Complex Terrain Databases for Urban Operations

Dr. Stephen J. Adelson
Leo Salemann
Steve Farsai
Dr. Dale D. Miller
Timothy Miller
Melissa E. Nakanishi
Lockheed Martin Simulation, Training & Support
Advanced Simulation Center (ASC)
3605 132nd Ave. SE, Suite 400
Bellevue, WA 98006
425-957-3209

Julio de la Cruz
SFC Paul Ray Smith Simulation and Training Technology Center
Army RDECOM/STTC
12423 Research Parkway
Orlando, Florida 32826
407 384-3733

sadelson@lads.is.lmco.com, lsaleman@lads.is.lmco.com, stevefarsai@sprintmail.com, dale.d.miller@lmco.com, tmiller@lads.is.lmco.com, mnakanishi@lads.is.lmco.com, julio.delacruz@us.army.mil

Keywords:
Synthetic environment, SNE, urban operations, databases, simulation

ABSTRACT: *Simulation of Urban Operations (UO) requires the ability to create and damage dense, realistic cityscapes. However, since these databases must be both large and complex, creating appropriate environments for UO is difficult at best. For example, a system capable of creating one building per minute would need nearly 42 months to generate a present-day ultra-high resolution building (UHRB) database of 1.8 million buildings. Size and complexity of urban databases will undoubtedly continue to increase in the future.*

During the simulation, a method must be found to damage buildings and other structures in real time, with visual as well as semantic (e.g., enclosure / aperture) modifications. Ideally, support would be included for tactical explosives as well as large-scale damage from artillery. Rubble and structural integrity should also be appropriately modeled for their effects on UO missions.

In this paper, we present our Automated Building Generation System (ABGS) and modification systems (UHRB-Sim), representing initial steps towards these goals. Our system can generate simple buildings in under a second and damage them in real time, leveraging our experience with dynamic terrain effects. We will review our methodology and present a number of future directions to improve the current work.

1 Introduction

One of the reasons why the simulation of Urban Operations (UO) is a difficult and open problem is that the environment the participants require is extremely complex, richly attributed, and polygonally intensive [1]. To our knowledge, only very small databases (in terms of number of buildings) have been created to address these requirements, and nothing for the square blocks of structures (or, indeed, square *kilometers*) that might be needed for a robust simulation. We call these databases *ultra-high resolution building* (UHRB) databases. For discussion purposes, we will use as our target the LM STS Jakarta database, containing approximately 1,800,000 buildings, developed from geospecific and/or geotypical building footprints and urban terrain zone (UTZ) information. Two views of a portion of the database are shown below in Figure 1 and Figure 2.

There are two primary obstacles to UHRB databases: database generation and real-time database modification. Intricate structures must be compiled within a reasonable timeframe - days, not years. Damage to buildings needs to be reflected in real-time, not after minutes or hours of calculation. To address these issues, we have implemented our prototype UHRB system, promising scalable database solutions for UO.

In this paper, we will cover our UHRB process from start to finish. We first discuss the ShapeFile generation process, which will result in generalized descriptors of our attributed buildings. Following this, we discuss how the ShapeFiles are turned into floorplan files, in which building components (walls, rooms, etc.) are generically represented. Floorplan files are used to generate polygonal representations of the buildings, to be used, in turn, with integration in the visual and SAF databases. Both floorplan and polygonal files are used in the UHRB-Sim, our real-time building model simulator. Finally, we briefly discuss future directions to make this work a better representation of a realistic urban environment.

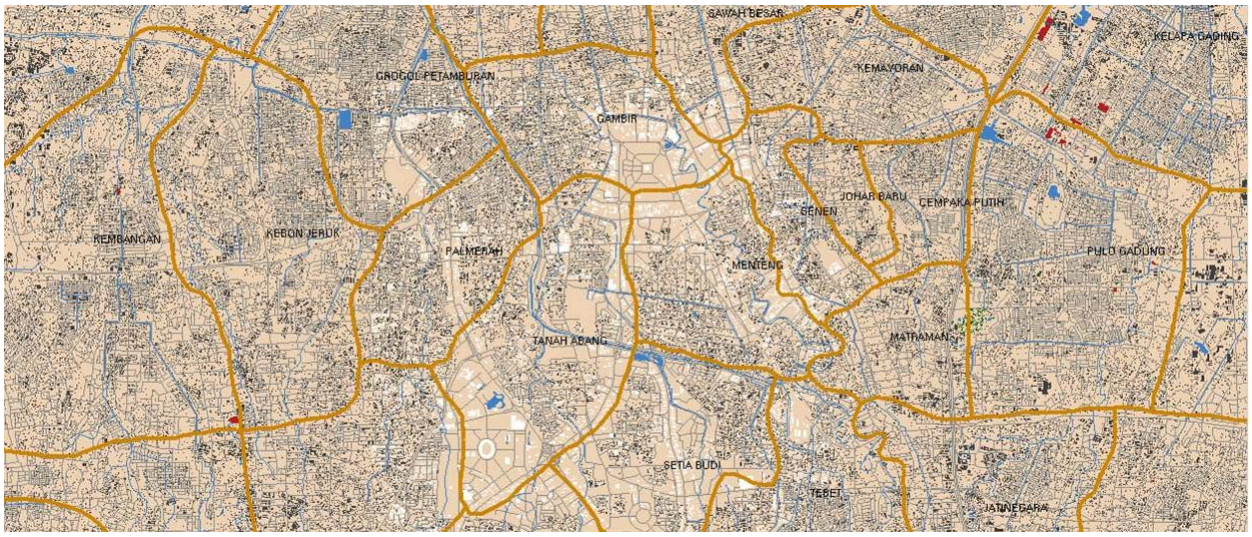

Figure 1: Jakarta View from JSAF Plan View Display


Figure 2: Jakarta Visualization

2 Automated Building Generation System (ABGS)

In current systems, compilation time of one minute per building (with interiors) is considered fast, particularly considering the complexity that a multi-story structure may entail. However, to render Jakarta on such a system would have required 30,000 computational hours, or nearly three and a half computational-*years* – and this excludes the minutes or hours of pre-computation time for design of each unique building. Clearly, this is not acceptable.

The Automated Building Generation System (ABGS) is able to generate buildings at a much faster rate. While the pre-computation time is still an issue to be addressed, our process addresses at least a large portion of the problem.

Typically, municipalities maintain GIS databases of building footprints for taxation purposes that are often available at a nominal cost. Generally, little other attribution (e.g., building height) is available. So while the ABGS can use geospecific building footprints, other attribution and the interior floorplans have been geotypically derived. The system, however, does not preclude the use of geospecific attribution and interior floorplans.

2.1 Overview

The ABGS consists of four major phases:

1. **Data Preparation**. Start with a set of building footprints, linear roads, terrain imagery and Urban Terrain Zone (UTZ) information.
2. **Attribute Generation**. Configure rules for allowable attribute value ranges and probabilistic distributions as a function of UTZ and building size and assign the values to the building outlines.
3. **FloorPlan Generation**. Read the 2D building ShapeFile, generate an "abstract" XML floorplan file for each building.

4. **Polygon Generation**. Read each 2D floorplan XML file; generate a 3D polygonal model in X3D format.

This process is illustrated below in Figure 3.

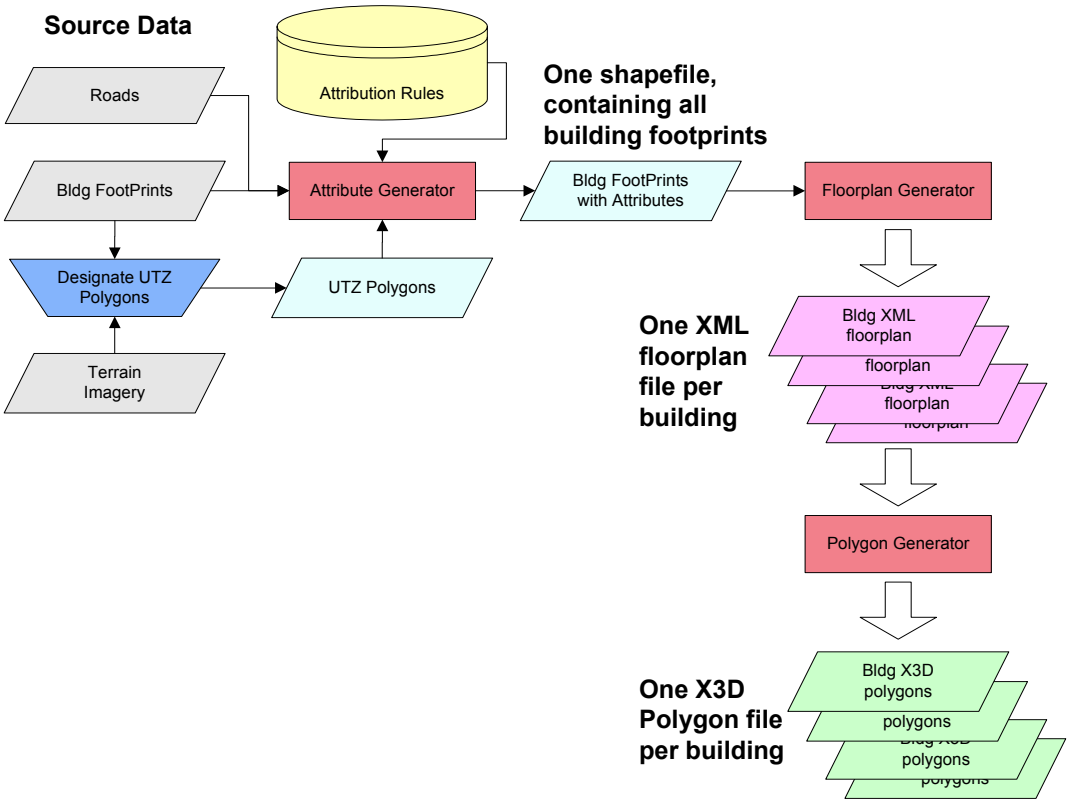

Figure 3: ABGS Process Flow

2.1.1 Building Representation

Several factors were key drivers in the choice of file formats used in the building representations throughout our ABGS process. During the requirements gathering and assessment phases of this project, we conducted a thorough investigation of available standard formats popular for representing models in the Modeling and Simulation (M&S) community.

The most significant factor in selecting standard file formats was the ability to leverage and quickly develop accompanying software and application programming interfaces (APIs) on Win32 as well as Linux platforms. The Win32 requirement was driven by the needs of the ABGS, which depends on an Open Database Connectivity (ODBC) connection, as well as several third-party tools which run best under Win32. The Linux requirement was driven by the needs of various compilers which translate the ABGS output into formats suitable for real-time visualization and constructive simulation.

The ABGS file formats also had to support a hierarchical structure, allowing us to model the relationships between buildings, floor levels, rooms, walls, and windows. Another priority was to be able to exchange data with the UHRB representation used by the OneSAF Objective System (OOS).

Given these requirements, the XML format and its emerging X3D specification [2] became the optimal formats. The XML floorplan format is a derivative of the XML floorplan format developed by OOS. Our current implementation produces one XML and one X3D data file per building in a spatially-organized folder tree on disk. In the future, we anticipate shifting this storage task to a relational database, where the XML and X3D data would be stored as binary large objects (BLOBs). Doing so will simplify configuration control during the ABGS process, and should allow faster access to individual buildings than regular file I/O. Also, several free and commercially available tools can convert from X3D to a variety of other formats, including VRML and OpenFlight.

We used the Altova XMLSpy product to create XML schemas for the two XML data formats (FloorPlan XML and PolyGen X3D) used in our process. This allowed us to automatically generate C++ classes and methods for each of the appropriate building components as represented in the XML structure for the FloorPlan and Polygon generator applications.

2.2 Data Preparation

The ABGS uses the following source data.

Building Footprints	Two-dimensional building outlines, typically in ShapeFile format.
Terrain Imagery	Needed if you do not have a complete set of Urban Terrain Zones (see below). The imagery must correlate with the building footprints. Resolution should be high enough to discern even small structures (one to five meter pixels).
Urban Terrain Zones	Large 2D polygons, designating "neighborhoods" of similar building function and density. Examples include clustered office buildings, open-set warehouses, and close-set houses. This data is usually manually derived from the Terrain Imagery, although semi-automated feature extraction may be used.
Road Network	Linear features, correlating with the building footprints. The roads are used for designating which wall of each building footprint will become the "front" and have the main entrance.

The attributes for the Building Footprints, Road Network, and Urban Terrain Zones are based on the Environmental Data Model (EDM) for Ultra-High Resolution Buildings (UHRB) [3]. The EDM provides a precise definition of UHRB features (floors, rooms, walls) and attributes (height, surface material, building function). The UHRB-Sim EDM is derived from the UHRB EDM for the One-SAF Objective System (OOS). Compared to the OOS UHRB EDM, the UHRB-Sim EDM has the following simplifications:

- Dropped Abstract and Generalized features in favor of a more explicit hierarchy.
- Reduced feature set to the core features necessary for a building with a basic interior.
- Reduced attribution to just those which support a visual representation.

2.3 Attribute Generation

The Attribute Generation Process consists of the following major steps, as shown in Figure 4.

1. **Generate Root Segments:** Designate one segment from each building footprint to become the "front."
2. **Configure Attribution Rules:** Configure allowable ranges for building height and function, based on Urban Terrain Zone and footprint size.
3. **Generate Attributes:** Run the Attribute Generator to assign attributes.
4. **Verification:** View the results, perform post-process edits if necessary.

We will examine each of these in more detail.

2.3.1 Generate Root Segments

The root segment denotes the "front" of the building, where the main entrance is to be placed. By convention, the root segment is defined as the first two vertices in the building footprint polygon. The ABGS includes a tool which takes the building footprints and the roads as input, and "re-orders" the vertices in the building footprints such that the first two will define the footprint segment which is closest to the nearest road (see Figure 5).

2.3.2 Configure Attribution Rules

The Attribute Generator is driven by a database of Attribution Rules. The Attribution Rules include both allowable ranges and the probabilistic distribution of values within those ranges. The Attribute Generator selects individual values at random, based on the probabilistic distribution defined in the Attribution Rules. The result is similar to rolling a loaded die. The attribution rules are user-configurable, allowing the user to modify the Attribute Generator's behavior without needing to modify software. While the Attribute Generator does include a complete set of Attribution Rules, it is always wise to review and modify the rules whenever buildings are generated for a new major geographic region (e.g. Western Europe vs. Southeast Asia).

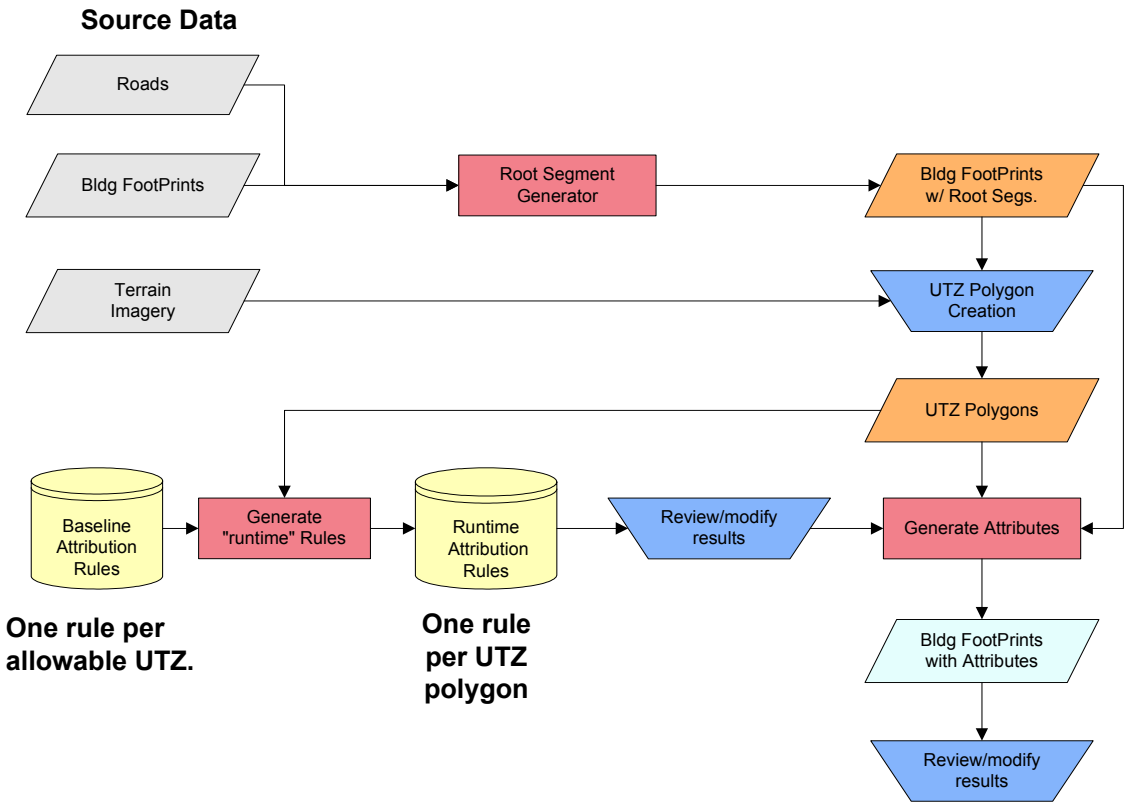

Figure 4: Attribute Generation Process

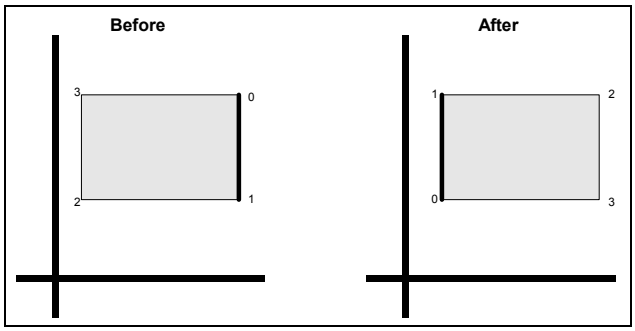

Figure 5: Root Segment Generation

2.3.2.1 Urban Terrain Zone (UTZ)

The most crucial building attribute is its Urban Terrain Zone (UTZ) [4] membership. The Urban Terrain Zone is a classification scheme for identifying groupings of buildings within a city based on building function, density and construction type. Examples of UTZs include close-set office buildings, attached houses, and open industrial regions.

All other attribution rules are based on a building's UTZ membership. For example, building height ranges and distributions are different between a residential UTZ and an office high-rise UTZ.

UTZs are assigned to buildings based on two factors:

- The UTZ polygon containing the building footprint.
- The size (area) of the building footprint.

Area plays a role because 100 square-meter footprints in an office high rise UTZ should have a different height distribution compared to 10,000 square-meter footprints. In fact, 100 square-meter footprints shouldn't be classified as office high rises at all. Therefore, the Attribute Generator detects footprints that are "too big" or "too small" for their enclosing UTZ and reassigns them. The reassignment involves finding another UTZ polygon whose constituent footprint area is a better match, and assigning the UTZ zone from the new polygon. For example, the 100 square-meter footprints in the office high-rise zone may match a zone for small commercial/retail buildings. The 100 square-meter footprints' UTZ attribute would then be set to "commercial/retail" instead of "office high rise".

2.3.2.2 Footprint Area Classification, "Core Classes"

UTZ (re)assignment is controlled by the Attribution Rules, using the concept of "Core Classes." During attribute generation time, the Attribute Generator collects foot-

print area statistics for each UTZ polygon. These statistics include a natural breaks (Jenks [5]) classification of the footprints' area into five classes. Each footprint is then assigned a class number. The Attribution Rules include a Core Values table, which specifies which of the five classes are "Core" classes for each UTZ. Once the classes have been assigned, the Attribute Generator checks whether a given footprint's class matches one of the "core" classes of its enclosing UTZ. If the footprint is in a "core" class, it inherits the UTZ zone value from its enclosing UTZ polygon, if it does not, it inherits its UTZ value from another UTZ polygon that satisfies two criteria:

1. Average area for core footprints of the UTZ polygon is a close fit to the area of the footprint in question.
2. New UTZ polygon is "semantically adjacent" to the original UTZ polygon.

Semantically adjacent means the new UTZ is typically found next-door to the original UTZ. For example, single-family house zones are semantically adjacent to retail areas, but not to office high rises.

2.3.2.3 Floor Count

Floor Count is a "range-based" attribute. Floor Count Attribution Rules include a minimum, maximum, mean and standard distribution for each UTZ. These parameters are used by the Attribute Generator to create a normal distribution of the probability that each floor count within the min/max range may occur. A "loaded die" is then configured and rolled to select the assigned floor count for each footprint in a given UTZ polygon.

2.3.2.4 Floor Height, Building Height

Floor Height (distance from floor to ceiling of one story, is inferred from the UTZ. Typical values are 3 to 4 meters. Building height is simply calculated from Floor Height and Floor Count.

2.3.2.5 Building Function

Building Function is a "discrete" attribute. Unlike floor count, the allowable building functions (house, office, store) can not be assembled into a range with a minimum and maximum. Instead, the Attribution Rules include a table of all allowable Building Functions for each UTZ. An explicit probability is defined for each building function within a given UTZ. Similar to Floor Count, the Attribute Generator uses these probabilities to configure a "loaded die" which is then "rolled" to determine an explicit Building Function for each footprint.

2.3.3 Generating Attribution

Once the Attribution Rules have been configured, Attribute Generation can proceed with minimal user intervention. Attribute Generation proceeds in several stages, with the user able to review/modify the output from each stage, or to run all stages at once.

The Attribute Generation stages include:

- Classify Building Footprints.
  Set up empty tables and fields for the Jenks classifications.
  Gather footprint area statistics.
  Assign Jenks classifications.
- (Optional) User can review the class assignments, and modify which classes are defined as "core" for each UTZ.
- Generate Run-Time Attribution Rules.
  Using the "baseline" per-UTZ Attribution Rules, generate set of "run-time" rules, consisting of an explicit rule for each individual UTZ polygon.
- (Optional) User can modify Attribution Rules for each UTZ polygon.
- Derive Building Attributes.
  Using the run-time Attribution Rules, derive and assign attributes to the building footprints.

2.3.4 Verification

In general, the Attribute Generator is highly effective in assigning reasonable attribution to large data sets covering a wide area. Due to the probabilistic nature of the attribute generation process, however, it is always wise to review the final output to make sure it "looks right". The final output of the Attribute Generation Process is a set of building footprints in ShapeFile format. These footprints can be viewed and edited by standard GIS tools. Items to look for include:

- Aesthetic Distribution of Height
  Height is the first cue anyone uses to identify buildings and cityscapes. Use a 3D viewer to extrude the building footprints by height. Office towers should be tall, warehouses and shopping malls should be shorter, single-family houses shouldn't exceed 2-3 stories. The visual "clustering" of buildings with similar heights should reveal office, industrial, residential districts.

- Distribution of Building Function
  Most urban terrain zones will have a few dominant functions (houses, apartments) with a "sprinkling" of supporting functions (stores, gas stations, libraries).

- Special Cases
  The Attribute Generator generates geotypical attribution. For geospecific cases, such as landmark buildings or specialized facilities, building height and function can be set manually.

2.4 Building Interior Generation

The literature is replete with papers from the architectural community on the automated generation of building interior layouts [6], [7], [8]. However, most of these techniques rely on nonlinear optimization, which is not scalable to the quantities of buildings we envision. Thus we have developed a more *ad hoc* approach, leveraging the native geospatial capabilities of a commercial GIS (ESRI's ArcGIS).

Building interiors, rooms, hallways, and doors are created in a programmatic process based on the spatial and attribution properties of the building. The building footprint is oriented with the first segment of the building geometry in the front of the building. From this point; a five part process begins to create the building interiors within an ESRI ArcGIS Geodatabase. The Geodatabase contains six topologically integrated feature classes: (1) Hallway Lines, (2) Shaft Areas, (3) Hallway Areas (4) Room Areas, (5) Door Points, and (6) Door Lines. After the building interiors have been created within the Geodatabase, they are exported into ESRI ShapeFiles that contains walls and rooms. Figure 6 shows a sample layout.

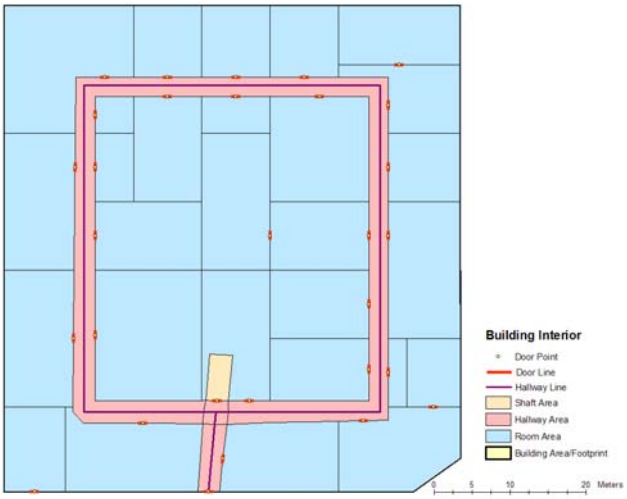

Figure 6: Building Interior features

2.4.1 Hallway Lines

The processing begins by creating the hallway centerline. Hallway centerlines are not created for smaller buildings or buildings attributed to having a single room. In all other buildings, the hallway centerlines are based on the size, shape of the building, and the size of the rooms. Large buildings will have multiple hallways, while smaller buildings will have a single hallway (larger building and smaller rooms will create more hallways, and vice-versa, smaller building and larger rooms will create fewer hallways).

2.4.2 Shaft Areas

Shafts are rectangular rooms with a fixed length and width, with no ceilings or floors, and are used to contain a stairway or an elevator. Shafts are created in all buildings with two or more floors. If the building has a hallway; the shaft is always connected to the hallway.

2.4.3 Hallway Areas

The hallway centerlines are deconflicted against the shaft areas then expanded into area features. Hallways that form a three-way or a four-way connection have a junction area created to split the hallways into single parts. This allows their geometry to be treated like a room in downstream processing.

2.4.4 Room Areas

Rooms are created for all buildings above a specified size and that are attributed to contain multiple rooms. Initially, a collection of rectangular rooms based on specified length and width is created, like a mesh, to cover the buildings. The collection of rooms is rotated to match the primary orientation of the building. The rooms that overlap with the hallways or shafts, and areas exterior of the building, are removed. The final processing step is to merge all rooms below a specified area tolerance with the neighbor that shares the longest edge.

2.4.5 Door Points and Lines

All buildings have doors placed on the front of the building. If the building has multiple rooms or shafts, doors are placed with the following logic: (1) A shaft can only have one door in a predetermined location based on the shaft type; (2) A shaft door must be associated with a hallway if a hallway exists; (3) Room doors are first placed with an associated hallway if a hallway exists; and (4) If hallways do not exist, a door is placed on a room edge that is not shared with a shaft.

2.4.6 Export

The feature classes of the Geodatabase are decomposed into a collection of walls (segments) and rooms (a collection of walls) for each building. The walls are broken down into three segment types: (1) Door segment, a polyline part with four vertices, the two interior vertices represents the door and the entire segment represents a wall;

(2) Empty segment, a polyline part with three vertices, represents a hole or opening in a wall; and (3) Wall segment, a polyline part with two vertices represents a wall (Figure 7). The rooms are exported as polyline format where each part of the room points to a wall segment.

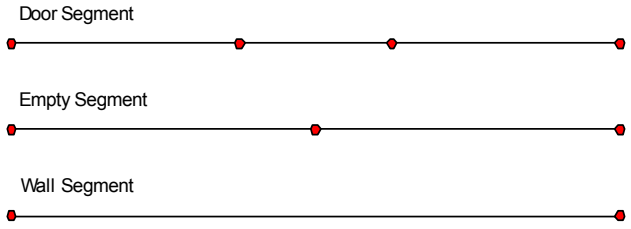

Figure 7: Wall segment types

2.5 Floorplan Generation

The FloorPlan Generator (FPGen) application uses building footprint and attribute data from the attribute generation process and interior floorplan ShapeFiles to produce a high-level abstract representation of the building, including both floor levels and features that connect levels (stairs, shafts). This representation includes basic geometry such as door, window, and wall locations. This basic geometry is later used as source for generating renderable 3-D polygons. This same data is also used as readily traversable structures for constructive simulation entity modeling in UO operations.

All building components and associated attributes created in FPGen, e.g., rooms and walls, conform to the definition and structure of UHRB EDM components. Attribute values for building components are assigned in one of several ways in the FloorPlan Generator:

a) directly from the ShapeFile source (e.g., number of floor levels),
b) default value from UHRB EDM (e.g., number of doors),
c) derived (e.g., number of rooms, or window width), or
d) some combination of the above.

Many constraint checks are performed to ensure that default attributes do not result in the creation of anomalous building components. For example, windows may not be created on walls deemed too narrow, or the EDM-recommended distance between adjacent window widths may be adjusted to compensate.

Each high level building component for which 3-D polygonal representation is required (e.g., an exterior wall) is tagged with a unique XML "DEF" attribute. This identifier will be propagated through the X3D creation process as well, and utilized by the real-time UHRB-Sim for damage assessment and generation, as discussed in Section 3.3.2.

2.6 Polygonal Generation

The Polygon Generator (PolyGen) application derives 3-D geometric information from the XML data produced by the FPGen, and generates an output file in the X3D format. Geometric data resulting from the Polygon Generator is comprised of 3-D polygonal facets with color or texture map information. These polygons are stored as Indexed Face Set objects according to the X3D specification. Appropriate texture map pattern and scale information is assigned through a class interface for querying the UHRB EDM.

PolyGen's basic algorithm involves traversing the XML building representation floor by floor, constructing walls, apertures, floors, and ceilings. These polygons may already exist in a simplified form, as in the case of walls; be implicit in the XML, as for floors and ceilings; or simply be a part of the extended model not included in the floorplan, such as "skirt" polygons to surround the base of the model. The basic difference between the polygonal data in FPGen output and that in the PolyGen is that cutting and triangulation occur in PolyGen. For example, a wall with a door (or other aperture) would be represented in a FPGen XML file as two overlapping rectangles, with some attribution (see Figure 8). PolyGen's job is to "cut out" the door, generate properly attributed and textured polygons for the door and wall, and to triangulate the result for display (see Figure 9).

In the future, PolyGen will also create its own polygonal structures that are only implicit in the XML representation, such as individual stairs in a stairwell, or large pieces of furniture ("fixtures") in a room.

2.7 File System Organization

On a database like Jakarta, there will be millions of .XML and .X3D files. Operating system limitations aside, these files should be organized in some way to aid human readability and comprehension. We have chosen to organize our files in a quadtree-like file system, based on each GTRS geotile [9]. A given building's floorplan and polygon files are placed in a leaf node based upon the geodetic representation of the building's placement point. We have chosen to use an 8-level quadtree, so the leaf nodes are approximately 500 meters on a side. Numbering the quad nodes 0 to 3, the location of a building's polygonal file might be:

<geotile root>/0/2/1/1/2/3/0/1/bldg5721.x3d

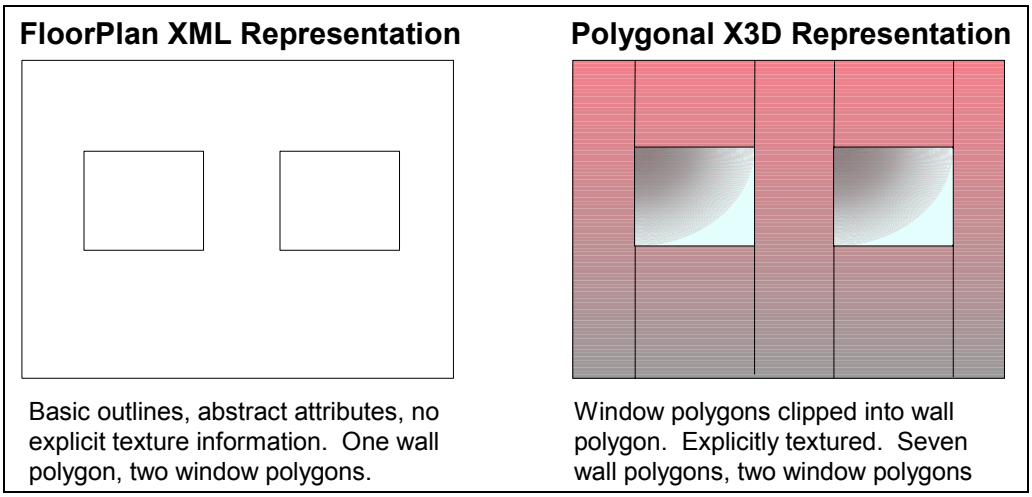

Figure 8: FloorPlan vs. Polygon Representation

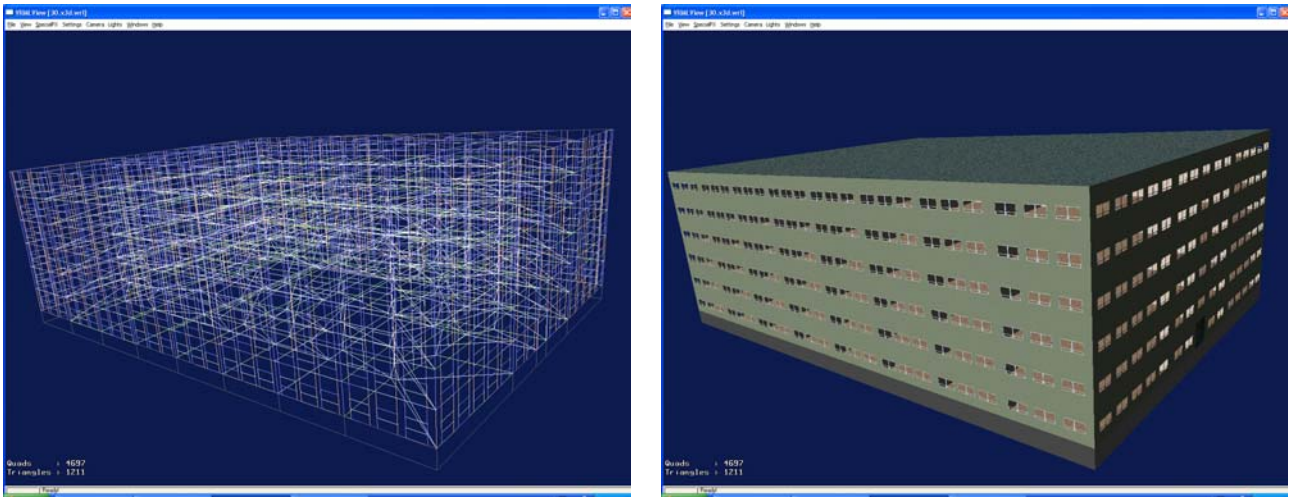
Figure 9: Wireframe and Filled Output of PolyGen

3 Real-Time Modification System: UHRB-Sim

The heart of the real-time modification system is the Ultra-High Resolution Building Simulator, or UHRB-Sim. The UHRB-Sim is a variant of the Dynamic Terrain Simulator (DTSim) platform [10], and uses the same software framework as the RunwaySim, Hydrogeologic Simulator (HydroSim) [11], and Dynamic Terrain Scribe (DTScribe). UHRB-Sim can be executed as a single process with any or all of these other components in the DT architecture. Figure 10 illustrates a building with breach holes in several of its surfaces.


Figure 10: A Damaged UHRB

The purpose of the UHRB-Sim is to listen for building-damaging munition detonations on the communications network, locate potentially affected structures, perform damage assessment, and transmit model changes to other members of the simulation. We now look at each of these operations in more detail.

3.1 Detonation Detection

Munition detonations are expressed in our system as a detonation interaction containing information such as munition type, detonation location, impact vector and speed (if appropriate), and other miscellaneous information that can be of use to some federates (e.g., fuse type). UHRB-Sim registers to receive these detonations from the Federation, and checks them against a list of damaging munitions, pre-generated as a reader file. For the initial implementation, only C4 was listed as a damaging munition. There are other detonations that might interest the UHRB-Sim, from cosmetic damage caused by small arms fire to collateral damage from nearby artillery impacts. These can be easily added later by editing the reader file appropriately.

3.2 Locating Affected Structures

The reader file also contains a blast radius for each damaging munition. Given the detonation location and blast radius, it is a simple matter to locate the structures that *may* be damaged by the blast.

While it would be a straightforward to search the file system described in Section 2.7, it is far more efficient to utilize the MSO database system. This is a small database, distributed with the CTDB database, which includes references to multi-state objects (MSOs), spatially organized by patch. We have incorporated UHRB structures into the MSO database by adding support for a file reference name and a bounding box, which serves as a first-order test for potential building damage.

The reference name is a combination of the geotile name, quadtree path, and file name. An example based on the building in Section 2.7 might be "gh0203_02112301_bldg5721". Therefore, given that the building's bounding box is within the detonation blast radius, we can immediately locate the building floorplan file for damage assessment.

3.3 Damage Assessment

Our damage assessment must be performed in two stages: first for adding apertures to the floorplan XML file, and second to physically damage building (X3D) polygons. We use a concept of a "damage sphere" to represent detonation damage to the building. While we recognize that this is not realistic for all detonations and materials, it allows fast damage assessment with reasonable results for small detonations such as C4. We address methods for improving damage assessment in Section 4.

The basic implementation of the damage sphere is to create a jagged circle (a 15- point circle with some perturbations), and project it orthogonally to the surface. We also shrink the radius of the circle based on distance from detonation to the plane of the damaged surface to approximate a sphere.

3.3.1 Floorplan Damage Assessment

In damaging the floorplan, we take advantage of the inherent ordering of the surfaces of the floorplan to avoid attempting to damage all surfaces of building. For example, if a floor level is outside of the blast radius, it follows that all the walls and rooms of that floor level will be undamaged by the blast.

Floorplan damage is not clipped. New apertures may extend outside of the surfaces they belong to, and this will not cause difficulties. This will allow us to delay any clipping until the polygonal damage stage, with certain benefits, as we shall see.

The floorplan damage assessment algorithm is summarized as follows:

Generate a 15-point blast outline

For each floor level N:
- **If floor level N does NOT intersect the blast radius in altitude, continue**

- **For each wall of floor level N that orthogonally intersects the blast radius:**
  - **Project the blast outline orthogonally to the wall, save the name of the wall and the projected outline.**

- **For each room of floor level N that intersects the blast radius:**
  - **Project the blast outline to the floor and ceiling of floor level N.**

  - **For each room on floor level N that intersects the blast radius, associate room name with the floor and ceiling-projected blast outline**

Send all information to the polygonal damage code.

3.3.2 Polygonal Damage Assessment

Polygonal damage is actually quite straightforward. Because of naming conventions used during the XML/X3D file generation, we know that any given floorplan structure is represented by one or more polygonal sets containing a unique, derivable name. For example, the floorplan structure "F0E0_ExteriorWall" is represented in the polygon file as polygon sets named "F0E0_ExteriorWallOutside",

"F0E0_ExteriorWallInside", and "F0E0A0_Door". By searching for a unique wall "F0E0…", we can locate all polygons which represent the abstract floorplan wall, including any apertures (windows, doors, etc).

Polygonal damage simply involves gathering the polygons representing damaged floorplan structures and clipping them against their associated projected blast outline. While clipping is not particularly quick (we use an implementation of the Weiler clipper), there are no additional calculations that need to be performed to determine which polygons to damage. Therefore, the overall operation of polygonal damage is of minimal execution time. The method becomes even more effective as buildings grow in complexity. The more floorplan structures that are *not* damaged, the larger the percentage of polygons ignored when clipping.

Of course, since all clipping is postponed until this stage, there will be cases in which we discover that no polygon representing a floorplan structure actually falls within the blast outline (see Figure 11). When this occurs, we remove the appropriate floorplan damage from the saved file.

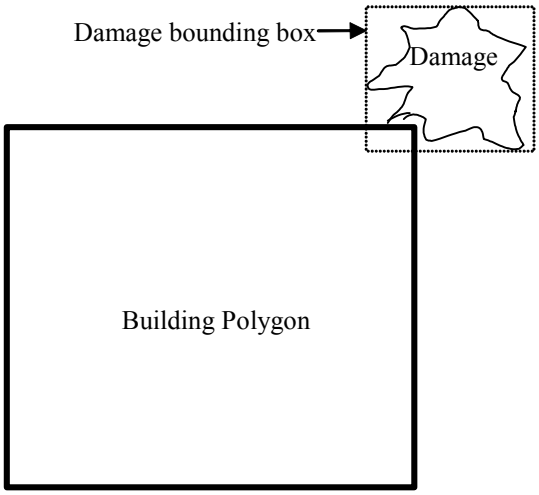

Figure 11: Example in which model of damage does not intersect building

3.4 Model Change Notification

Abstract and polygonal changes are sent as separate interactions in order to minimize traffic to receiving Federates. UHRB polygon change structures have been merged into our terrain polygonal change structures. Although they go out as separate interactions, this allows us to reuse existing code and structures.

Like any other DT interaction, these changes are transmitted to the DT Scribe, which validates and records the interactions for latecomers, and then retransmits them to the interested federates. Among these federates is the UHRB-Sim itself, which will receive the validated changes and utilize a specialized XML API to incorporate the changes into the appropriate (temporary copies of the) floorplan and polygonal files. This action facilitates changes-in-changes. The next time the same building is modified, the changed building files will be accessed. A reset is simply implemented by removing the temporary file, thereby restoring the original files for access.

3.5 Extensions to the OOS UHRB Environmental Data Model (EDM) to adequately describe damage

In October, 2003, a two-day meeting was held at the RDECOM Simulation Technology Center for requirements discovery to extend the OOS EDM [3] to better support building damage. Present were representatives from OOS, COMBAT^{XXI}, and the ERDC Waterways Experiment Station (WES) (with expertise in physics-based modeling of damage to buildings). In preparation for the EDM review, a set of potential requirements was compiled from various documents pertaining to urban operations. Army doctrine was studied to determine the factors in an urban environment that are significant to urban operations. Some of the topics addressed in FM 3-06, Urban Operations, June 2003, FM 3-06.11, Combined Arms Operations in Urban Terrain (formerly FM 90-10-1), February 2002 and Joint Publication 3-06, Doctrine for Joint Urban Operations, September 2002 are shown in Table 1.

Table 1: Urban Operation Requirements from Army Doctrine

Construction Materials and Methods	Engineer Tasks in UO
Rubble	Fragmentation Effects
Toxic Industrial Materials (TIM)	Building Entry
Weapons Effects on Shantytowns	Intentional Rubbling
Infrastructure	Defending a Building
Mobility	Fire Prevention
Removal of Features by Defending Forces	Defending Rooftops
Burned Out Modern Buildings	Small Arms and Machine Guns for Breaching
Door Breaching	Trapped and Injured Survivors

Also used as a resource for the requirements analysis were the inputs and outputs to physics based models for structural damage. The models considered were:

- BLASTX 4.2, March 2001
- A Simplified Analytical Model of Penetration with Lateral Loading (SAMPLL)
- ConWep Penetration
- PENCURV
- CATS-JACE

A sample of the parameters used in these models are given in Table 2.

Table 2: Physics Based Model Inputs and Outputs

Ambient Pressure
Ambient Temperature
Concrete Compressive Strength (MPa)
Concrete Thickness (cm)
Global thermal conductivity of walls in all rooms
Layer thickness
Material type
Number of Openings
Number of Rooms
Opening geometry
Oxygen Fraction
Room type: general room, L-shaped room, tunnel room
S number
Soil Type
Target locations
Wall geometry
Wood Density
Wood Hardness (kg)
Wood Thickness

These requirements, along with the attributes to support structural weapons effect for the WES Structural Weapons Effect (SWE) API, a Java-based application programming interface to provide structural weapons effects functionality, were then compared against the OOS EDMs for Terrain, Atmosphere, Ocean and Space (AOS) and the Ultra-High Resolution Buildings (UHRB). These results were reviewed to determine which of these requirements were important, if they were satisfied by the OOS EDM, and what revisions were needed to support them. The changes resulting from the meeting are given in Table 3.

Table 3: OOS EDM Modifications

Requirement	OOS EDM modification
Flooding buildings or subsurface areas	Added MAXIMUM_STANDING_WATER_DEPTH to FLOOR_LEVEL
Unstable structure from partial damage	Added new attribute STRUCTURE_STABLE to BUILDING
Toxic Industrial Materials (TIM)	Added new attributes HAZARDOUS_GAS and EXPLOSIVE_GAS to room features. Also added TEAR_GAS, NATURAL_GAS, GASOLINE_VAPOR, LIQUID_PETROLEUM_GAS enumerations to HAZARDOUS_GAS_TYPE and applied to COMPARTMENT_ROOM_INTERIOR features.
Rubble	Added RUBBLE 2DGRID. Added MEAN_OBJECT_DIAMETER.
Weapons Effects on Shantytowns	Added SHANTY_TOWN enumeration to URBAN_TERRAIN_ZONE_TYPE of BUILT_UP_REGION
Defending a Building	Added SANDBAG as an enumeration to PRIMARY_MATERIAL_TYPE and SURFACE_MATERIAL_TYPE of WALL.
ConWep Penetration	No change necessary. THICKNESS is applied to ROOF_ASSEMBLY, FLOOR, and EXTERIOR_WALL
ConWep Cratering	Added THICKNESS and STEEL_CONCRETE_REINFORCEMENT_RATIO to ROAD and RUNWAY
Obscurants	Added EXTINCTION_COEFFICIENT_LOSS_QD_BY_EM_BAND and new attribute VISIBLITY_OK to COMPARTMENT_ROOM_INTERIOR features (ROOM, HALLWAY, CLOSET, CATWALK, BALCONY, ATRIUM, ANTE_ROOM)
Reinforced concrete/masonry-	Added PRESTRESSED_CONCRETE as an enumeration to wall construction types.

4 Future Directions

Our implementation of the ABGS and UHRB-Sim are an acceptable first version, but as with all initial implementations, some assumptions and limitations apply. In this section, we discuss some future research that could be directed at this system to resolve some of these limitations.

4.1 More complex buildings

After real-time damage functions, the most difficult part of UHRB is creating realistic, complex buildings in a rea-

sonable amount of time. While there are tools available to manually create good interiors, or to import existing CAD data, these tools operate on a scale of minutes, if not hours, and usually require a human in the loop. If the database is to contain more than a handful of "complete" buildings, better and more automated techniques must be developed.

Secondly, our EDM needs to be extended. For example, currently all building walls are vertical, all ceilings horizontal, and all rooms on one floor have the same ceiling height. These are certainly reasonable assumptions that cover most situations, but completeness (not to mention simulation of artistic design) requires a somewhat more flexible abstract representation.

4.2 Level of Detail

One issue we are already working on for visual and SAF systems is the ability to add levels of detail to our buildings. LODs will be the critical issue for visualization as building polygonal count skyrockets with complexity. Our system will already support LODs, damaging the first LOD only. We are not yet certain if there is a need to change lower levels of detail and, if so, how to best represent the damage.

4.3 Structures Other Than Buildings

Our initial implementation has focused on buildings, but the technique could be applied to other structures which are typically considered difficult to physically model, including tunnels, multi-level bridges, subways, sewers, sky bridges, and so forth. Each of these structures certainly has their own modeling challenges, but this technique provides a generally applicable means of achieving this modeling.

4.4 Incremental database creation

Even if a complex building could be fully generated in one second – and certainly, additional building complexity could be added to any level of sophistication – it would still require 22 computing-days to create the Jakarta terrain database. This turn around time is inconvenient at best, particularly when database errors are discovered at the end of that time.

To solve this problem, we are developing the ability to incrementally build databases, so that individual buildings can be substituted easily. For the initial database, buildings may be partially (or completely) absent, or they may simply be extruded shells with the correct footprint. These will serve as placeholders in the database until the buildings of interest are created and verified.

Further, a distributable building integration tool might be developed, which would allow distribution of the initial database for testing purposes (scenario development, terrain verification, etc.). Later, the building files could be distributed, and each site incorporate them using their local copy of the building integration tool. In this way, databases of arbitrary complexity might be designed and distributed in the most efficient way to the customer.

4.5 More robust damage simulation

Our damage model is currently primitive, based upon the Army field manual description that 10 lbs of C4 at chest height will create a hole "large enough for a man to fit through" [12]. We feel that the damage model in our system as it currently exists fits this description to an acceptable level. However, we do not account for building construction material when damaging a building, nor that a near surface might absorb much of the damage that would otherwise reach a farther surface. Furthermore, enlarging the damage sphere to simulate more powerful detonations can result in unrealistic appearing building damage.

Given some access to subject matter experts, it would be simple enough to extend the munition reader file to include damage to various building materials. More difficult would be a quick approximation of a physics-based damage model. However, we are working on these issues.

4.5.1 Rubble

Our current implementation of damage simply vaporizes the damaged portions of building; expedient, and reasonable for building "mouse holes", but not appropriate for damage on a larger scale. We are currently studying how best to support rubble and debris from building damage.

4.5.2 External damage assessment plugins

Ideally, damage would be handled by a true physics-based model, either as a callable library or as a separate process that could be accessed by the UHRB-Sim. It would also be necessary to find a model that could operate in real-time, and interface in some way with our floorplan polygons.

4.6 More variable attribution

The Attribute Generator currently assigns building height, floor count, and function. More attributes can be supported by updating the Attribution Rules database. Valuable attributes to support include:

- **Roof Style:** Can be used to determine shaped roofs (pitched, curved, saw tooth, …)

- **Building Construction Type:** Can be used to help infer placement interior features such as rooms, hallways.
- **Exterior Surface Pattern:** Can be used to derive visual textures.
- **Color:** Can be used to derive visual textures.

As more attributes are supported, an "inference tree" will need to be developed to identify "primary" attributes which are derived based on UTZ, vs. "secondary attributes" which are derived from primary attributes, and so on.

4.7 More optimized storage

Our current implementation produces one XML and one X3D data file per building in a spatially-organized folder tree on disk. In the future, we anticipate shifting this storage task to a relational database, where the XML and X3D data would be stored as binary large objects (BLOBs). Doing so will simplify configuration control during the ABGS process, and should allow faster access to individual buildings than regular file I/O.

5 Summary

We have demonstrated how our Automated Building Generation Systems and our real-time simulation system, UHRB-Sim, can be used to effectively support UO operations in dense urban environments. Our database generation system can create buildings in under a second, given abstract ShapeFiles gathered from authoritative or synthetic sources. UHRB-Sim, like the DTSim from which it is descended, is a scaleable solution for real-time building damage, easily extended to incorporate external, physics-based damage models.

Much work remains to be done, including ever more complex models, incremental database releases, integration of damage models, and extensions into other structures that can be modeled with this technology. However, we believe that this system displays many of the qualities one would wish in a fully functional system, and presents a strong framework for future development.

6 Acknowledgments

The capabilities described in the paper have been funded by several sponsors. The ABGS development was funded using internal Lockheed Martin funds. Currently, RDECOM Simulation Technology Center is funding additional plug-ins to the ABGS. The UHRB-Sim is being funded by the AMSO MOUT Focus Area Collaborative Team (FACT). The Jakarta terrain database was funded by the Joint Experimentation Directorate (J9) at USJFCOM. We thank each of these sponsors for their investments.

7 References

[1] S Crino: "Representation of Urban Operations in Military Models and Simulations", Proceedings of the 2002 Fall Simulation and Interoperability Workshop, paper 01F-SIW-021, September 2001.

[2] Web3D consortium: "Information technology — Computer graphics and image processing — Extensible 3D (X3D)", ISO/IEC 19775:200x, available online at http://www.web3d.org/specifications/ISO-IEC-19775/index.html.

[3] D. Miller, et al.: "An Environmental Data Model for the OneSAF Objective System", Proceedings of the 2002 Fall Simulation and Interoperability Workshop, paper 02F-SIW-082, September 2002.

[4] Liu, J. and Ellefsen, Richard. Small business innovative research, Phase II: Final scientific and technical report, Volume 1 and volume 2: UTZ-based urban terrain feature database. Sunnyvale, CA: TERA Research Incorporated, 1996.

[5] Jenks, G. Optimal Data Classification for Choropleth Maps. Occasional Paper 2, Department of Geography, University of Kansas, 1977.

[6] J. J. Michalek et al., Architectural Layout Design Optimization, Eng. Opt., 2002, Vol. 34(5), pp. 461–484.

[7] J. J. Michalek et al., . Interactive design optimization of architectural layouts. Eng. Opt., 2002, Vol. 34(5).

[8] B. Medjdoub and B. Yannou, Separating topology and geometry in space planning. Computer-Aided Design, Vol. 32, pp. 39–61, 1999.

[9] P.Birkel, et al.: "Pushing the Envelope Toward a Common Process for the Generation of Terrain Data Bases across Federations", Proceedings of the 1999 Spring Simulation and Interoperability Workshop, paper 99S-SIW-016, March 1999.

[10] D. Miller, et al.: "Dynamic Terrain in the Environment Federation", Proceedings of the 2000 Spring Simulation and Interoperability Workshop, paper 00S-SIW-015, March 2000.

[11] S. Adelson: "A Scaleable Solution for the Hydrogeologic Simulator - Environment Federation", Proceedings of the 2002 Spring Simulation and Interoperability Workshop, paper 02S-SIW-017, March 2002.

[12] Army Field Manual FM 90-10: "Military Operations on Urbanized Terrain (MOUT)", 15 August 1979.

Author Biographies

DR. STEPHEN J. ADELSON is a senior software engineer at Lockheed Martin Simulation, Training and Support Advanced Simulation Center in Bellevue, WA. He contributed to all aspects of the DARPA and USATEC projects *Dynamic Terrain* and *Objects in a Virtual World*, with a special interest in the repoly service, which allows

dynamic terrain modifications of all sizes (a result of engineering or munition damage) and database tailoring aimed towards specific training goals. He was an integral developer in the first phase of the DMSO-sponsored *Dynamic Terrain in the Environment Federation* project, and continued as a primary developer of the latest incarnation of the Hydrogeologic Simulator federate in the FY 2000 and 2001 follow-on projects. Dr. Adelson received his Ph.D. in computer science from the Georgia Institute of Technology in 1993, with a specialty in efficient graphics algorithms for VR, simulation, and animation. Before joining LM STS-ASC in 1996, he served as a postdoctoral research associate at Los Alamos National Laboratory, researching advanced computer simulation algorithms and web-based tools for collaboration.

STEVE FARSAI was a senior software engineer at Lockheed Martin Simulation, Training and Support Advanced Simulation Center in Bellevue, WA.. Mr. Farsai received his B.S. degree in Computer Science from the University of Washington in 1986 and has been employed at Lockheed Martin for 15 years. He was one of the key developers of the S1000 terrain database generation tools used in the construction of Synthetic Environments (SE) for Modeling and Simulation (M&S). He led the development of the S1000 platform-independent API used in applications generating specialized native synthetic environments for CGF systems such as ModSAF, JSAF, OneSAF, and CCTT. Mr. Farsai has developed a user-interactive toolkit for forward engineering and reverse engineering environmental data models represented in relational database form enabling representation in alternative entity-relationship form. Mr. Farsai was an original member of the 6-person industry-wide selected team to conceive and design the Synthetic Environment Data Representation and Interchange Specification (SEDRIS). Most recently, Mr. Farsai was the lead engineer on the Floorplan Generator reported on herein.

LEO SALEMANN is a senior staff software engineer with Lockheed Martin Simulation, Training and Support Advanced Simulation Center in Bellevue, WA. He received his Bachelor's of Science in Computer Science & Engineering from the University of Washington in 1993 and has been working for the LM STS Bellevue office ever since. Leo is an expert in object-oriented development in Visual Basic as well as UNIX/C environments. Leo contributed to the WARSIM/TDFS program from 1998 to 2003, and has participated in the design, implementation, documentation, and testing phases. His focus has been in the User Interface and Application layers, in which he was primary author of numerous mission critical software components. Leo's previous work includes training, documentation, release generation and customer support for the Vistaworks real-time visualization software, and the GT200 image generator.

DR. DALE D. MILLER is the manager of Advanced Technology Development for the Advanced Simulation Center group of Lockheed Martin Information Systems. He led the projects *Dynamic Terrain and Objects in a Virtual World* and the *Terrain Database Generation & Technology for Synthetic Environments* both sponsored by DARPA and USATEC. He was also the technical leader for the terrain database and terrain data fusion development for the WARSIM Program. He was the Principal Investigator for the OOS EDM development and is currently leading environmental data modeling activities for the AMSO / TEC Environmental Database IPT. Dr. Miller received his Ph.D. in Mathematics from the University of Washington in 1976.

TIMOTHY MILLER is a geologist/geomorphologist and GIS applications developer with Lockheed Martin Simulation, Training and Support Advanced Simulation Center. He has 15 years of experience in all aspects of GIS, application and data development, relational database development and management, and web-based application development. He led the GIS activities for the STOW Terrain Scenario Generation and Archiving (TSGA), the JSIMS Terrain Data Fusion System (TDFS) programs, and the Geospatial-Intelligence Database Integration (GIDI) project. He also has been involved in all aspects of simulation Terrain Databases (TDB) generation, including software and data generation development. In TDB construction, he has been involved with the synthesis of available digital data sources and developing new data sets for accurate representation of real world phenomena.

MELISSA NAKANISHI is an environmental data modeler with Lockheed Martin Simulation, Training and Support Advanced Simulation Center. She had lead responsibility for developing and maintaining the OOS EDM-T, and has developed EDMs for several legacy M&S and C4I systems, including Janus, CASTFOREM, BBS, and the Joint Common Data Model (JCDM).

JULIO DE LA CRUZ is the lead for synthetic natural environment and simulation technologies at the Simulation Technology and Training Center, Research Development and Engineering Command (RDECOM-STTC). Mr. de la Cruz has an undergraduate degree in Electrical Engineering from the City College of New York and a Master degree of science from Texas A&M University. Mr. de la Cruz has over fifteen years experience in government and private sector. Mr. DeLaCruz oversees research and science and technology objectives focused in the development of current Database Generation Systems/tools technology to address the defined Army need in Modeling and Simulation.