AZ-SMART Architecture Draft

Center for Urban Simulation and Policy Analysis
University of Washington

February 28, 2007

1 Proposed Architecture for AZ-SMART

This document develops an architecture plan for the AZ-SMART project. It is intended to provide an overview of the architecture from multiple perspectives: system, software, user interface, data and models. Design choices are generally provided as recommendations, but in some cases two or more alternatives are presented for further discussion. The intent is to update this document as design choices are finalized so that it can serve as an ongoing documentation effort.

1.1 Functional Architecture Perspective

The functional overview of the AZ-SMART system is depicted in Figure 1, showing multiple activities connected in a sequential but also bi-directional workflow. One can begin with the top-right of the diagram and work counter-clockwise to understand the intended workflow pattern. An AZ-SMART user will begin with the GIS tasks of compiling a database in the Geodatabase that will contain all the needed information for the AZ-SMART system, with the exception of the travel model and a mid-level model that will be interfaced to AZ-SMART (in a later phase, the mid-level model is likely to become a component of AZ-SMART rather than an external process).

As the database is assembled in the gepdatabase, including all land use layers, environmental, planning and political boundaries, development projects, and other related inputs, the AZ-SMART user will employ a suite of diagnostic tests to examine the database for missing, erroneous, and inconsistent data, and will use editing and validation tools to repair problems, and impute values for missing and erroneous data. During this process, there would be considerable iteration between the data assembly step and the data diagnostic step, as some input data might be replaced or updated. This step would also involve the assembly of metadata concerning the data sources, vintage, and any information developed on data quality.

The next step in the workflow is the development of models that are consistent with the database and can be used to allocate land uses to small areas (we discuss the form of the data in detail in the data subsection). The models used by AZ-SMART will be configured to undertake land use allocation as described in the model subsection, and will be flexible with respect to the variables and weights (parameters) used. The models themselves will be modular and flexible to allow ongoing refinement over time by AZ-SMART users as they gain insights into alternative means of doing the allocations. This step also involves selecting the variables to use in each model component, and estimating model parameters statistically, or if desired, inputting parameters manually for some model components. The tools for model estimation will be internal to the system, allowing ease of re-estimation if the user wishes to experiment with alternative specifications. Diagnostics from model estimation, such as sensitivity to variables and correlation among variables, will be available to assess the robustness and sensitivity of each model component. The user will also be able to compare observed to predicted data over an observed period if such data is available. This model specification and calibration process may identify problems in the data, which will require moving back upstream in the workflow process to refine input data.

Once the models are estimated and the system calibrated to the users' satisfaction, the model system may be put into use. Putting the model into use requires the specification of one or more 'scenarios' that contain the input data that define the input conditions and assumptions for a run of the model system, including control totals, land development constraints, development projects, and transportation system. These scenarios may involve graphical editing in the GIS environment, to edit boundaries of designated areas or to generate buffers or distances from specified features such as access points on the transportation system. The scenario specification will also include controls for the interfacing of the travel model system (which years to run, what variables to pass, for example), and the mid-level land use model system or the control totals that have been generated by them.

The next step in the workflow is to run one or more scenarios over a forecast period defined by the scenarios. This may be done in a batch mode that automates the entire modeling process from the loading of initial data for the base year, through multiple time steps and exchanges with the travel model, and possibly the mid-level model, until completion of the entire modeling period of 30 or more years (for example). In other cases, the user may wish to run a scenario only for a portion of the intended forecast period, then stop the simulation to examine results, and possibly to make changes in the scenario assumptions that will shape the simulation in the next section of the forecast period. Yet another possibility is that the user will see patterns in the results that are not desired, and decide to back up to the scenario specification stage, or even to the model specification stage, to make refinements and resume the process.

Typically the process of examining simulation results will involve creating indicators and extracts (or summaries) of the detailed simulation results. We shall refer broadly to these as indicators. The indicators may be simple aggregation of results, such as housing units or acres of each land use sector by zone, or RAZ. Or they may involve computations that are more complex, such as estimating the remaining development capacity within each city. The indicators may be produced as maps, or charts, or tables. They may be of three types: snapshots (looking at an indicator for a single scenario and year), changes (usually from the base year to some forecast year), and differences (usually between a specific scenario and a baseline scenario for a selected year). it is likely that many of these would be map indicators and be viewed in the GIS environment.

Simulation results may also be stored to the Geodatabase. A user may wish to store either the full set of simulation results with all of the intermediate calculations needed to understand or even re-start a simulation from an intermediate year, or alternatively only a subset of specified results can be written to the Geodatabase for archiving and further visual examination. The system will provide tools for both use cases.

This brings the workflow full circle to the top-right, where the data used as inputs and the data produced by the simulation may all be visually examined and edited. Refinement of forecast results would be done at this stage also. There are of course many possible variants on this workflow description, and theoretically one could draw connections among almost any pair of these task components, and the AZ-SMART system will support this kind of flexibility (but the diagram would be quite hard to interpret at that point). The next section moves to the system architecture perspective.

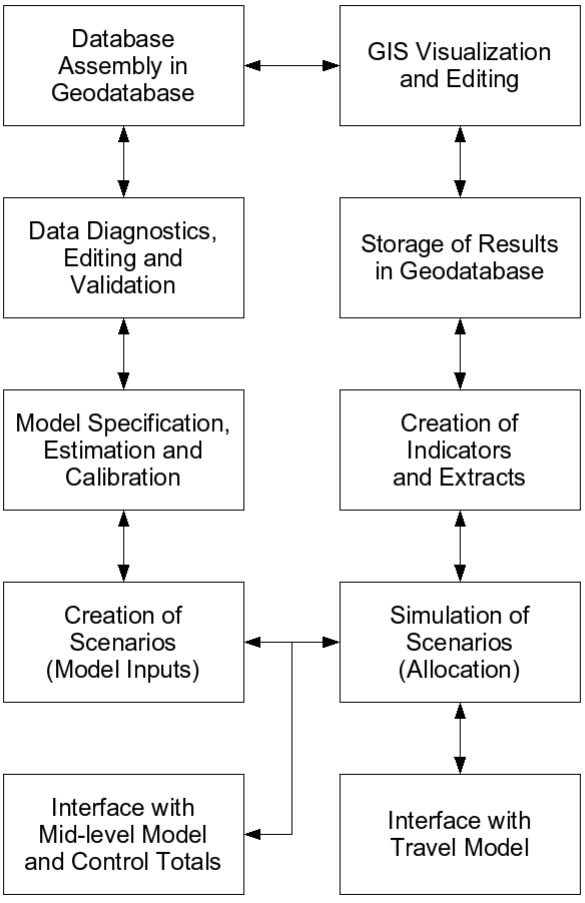

Figure 1: AZ-SMART Workflow Diagram

1.2 System Architecture Perspective

This section provides a system perspective on AZ-SMART. We describe a long-term perspective for the system, and suggest a first phase implementation. The system is anticipated to evolve into a distributed configuration as depicted in Figure 2, in which there are possibly different servers for the Geodatabase, running the land use model, running the travel model, and potentially a web server using ArcIMS or its successor.

Clients wille be ArcGIS workstations and OPUS Clients that would generally be on the same machine, integrated through the ArcGIS user interface (described in more detail in the subsection on user interface). There may be situations in which a user wishes to run an OPUS client independently of ArcGIS, for example when focusing on model estimation, or at times when controlling simulation runs, or in the use of a fully automated batch simulation linking the land use and travel model systems. The interface between OPUS clients and servers would use networking infrastructure such as the Twisted Python framework.

In the first phase, the OPUS client and server will be on the same machine as a an ArcGIS workstation, with multiple installations, one per user. Data will be shared via the Geodatabase, and if desired also through a drive that is mapped from each of the clients to share the cache directory containing simulation results.

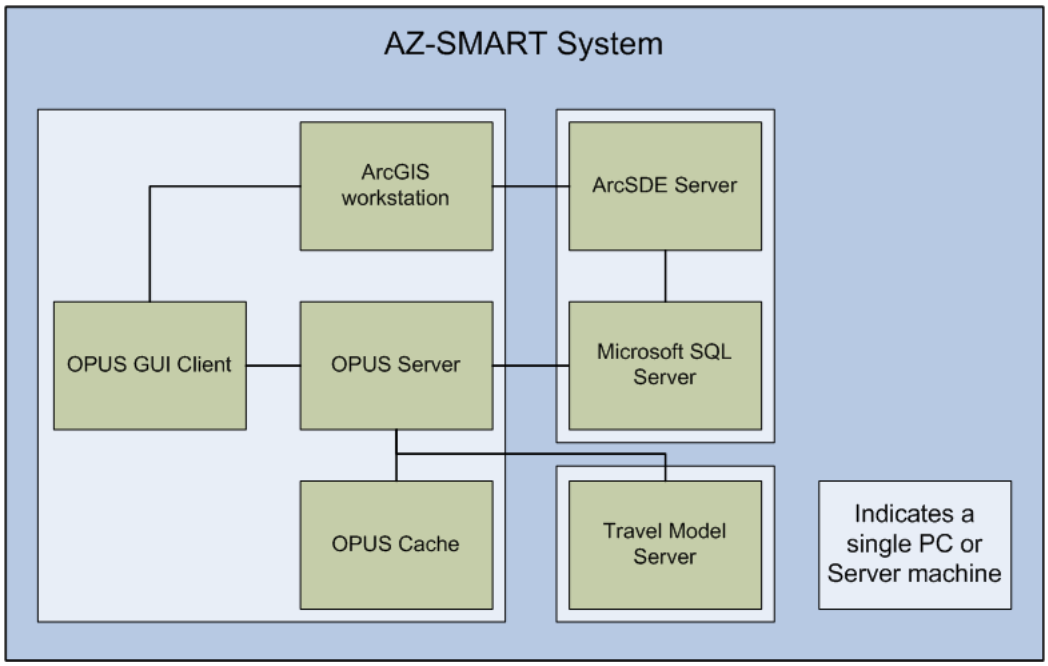

Figure 2: AZ-SMART System Architecture

1.3 Data

1.3.1 Data Models

Insert narrative and E-R diagrams.

1.3.2 Storage

Insert stuff about Geodatabase and OPUS Cache.

1.3.3 Data Diagnostics, Validation, and Imputation

Insert stuff about data quality.

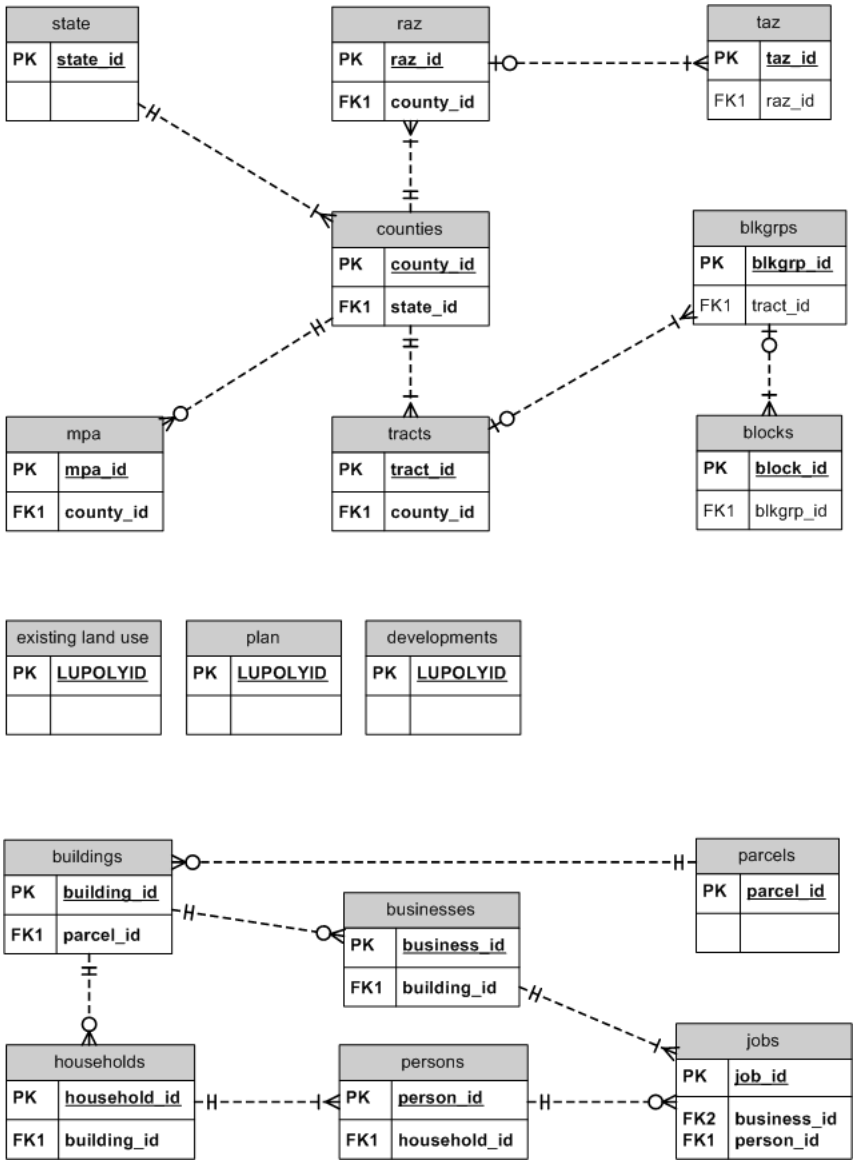

Figure 3: AZ-SMART Data Model

1.4 Software Components

In this section we describe the software implementation of the AZ-SMART system. AZ-SMART will include three modules (Project Manager, Data Manager, and Tool Manager) which are largely based on the requirements in AZ-SMART RFQ Appendix G. Each module's organization and implementation are described in turn in the following subsections.

1.4.1 Project Manager

The Project Manager will be the heart of the modeling system. CUSPA proposes to implement the Project Manager user interface as an OPUS GUI (developed using Envisage) that could be launched from an AZ-SMART Tool within ArcGIS. See below for a screen capture of a very basic Envisage GUI application. The Project Manager will be the central application where simulation runs are configured, managed, started and stopped. Projects are defined as the database and model configuration used to support running a variety of scenarios that would share data and model specifications and parameters. Scenarios are defined as a set of input data, assumptions, and run configuration parameters used for a specific run of the model system.

Several of the items listed in the Project Manager requirements in AZ-SMART RFQ Appendix G are very focused on control of model operations: controlling model execution, accessing the status of a model while executing, and accessing execution logs and error logs associated with a model run. Compared to an approach of using only native ArcGIS GUI tools to code the Project Manager, the approach of coding a native OPUS GUI will make it more feasible to implement these requirements. It would need to be implemented in a way that could inter-operate transparently with other tools in the Tool Manager and Data Manager components, as noted in the first requirement below. Some initial tests of this approach should be developed early in the project to flesh out this aspect of the user interface.

1.4.2 Data Manager

The Data Manager will be the central location for the management of data within AZ-SMART. We propose that the Data Manager be implemented within the ArcGIS framework as a dockable window, coded in VB.NET and ArcObjects, that organizes the necessary data management tools in a 'tree-view' framework. The Data Manager dockable window would be accessible within both ArcMap and ArcCatalog to provide flexibility in use, although in regular production operation it would probably make the most sense to utilize the Data Manager primarily within ArcCatalog. See below for example screen captures of this type of dockable window implementation in both ArcCatalog and ArcMap.

The Data Manager will focus on metadata maintenance and management, management of security levels and users, data archiving, and the movement of data between the OPUS cache and the geodatabase. To the greatest extent possible, existing ArcCatalog functionality will be utilized to manage the geodatabase, including managing and maintaining geographic datasets and creating and tracking relationships.

There is a need to define requirements for metadata, security levels, and data archiving in the AZ-SMART system. Once these requirements are identified we can proceed with devising suitable designs to address these requirements.

1.4.3 Tool Manager

The Tool Manager will be the central location for the management and execution of the wide variety of tools to be developed for AZ-SMART. CUSPA proposes to utilize the ArcToolbox/ModelBuilder framework for the organization, management, indexing, and execution of tools. Specifically, a new AZ-SMART toolbox will be populated with the required tools, which will then be organized into toolsets based on common functionality. To the greatest extent possible new functionality for AZ-SMART will be developed as Python geoprocessing tools within these toolsets, while minimizing custom GUI development within the ArcMap and ArcCatalog applications. If the required functionality cannot be implemented in Python through the existing ESRI geoprocessing object and/or other add-on Python libraries, CUSPA will develop custom geoprocessing tools using VB.NET and ArcObjects.

1.4.4 Other ArcGIS AZ-SMART Components

Note from Jesse: we may need other customizations within ArcGIS to accomplish what they want in App G. For instance, they desribe a land use editor that may require some custom GUI development. Should we attempt to describe those here?

1.5 Models

Insert stuff about the model system.