Sebastopol, CA: O'Reilly Media Inc.

Lynch, A. M. (2010) PenBay Solutions Announces New In-Vision OnePass Data Collection Capabilities for In-Building Geographic Information Systems. Directions Magazine, Glencoe, IL, United States: Directions Media.

Maidment, D. & Djokic D. (2000) *Hydrologic and Hydraulic Modeling Support with Geographic Information Systems.* Redlands, CA: Esri Press.

Mitchel, A. (1999) *The Esri Guide to GIS Analysis, Volume I: Geographic Patterns and Relationships.* Redlands, CA: Esri Press.

Open GIS Consortium. (2007). CityGML, Exchange and Storage of Virtual 3D City Models. Retrieved May 2nd, 2010, from http://www.citygml.org/.

Penobscot Bay Media (2007). Penobscot Bay Media, GIS Data Model for Interior Spaces. Retrieved February 21st, 2007, from http://www.penbaymedia.com/solutions/gis_data_model_for_interior_spaces.htm..

Weigel, D. & Cao, B. (1999). Applying GIS and OR Techniques to Solve Sears Technician-Dispatching and Home-Delivery Problems. White Paper. Redlands, CA: Institute for Operations Research and the Management Sciences.

Xu J., & Lathrop, Jr R.G. (1994) Improving cost-path tracing in a raster data format. *Computers and Geosciences* 20, no.10, p. 1455-1465.

Appendixes

Appendix A—Building Data Model

This data dictionary details the components used to model the interior of a building in GIS. It includes the basic features of a building such as the building footprint, floors, rooms, hallways, walls, and doors. It also includes derived data for purposes of visualizing building such as buffered walls and doors. Lastly it contains the components for a 3D transportation network that is derived through geoprocessing from the core building components.

Components of the Source Data Model

Name: Building Footprints
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the outer edge of the building where it meet the surface of the ground.

Field Name:	BldgID
Field Alias:	Building Identifier
Data Type:	Small Integer
Width:	10
Precision:	NA
Scale:	NA

Field Name:	Owner
Field Alias:	Owner
Data Type:	Text
Width:	50
Precision:	NA
Scale:	NA

Field Name:	Address
Field Alias:	Address
Data Type:	Text
Width:	255
Precision:	NA
Scale:	NA

Field Name:	SqrFootage
Field Alias:	Square Footage
Data Type:	Float
Width:	24
Precision:	2
Scale:	NA

Field Name: ConstructDate
Field Alias: Construction Date
Data Type: Date
Width: NA
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Floors
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the shape of each floor at its base.

Field Name: FloorID
Field Alias: Floor Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: SqrFootage
Field Alias: Square Footage
Data Type: Float
Width: 24
Precision: 2
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Rooms
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the shape of each room, defined by center lines of the rooms walls, windows, and doors. They must be able to store a field that represents what type of room they are, including support for existing typologic systems such as the Omniclass code system.

Field Name: RoomID
Field Alias: Room Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: SqrFootage
Field Alias: Square Footage
Data Type: Float
Width: 24
Precision: 2
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Walls
Format: Feature Class
Geometry: Line
Description: The lines representing the centerlines of walls, at their base.

Field Name: WallID
Field Alias: Wall Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Width
Field Alias: Width
Data Type: Float
Width: 12
Precision: 2
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Doors
Format: Feature Class
Geometry: Line
Description: The lines representing the centerlines of doors, at their base.

Field Name: WallID
Field Alias: Wall Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: BldgFurn
Format: Building Furniture
Geometry: Point
Description: The points representing the furniture inside of buildings.

Field Name: FurnID
Field Alias: Furniture Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Elevation
Format: Table
Geometry: NA
Description: This table stores all the elevation information for all features in the source data model. Features are related to this table using their elevation identifier. This allows for quick editing of elevation or extrusion for a collection of features across the model.

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: BHeight
Field Alias: Base Height
Data Type: Float
Width: 24
Precision: 2
Scale: NA

Field Name: Extrusion
Field Alias: Extrusion
Data Type: Float
Width: 24
Precision: 2
Scale: NA

Components of the Visualization Data Model

Name: Buffered Walls
Format: Feature Class
Geometry: Polygon
Description: The wall lines buffered by their width, with square capped ends.

Field Name: WallID
Field Alias: Wall Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Buffered Doors
Format: Feature Class
Geometry: Polygon
Description: The door lines buffered by their width, with square capped ends.

Field Name: DoorID
Field Alias: Door Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Room Interiors
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the shape of each room, defined by the interior of the walls and windows, and the center lines of doors. They are generated by subtracting the buffered walls and windows from each room to get their actual interior space.

Field Name: RoomID
Field Alias: Room Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: Type
Field Alias: Type
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: SqrFootage
Field Alias: Square Footage
Data Type: Float
Width: 24
Precision: 2
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Hallways
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the hallways of a building, the transit corridors between rooms. They are generated by subtracting the rooms, and buffered walls from the floor polygons.

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Name: Ceilings
Format: Feature Class
Geometry: Polygon
Description: These polygons represent the false ceilings of each room and hallway. They are generated by combing the room and hallway polygons, and using the base elevations and extrusion attributes from the Elevation table combined to represent the heights of the ceiling.

Field Name: FloorNum
Field Alias: Floor Number
Data Type: Text
Width: 50
Precision: NA
Scale: NA

Field Name: ElevID
Field Alias: Elevation Identifier
Data Type: Small Integer
Width: 10
Precision: NA
Scale: NA

Components of the Network Data Model

Name: Floor Transportation Network
Format: Feature Class
Geometry: Line
Description: These lines represent the transportation network across the floor of a building. In order to effectively represent transportation across multiple floors of a building, one network per floor is required.

Field Name:	Length
Field Alias:	Length
Data Type:	Float
Width:	12
Precision:	2
Scale:	NA

Field Name:	Impedance
Field Alias:	Impedance
Data Type:	Float
Width:	12
Precision:	2
Scale:	NA

Name: Vertical Nodes
Format: Feature Class
Geometry: Point
Description: These points represent points where someone can change floor, and hence transportation network. Inside a building the can represent stairs, elevators, or escalators.

Field Name:	Type
Field Alias:	Type
Data Type:	Text
Width:	50
Precision:	NA
Scale:	NA

Field Name:	ElevID
Field Alias:	Elevation Identifier
Data Type:	Small Integer
Width:	10
Precision:	NA
Scale:	NA

Appendix B—ArchiCAD Extension

This extension to ArchiCAD allows you to convert BIM features into feature classes in the geodatabase. It does so by exporting the BIM to an an ascii file format designed to help ArchiCAD users share their building information. This ASCII format is then parsed by the code, and it components distributed into different feature collections in a shapefile. So for instance all the window features would end up in a single shapefile. These shapefile can then be imported into the geodatabase. It should be noted however the the shapefile format does not support textures, so any texture information of the object in ArchiCAD will be lost during the process.

```
#!/usr/bin/env python
"""Encode the contents of a bim directory into a MIME message file.
Usage: writeBIMfile [options] outputBIMfilePath
Options:
    -h / --help
        Print this message and exit.
    -d directory
    --directory=directory
        Mail the contents of the specified directory, otherwise use the
        current directory.  Only the regular files in the directory are
        sent, and we don't recurse to subdirectories.

`from' is the email address of the sender of the message.

`"""

import sys
import os
import getopt
import smtplib
# For guessing MIME type based on file name extension
import mimetypes

from email import Encoders
from email.Message import Message
from email.MIMEAudio import MIMEAudio
from email.MIMEBase import MIMEBase
from email.MIMEMultipart import MIMEMultipart
from email.MIMEImage import MIMEImage
from email.MIMEText import MIMEText

COMMASPACE = ', '


def usage(code, msg=''):
    print >> sys.stderr, __doc__
    if msg:
        print >> sys.stderr, msg
    sys.exit(code)
```

```
def main():
    try:
        opts, args = getopt.getopt(sys.argv[1:], 'hd:', ['help',
'directory='])
    except getopt.error, msg:
        usage(1, msg)

    dir = os.curdir
    for opt, arg in opts:
        if opt in ('-h', '--help'):
            usage(0)
        elif opt in ('-d', '--directory'):
            dir = arg

    if len(args) < 1:
        usage(1)

    outputBIMfilePath = args[0]

    # Create the enclosing (outer) message
    outer = MIMEMultipart()
    outer['Subject'] = 'Contents of BIM directory %s' %
os.path.abspath(dir)
    outer.preamble = 'This file contains a building information model.\n'
    # To guarantee the message ends with a newline
    outer.epilogue = ''

    for filename in os.listdir(dir):
        path = os.path.join(dir, filename)
        if not os.path.isfile(path):
            continue
        # Guess the content type based on the file's extension.  Encoding
        # will be ignored, although we should check for simple things like
        # gzip'd or compressed files.
        ctype, encoding = mimetypes.guess_type(path)
        if ctype is None or encoding is not None:
            # No guess could be made, or the file is encoded (compressed),
so
            # use a generic bag-of-bits type.
            ctype = 'application/octet-stream'
        maintype, subtype = ctype.split('/', 1)
        if maintype == 'text':
            fp = open(path)
            # Note: we should handle calculating the charset
            msg = MIMEText(fp.read(), _subtype=subtype)
            fp.close()
        elif maintype == 'image':
            fp = open(path, 'rb')
            msg = MIMEImage(fp.read(), _subtype=subtype)
            fp.close()
        elif maintype == 'audio':
            fp = open(path, 'rb')
            msg = MIMEAudio(fp.read(), _subtype=subtype)
            fp.close()
        else:
```

```
            fp = open(path, 'rb')
            msg = MIMEBase(maintype, subtype)
            msg.set_payload(fp.read())
            fp.close()
            # Encode the payload using Base64
            Encoders.encode_base64(msg)
        # Set the filename parameter
        msg.add_header('Content-Disposition', 'attachment',
filename=filename)
        outer.attach(msg)

    # Now send the message
    #s = smtplib.SMTP()
    #s.connect()
    #s.sendmail(sender, recips, outer.as_string())
    #s.close()
    print(outer.as_string());

    #create an output file
    outfile = open(outputBIMfilePath, 'w')
    outfile.write(outer.as_string());
    outfile.close();


if __name__ == '__main__':
    main()


'''-----------------------------------------------------------------------

 Tool Name:     ReadFeaturesFromBIMTextFile

 Source Name:   ReadFeaturesFromBIMTextFile.py

 Version:       0.1

 Author:        Dr. Michael Flaxman

                Environmental Systems Research Institute Inc.

 Required Argumuments:
      An Input Polygon "Generate" File containing feature coordinates
      An output feature class

 Description:   Reads a set of text file in "Simple Building Information
Model" format and creates a feature class representing either a building
zone, floor building or site polygon.  The resultant feature class type is
dependent on the attribute information in the input .csv file

-----------------------------------------------------------------------'''



import string, os, sys, locale, arcgisscripting
```

```
gp = arcgisscripting.create()

gp.overwriteoutput = 1



msgErrorTooFewParams = "Not enough parameters provided."

msgUnknownDataType = " is not a valid datatype. Datatype must be point,
multipoint, polyline or polygon."

msgErrorCreatingPoint = "Error creating point %s on feature %s"



# sets all the point properties

def createPoint(point, geometry):

    try:

        point.id = geometry[0]

        point.x = geometry[1]

        point.y = geometry[2]

        # When empty values are written out from pyWriteGeomToTextFile,
they come as 1.#QNAN

        # Additionally, the user need not supply these values, so if they
aren't in the list don't add them

        if len(geometry) > 3:

            if geometry[3] != "1" + sepchar + "#QNAN": point.z =
geometry[3]

        if len(geometry) > 4:

            if geometry[4] != "1" + sepchar + "#QNAN": point.m =
geometry[4]

        return point

    except:

        raise Exception, msgErrorCreatingPoint




try:
```

```
# make sure enough arguments are provided
if len(sys.argv) < 3:
    raise Exception, msgErrorTooFewParams

# get the provided parameters
inputTxtFile = open(sys.argv[1])
fileSepChar = "."
outputFC = sys.argv[2]

# attribute file handling
attribFilePath = str(os.path.split(sys.argv[0]))
print (attribFilePath)
attribFileName = str(os.path.split(sys.argv[1]) + ".csv")
print (attribFileName)
attribFileFullPath = attribFilePath + attribFileName
print (attribFileFullPath)
attribFile = open(attribFileFullPath)
if (attribFile == None):
    print("Problem opening .csv file\n")

# spatial reference is optional
if len(sys.argv) > 3:
    outputSR = sys.argv[3]
else:
    outputSR = "#"

# make sure the text type specified in the text file is valid.
inDataType = inputTxtFile.readline().replace("\n","").lower()
```

```
    dataTypes = ["polyline", "polygon"]

    if inDataType.lower() not in dataTypes:

        msgUnknownDataType = "%s%s" % (inDataType, msgUnknownDataType)

        raise Exception, msgUnknownDataType


    # create the new featureclass

    gp.toolbox = "management"

    gp.CreateFeatureclass(os.path.split(outputFC)[0],
os.path.split(outputFC)[1], inDataType, "#", "ENABLED", "ENABLED",
outputSR)


    # read header line from csv file to determine which fields need
creation

    # problem: no field typing, and for non-standard field names don't
know

    # (might need to infer from values...yuch)

    headerLine = attribFile.readlines()

    fieldNames = attribLine.split(",")

    #for fieldName in fieldNames:

    #    gp.addfield(outputFC, fieldName, "LONG")

    # create a new field to assure the id of each feature is preserved.

    idfield = "PolyID"

    gp.addfield(outputFC, idfield, "LONG")

    # get some information about the new featureclass for later use.

    outDesc = gp.describe(outputFC)

    shapefield = outDesc.ShapeFieldName

    # create the cursor and objects necessary for the geometry creation

    rows = gp.insertcursor(outputFC)

    pnt = gp.createobject("point")
```

```
    pntarray = gp.createobject("Array")

    partarray = gp.createobject("Array")


    locale.setlocale(locale.LC_ALL, '')

    sepchar = locale.localeconv()['decimal_point']


    # loop through the text file.

    featid = 0

    lineno = 1

    for line in inputTxtFile.readlines():

        lineno += 1


        #read attribute line from csv file

        #to test: comma space breaks it?

        attribLine = attribFile.readlines()

        attribs = attribLine.split(",")


        # create an array from each line in the input text file

        values = line.replace("\n", "").replace(fileSepChar,
sepchar).split(" ")


        # for a point feature class simply populate a point object and
insert it.

        if inDataType ==  "polygon" or inDataType == "polyline":

            #takes care of

            #adds the point array to the part array and then part array to
the feature

            if (len(values) == 2 and float(values[1]) == 0 and lineno !=
2) or values[0].lower() == "end":

                partarray.add(pntarray)
```

```
                row = rows.newrow()

                row.SetValue(shapefield, partarray)

                # store the feature id just in case there is an error.
helps track down the offending line in the input text file.

                if values[0].lower() != "end":

                    row.SetValue(idfield, featid)

                    featid = int(values[0])

                else:

                    row.SetValue(idfield, featid)

                rows.insertrow(row)

                partarray.removeall()

                pntarray.removeall()

            #adds parts and/or interior rings to the part array

            elif (len(values) == 2 and float(values[1]) > 0) or
values[0].lower() == "interiorring":

                partarray.add(pntarray)

                pntarray.removeall()

            #add points to the point array

            elif len(values) > 2:

                pnt = createPoint(pnt, values)

                pntarray.add(pnt)

            elif (len(values) == 2 and lineno == 2):

                featid = int(values[0])


    inputTxtFile.close()

    del rows

    del row
```

```
except Exception, ErrorDesc:

    # handle the errors here. if the point creation fails, want to keep
track of which point failed (easier to fix the

    # text file if we do)

    if ErrorDesc[0] == msgErrorCreatingPoint:

        if inDataType.lower() == "point":

            msgErrorCreatingPoint = msgErrorCreatingPoint % (values[0],
values[0])

        else:

            msgErrorCreatingPoint = msgErrorCreatingPoint % (values[0],
featid)

        gp.AddError(msgErrorCreatingPoint)

    elif ErrorDesc[0] != "":

        gp.AddError(str(ErrorDesc))


    gp.AddError(gp.getmessages(2))


    # make sure to close up the fileinput no matter what.

    if inputTxtFile: inputTxtFile.close()

    #if attribFile: attribFile.close()
```

Appendix C—Network Generation Scripts

This python script is used to generate a transportation network based upon the data model outlined in Appendix A. What it does is convert the walls of the building, for each floor, into a raster data set. Those walls are given null values, hence any cost distance analysis will identify them as un-crossable. For each office in the building, a point at it's center is generated. The model then does cost distance analysis across the raster from every office to every other office. The result is collection of least-cost path based routes from every room to every other room. These rasters are then vectorized, and combined together, to create a comprehensive, if very crude, transportation network for each floor of the building.

```
# ---------------------------------------------------------------------------
# NetworkGenerator.py
# Created on: Wed Oct 31 2007 01:11:32 PM
# Generates a transportation network for floor one of Esri Buildings M & N
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting

# Create a GP object, set product code, check out licenses
gp = arcgisscripting.create()
gp.SetProduct("ArcInfo")
gp.CheckOutExtension("spatial")
gp.CheckOutExtension("3D")

# Load required toolboxes...
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Spatial
Analyst Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Conversion
Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Data
Management Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/3D Analyst
Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Analysis
Tools.tbx")

# Set the Geoprocessing environment...
gp.scratchWorkspace = "C:\\data\\MIPIII\\MnNDemo.gdb"
gp.outputCoordinateSystem =
"PROJCS['NAD_1983_UTM_Zone_11N',GEOGCS['GCS_North_American_1983',DATUM['D_
North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM[
'Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse
_Mercator'],PARAMETER['False_Easting',500000.0],PARAMETER['False_Northing'
,0.0],PARAMETER['Central_Meridian',-
117.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0
],UNIT['Meter',1.0]],VERTCS['WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_19
84',6378137.0,298.257223563]],PARAMETER['Vertical_Shift',0.0],PARAMETER['D
irection',1.0],UNIT['Meter',1.0]]"
gp.cellSize = "0.1"
```

```
gp.workspace = "C:\\data\\MIPIII\\MnNDemo.gdb"

# Local variables...
MnNLvl2 = "C:\\data\\MIPIII\\MnNDemo.gdb\\MnNLvl1"
Rooms = "C:\\data\\MIPIII\\MnNDemo.gdb\\Rooms"
RoomPnts = "C:\\data\\MIPIII\\MnNDemo.gdb\\RoomPnts"
JustWalls = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWalls"
SrcPnt = "C:\\data\\MIPIII\\MnNDemo.gdb\\SrcPnt"
DestTmp1 = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1"
DestTmp2 = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2"
DestPnts = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestPnts"
CostDist = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostDist"
Backlink = "C:\\data\\MIPIII\\MnNDemo.gdb\\Backlink"
JustWallsGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsGRD"
JustWallsBuf = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsBuf"
JustWallsRcl = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsRcl"
CostSurface = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostSurface"
BldgBuf = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgBuf"
BldgDis = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgDis"
BldgPoly = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgPoly"
BldgGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgGRD"
BldgRcl = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgRcl"
CostPathGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathGRD"
MnNDemo_gdb = "C:\\data\\MIPIII\\MnNDemo.gdb"
CostPathLine = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathLine"
CostPathRSP = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathRSP"

# The list of rooms to calculate paths from
a = [2, 4, 30, 50, 71, 100, 150, 202, 234, 236]

# Process: Feature To Polygon...
gp.FeatureToPolygon_management("C:\\data\\MIPIII\\MnNDemo.gdb\\MnNLvl1",
Rooms, "", "ATTRIBUTES", "")

# Process: Feature To Point...
gp.FeatureToPoint_management(Rooms, RoomPnts, "CENTROID")

# Process: Buffer (2)...
gp.Buffer_analysis(MnNLvl2, BldgBuf, "1.5 Meters", "FULL", "ROUND", "ALL",
"")

# Process: Feature To Polygon (2)...
gp.FeatureToPolygon_management("C:\\data\\MIPIII\\MnNDemo.gdb\\BldgBuf",
BldgPoly, "", "ATTRIBUTES", "")

# Process: Dissolve...
gp.Dissolve_management(BldgPoly, BldgDis, "", "", "MULTI_PART",
"DISSOLVE_LINES")

# Process: Feature to Raster (3)...
gp.FeatureToRaster_conversion(BldgDis, "Value", BldgGRD, "0.313519")

# Process: Reclassify (3)...
gp.Reclassify_3d(BldgGRD, "VALUE", "1 10000 1", BldgRcl, "DATA")
```

```
# Process: Select...
gp.Select_analysis(MnNLvl2, JustWalls, "\"TYPE\" = 1")

# Process: Buffer...
gp.Buffer_analysis(JustWalls, JustWallsBuf, "0.2 Meters", "FULL", "ROUND",
"NONE", "")

# Process: Feature to Raster...
gp.FeatureToRaster_conversion(JustWallsBuf, "OBJECTID", JustWallsGRD,
"0.1")

# Process: Reclassify...
gp.Reclassify_3d(JustWallsGRD, "VALUE", "0 1000000 NODATA;NODATA 0",
JustWallsRcl, "DATA")

# Process: Plus...
gp.Plus_sa(BldgRcl, JustWallsRcl, CostSurface)

# Process: Create Raster Dataset...
gp.CreateRasterDataset_management(MnNDemo_gdb, "CostPathGRD", "0.1",
"8_BIT_UNSIGNED",
"PROJCS['NAD_1983_UTM_Zone_11N',GEOGCS['GCS_North_American_1983',DATUM['D_
North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM[
'Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse
_Mercator'],PARAMETER['False_Easting',500000.0],PARAMETER['False_Northing'
,0.0],PARAMETER['Central_Meridian',-
117.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0
],UNIT['Meter',1.0]],VERTCS['WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_19
84',6378137.0,298.257223563]],PARAMETER['Vertical_Shift',0.0],PARAMETER['D
irection',1.0],UNIT['Meter',1.0]]", "1", "", "PYRAMIDS -1 NEAREST", "128
128", "LZ77", "-5120763.26772381 9997964.23583161")

# The loop that builds the cost paths for each of the selected rooms
for FirstRoom in a:

      Expression = "\"ORIG_FID\" = %d" % FirstRoom
      Expression__2_ = "\"ORIG_FID\" > %d" % FirstRoom
      Expression__3_ = "\"ORIG_FID\" < %d" % FirstRoom
      CostPath = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPath%d" % FirstRoom
      CostPathLine = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathLine%d" %
FirstRoom
      DestPnts = "DstPnts%d" % FirstRoom

      # Process: Select (2)...
      gp.Select_analysis(RoomPnts, SrcPnt, Expression)

      # Process: Select (3)...
      gp.Select_analysis(RoomPnts, DestTmp1, Expression__2_)

      # Process: Select (4)...
      gp.Select_analysis(RoomPnts, DestTmp2, Expression__3_)

      # Process: Merge...
      gp.Merge_management("C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1;C:\\dat
a\\MIPIII\\MnNDemo.gdb\\DestTmp2", DestPnts, "Shape_length 'Shape_length'
```

```
true true false 0 Double 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,Shape_length,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,Shape_length,-1,-1;Shape_area
'Shape_area' true true false 0 Double 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,Shape_area,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,Shape_area,-1,-1;ORIG_FID
'ORIG_FID' true true false 0 Long 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,ORIG_FID,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,ORIG_FID,-1,-1")

      # Process: Cost Distance...
      gp.CostDistance_sa(SrcPnt, CostSurface, CostDist, "", Backlink)

      # Process: Cost Path...
      gp.CostPath_sa(DestPnts, CostDist, Backlink, CostPath, "EACH_CELL",
"ORIG_FID")

      # Process: Raster to Polyline...
      # gp.RasterToPolyline_conversion(CostPath, CostPathLine, "ZERO",
"0", "SIMPLIFY", "VALUE")

      # Clean up data for this run
      gp.Delete_management(DestTmp1, "FeatureClass")
      gp.Delete_management(DestTmp2, "FeatureClass")
      gp.Delete_management(SrcPnt, "FeatureClass")
      gp.Delete_management(DestPnts, "FeatureClass")
      gp.Delete_management(Backlink, "RasterDataset")
      gp.Delete_management(CostDist, "RasterDataset")
      # gp.Delete_management(CostPath, "RasterDataset")

      # Process: Mosaic...
      gp.Mosaic_management(CostPath, CostPathGRD, "LAST", "FIRST", "", "",
"NONE", "0", "NONE")
      gp.Delete_management(CostPath, "RasterDataset")

# Process: Raster to Polyline...
gp.Resample_management(CostPathGRD, CostPathRSP, "0.2", "NEAREST")
gp.RasterToPolyline_conversion(CostPathRSP, CostPathLine, "ZERO", "0",
"SIMPLIFY", "Value")

# ---------------------------------------------------------------------------
# NetworkGenerator.py
# Created on: Wed Oct 31 2007 01:11:32 PM
# Generates a transportation network for floor two of Esri Buildings M & N
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, os, arcgisscripting

# Create a GP object, set product code, check out licenses
gp = arcgisscripting.create()
gp.SetProduct("ArcInfo")
gp.CheckOutExtension("spatial")
gp.CheckOutExtension("3D")
```

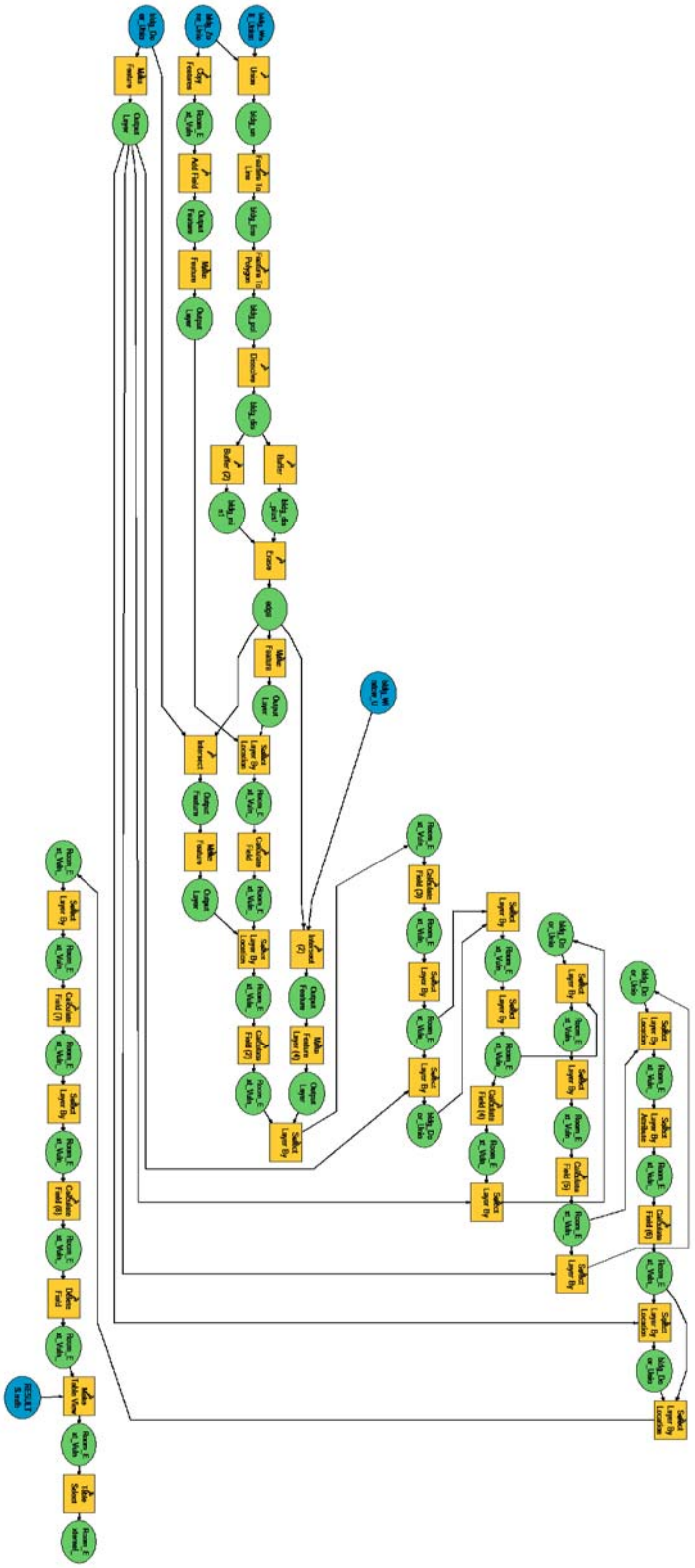
```
# Load required toolboxes...
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Spatial
Analyst Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Conversion
Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Data
Management Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/3D Analyst
Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Analysis
Tools.tbx")

# Set the Geoprocessing environment...
gp.scratchWorkspace = "C:\\data\\MIPIII\\MnNDemo.gdb"
gp.outputCoordinateSystem =
"PROJCS['NAD_1983_UTM_Zone_11N',GEOGCS['GCS_North_American_1983',DATUM['D_
North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM[
'Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse
_Mercator'],PARAMETER['False_Easting',500000.0],PARAMETER['False_Northing'
,0.0],PARAMETER['Central_Meridian',-
117.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0
],UNIT['Meter',1.0]],VERTCS['WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_19
84',6378137.0,298.257223563]],PARAMETER['Vertical_Shift',0.0],PARAMETER['D
irection',1.0],UNIT['Meter',1.0]]"
gp.cellSize = "0.1"
gp.workspace = "C:\\data\\MIPIII\\MnNDemo.gdb"

# Local variables...
MnNLvl2 = "C:\\data\\MIPIII\\MnNDemo.gdb\\MnNLvl2"
Rooms = "C:\\data\\MIPIII\\MnNDemo.gdb\\Rooms"
RoomPnts = "C:\\data\\MIPIII\\MnNDemo.gdb\\RoomPnts"
JustWalls = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWalls"
SrcPnt = "C:\\data\\MIPIII\\MnNDemo.gdb\\SrcPnt"
DestTmp1 = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1"
DestTmp2 = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2"
DestPnts = "C:\\data\\MIPIII\\MnNDemo.gdb\\DestPnts"
CostDist = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostDist"
Backlink = "C:\\data\\MIPIII\\MnNDemo.gdb\\Backlink"
JustWallsGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsGRD"
JustWallsBuf = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsBuf"
JustWallsRcl = "C:\\data\\MIPIII\\MnNDemo.gdb\\JustWallsRcl"
CostSurface = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostSurface"
BldgBuf = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgBuf"
BldgDis = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgDis"
BldgPoly = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgPoly"
BldgGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgGRD"
BldgRcl = "C:\\data\\MIPIII\\MnNDemo.gdb\\BldgRcl"
CostPathGRD = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathGRD"
MnNDemo_gdb = "C:\\data\\MIPIII\\MnNDemo.gdb"
CostPathLine = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathLine"
CostPathRSP = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathRSP"

# The list of rooms to calculate paths from
a = [1, 61, 72, 99, 114, 190, 224, 256]
```

```
# Process: Feature To Polygon...
gp.FeatureToPolygon_management("C:\\data\\MIPIII\\MnNDemo.gdb\\MnNLvl2",
Rooms, "", "ATTRIBUTES", "")

# Process: Feature To Point...
gp.FeatureToPoint_management(Rooms, RoomPnts, "CENTROID")

# Process: Buffer (2)...
gp.Buffer_analysis(MnNLvl2, BldgBuf, "1.5 Meters", "FULL", "ROUND", "ALL",
"")

# Process: Feature To Polygon (2)...
gp.FeatureToPolygon_management("C:\\data\\MIPIII\\MnNDemo.gdb\\BldgBuf",
BldgPoly, "", "ATTRIBUTES", "")

# Process: Dissolve...
gp.Dissolve_management(BldgPoly, BldgDis, "", "", "MULTI_PART",
"DISSOLVE_LINES")

# Process: Feature to Raster (3)...
gp.FeatureToRaster_conversion(BldgDis, "Value", BldgGRD, "0.313519")

# Process: Reclassify (3)...
gp.Reclassify_3d(BldgGRD, "VALUE", "1 10000 1", BldgRcl, "DATA")

# Process: Select...
gp.Select_analysis(MnNLvl2, JustWalls, "\"TYPE\" = 1")

# Process: Buffer...
gp.Buffer_analysis(JustWalls, JustWallsBuf, "0.2 Meters", "FULL", "ROUND",
"NONE", "")

# Process: Feature to Raster...
gp.FeatureToRaster_conversion(JustWallsBuf, "OBJECTID", JustWallsGRD,
"0.1")

# Process: Reclassify...
gp.Reclassify_3d(JustWallsGRD, "VALUE", "0 1000000 NODATA;NODATA 0",
JustWallsRcl, "DATA")

# Process: Plus...
gp.Plus_sa(BldgRcl, JustWallsRcl, CostSurface)

# Process: Create Raster Dataset...
gp.CreateRasterDataset_management(MnNDemo_gdb, "CostPathGRD", "0.1",
"8_BIT_UNSIGNED",
"PROJCS['NAD_1983_UTM_Zone_11N',GEOGCS['GCS_North_American_1983',DATUM['D_
North_American_1983',SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM[
'Greenwich',0.0],UNIT['Degree',0.0174532925199433]],PROJECTION['Transverse
_Mercator'],PARAMETER['False_Easting',500000.0],PARAMETER['False_Northing'
,0.0],PARAMETER['Central_Meridian',-
117.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0
],UNIT['Meter',1.0]],VERTCS['WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_19
84',6378137.0,298.257223563]],PARAMETER['Vertical_Shift',0.0],PARAMETER['D
```

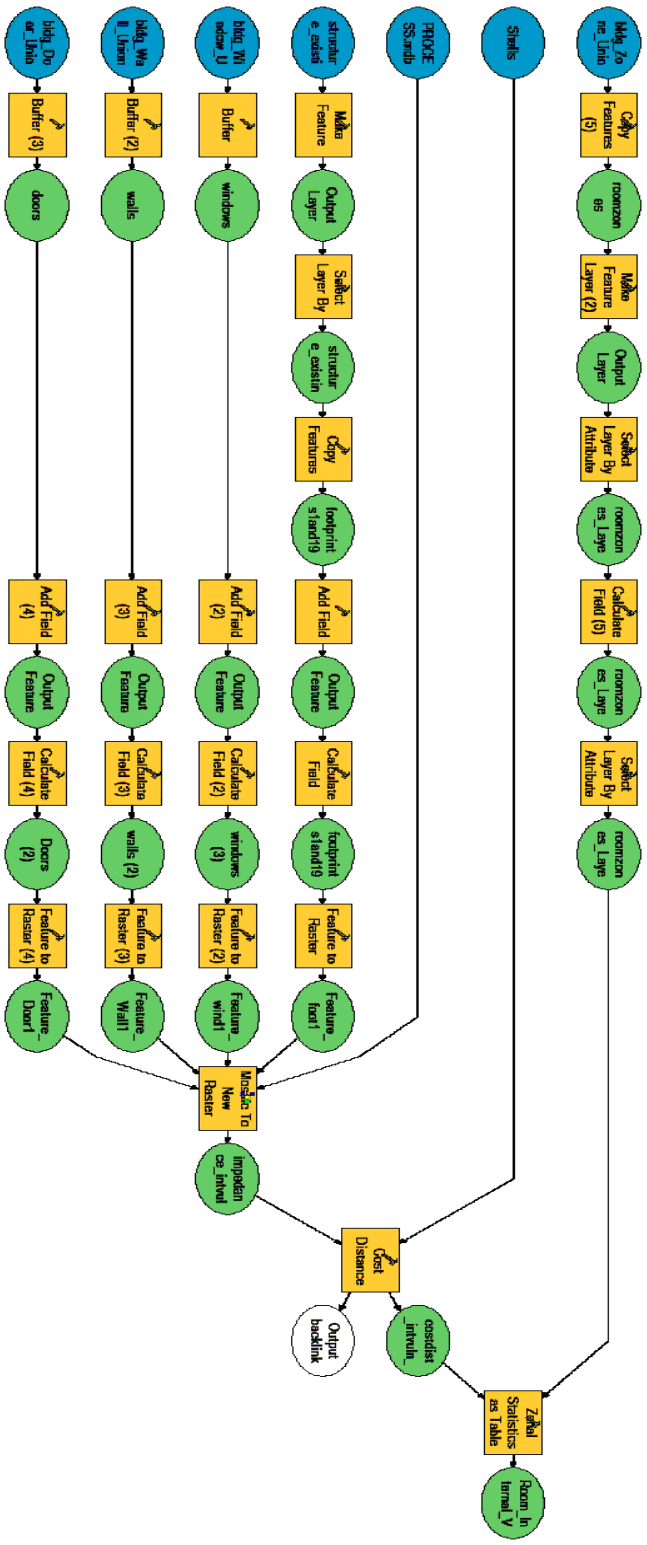
```
irection',1.0],UNIT['Meter',1.0]]", "1", "", "PYRAMIDS -1 NEAREST", "128
128", "LZ77", "-5120763.26772381 9997964.23583161")

# The loop that builds the cost paths for each of the selected rooms
for FirstRoom in a:

      Expression = "\"ORIG_FID\" = %d" % FirstRoom
      Expression__2_ = "\"ORIG_FID\" > %d" % FirstRoom
      Expression__3_ = "\"ORIG_FID\" < %d" % FirstRoom
      CostPath = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPath%d" % FirstRoom
      CostPathLine = "C:\\data\\MIPIII\\MnNDemo.gdb\\CostPathLine%d" %
FirstRoom
      DestPnts = "DstPnts%d" % FirstRoom

      # Process: Select (2)...
      gp.Select_analysis(RoomPnts, SrcPnt, Expression)

      # Process: Select (3)...
      gp.Select_analysis(RoomPnts, DestTmp1, Expression__2_)

      # Process: Select (4)...
      gp.Select_analysis(RoomPnts, DestTmp2, Expression__3_)

      # Process: Merge...
      gp.Merge_management("C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1;C:\\dat
a\\MIPIII\\MnNDemo.gdb\\DestTmp2", DestPnts, "Shape_length 'Shape_length'
true true false 0 Double 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,Shape_length,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,Shape_length,-1,-1;Shape_area
'Shape_area' true true false 0 Double 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,Shape_area,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,Shape_area,-1,-1;ORIG_FID
'ORIG_FID' true true false 0 Long 0 0
,First,#,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp1,ORIG_FID,-1,-
1,C:\\data\\MIPIII\\MnNDemo.gdb\\DestTmp2,ORIG_FID,-1,-1")

      # Process: Cost Distance...
      gp.CostDistance_sa(SrcPnt, CostSurface, CostDist, "", Backlink)

      # Process: Cost Path...
      gp.CostPath_sa(DestPnts, CostDist, Backlink, CostPath, "EACH_CELL",
"ORIG_FID")

      # Process: Raster to Polyline...
      # gp.RasterToPolyline_conversion(CostPath, CostPathLine, "ZERO",
"0", "SIMPLIFY", "VALUE")

      # Clean up data for this run
      gp.Delete_management(DestTmp1, "FeatureClass")
      gp.Delete_management(DestTmp2, "FeatureClass")
      gp.Delete_management(SrcPnt, "FeatureClass")
      gp.Delete_management(DestPnts, "FeatureClass")
      gp.Delete_management(Backlink, "RasterDataset")
      gp.Delete_management(CostDist, "RasterDataset")
      # gp.Delete_management(CostPath, "RasterDataset")
```

```
      # Process: Mosaic...
      gp.Mosaic_management(CostPath, CostPathGRD, "LAST", "FIRST", "", "",
"NONE", "0", "NONE")
      gp.Delete_management(CostPath, "RasterDataset")

# Process: Raster to Polyline...
gp.Resample_management(CostPathGRD, CostPathRSP, "0.2", "NEAREST")
gp.RasterToPolyline_conversion(CostPathRSP, CostPathLine, "ZERO", "0",
"SIMPLIFY", "Value")
```

Appendix D — Vulnerability model



Appendix E — Hazard Model



Appendix F — Accessibility model