AZ-SMART Architecture Draft

Center for Urban Simulation and Policy Analysis
University of Washington

March 1, 2007

1 AZ-SMART Architecture Plan

This document develops an initial 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 made so that it can serve as an ongoing documentation effort. It is very unlikely that the final AZ-SMART application will implement this plan exactly. Rather, it provides a point of departure, and a general map for the intended application, which will be adapted continuously as the project participants develop, use, test, and refine components and their interactions and interfaces.

1.1 Functional Overview

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.

1.2 System Overview

This section provides a system perspective of AZ-SMART. The system is a distributed configuration as depicted in Figure 2, in which there are possibly different servers for the Geodatabase (ArcSDE and MS SQL Server), the land use model (OPUS), the travel model, and potentially a web server (using ArcIMS or its successor). Initially we expect that OPUS and ArcGIS will be running strictly on the same machine.

The interface between ArcGIS and OPUS will be 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. Similarly, a user might want to have multiple OPUS Clients on different machines, but share an OPUS Server on another machine, to take advantage of a fast server, for example. The interface between OPUS clients and servers would use networking infrastructure, though there may be no need for this infrastructure during the first phase of this project.

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 (or machine). 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.

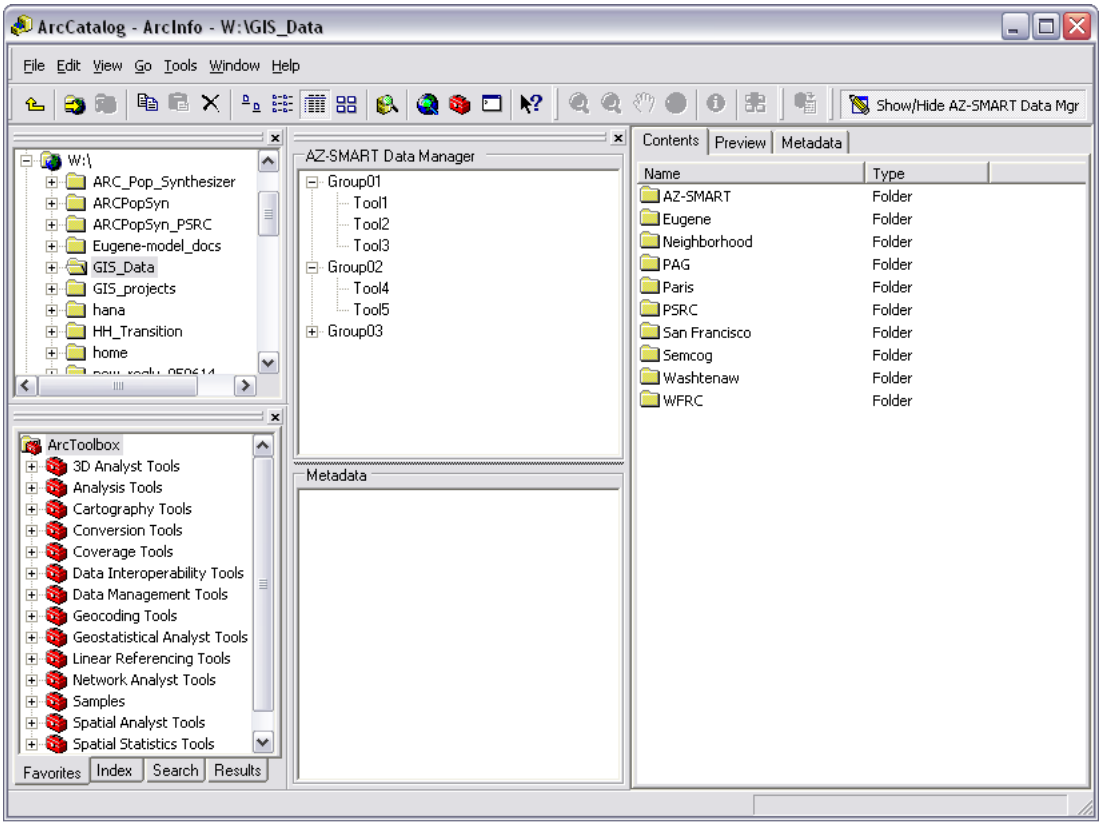
1.3 Software Components

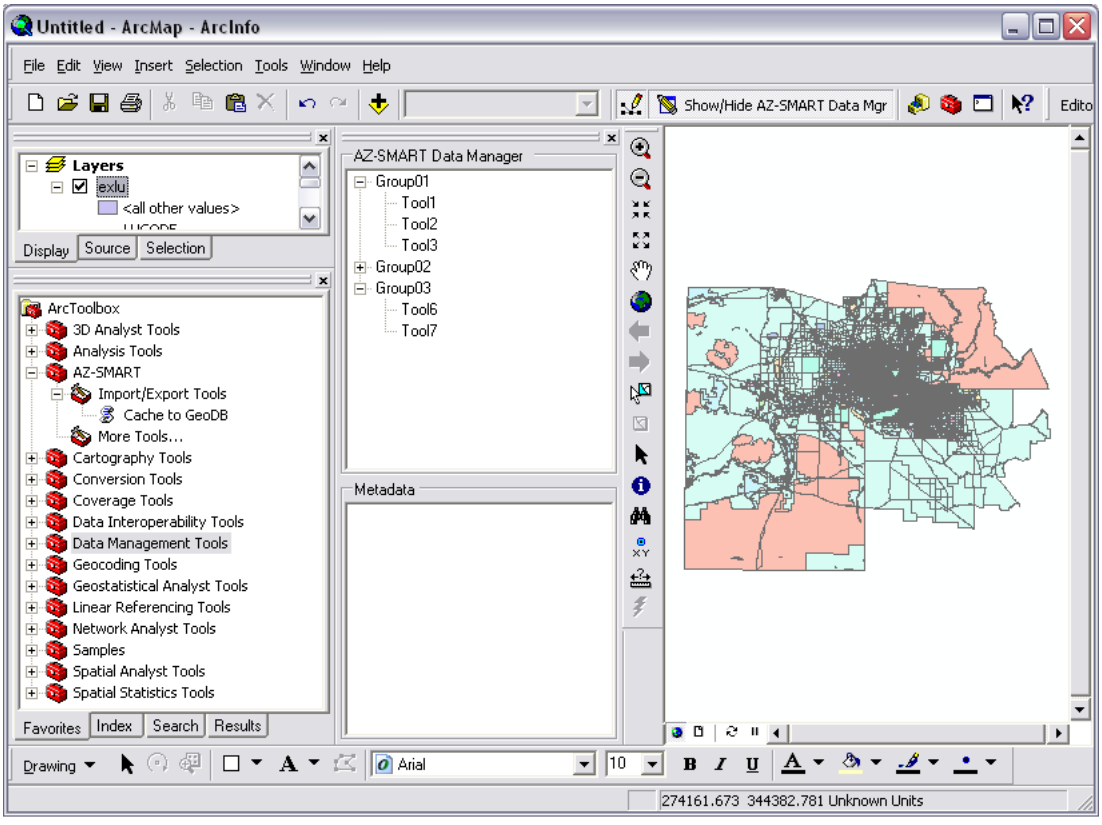
In this section we describe the principal software components and organization of the AZ-SMART system. AZ-SMART will include three modules (Project Manager, Data Manager, and Tool Manager) based on the AZ-SMART RFQ Appendix G. Each module's organization and implementation are described in turn in the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

1.3.1 Data Manager

The Data Manager will focus on the management and visualization of data within AZ-SMART. 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 figures XXX and XXX for example screen captures of this type of dockable window implementation in both ArcCatalog and ArcMap.

.....change the reference in the preceding sentence to the Latex reference....I am having trouble getting the code in the .tex doc to build, so I just inserted the figure as you see here





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, and data archiving in the AZ-SMART system. Once these requirements are identified we can proceed with devising suitable designs to address these requirements.

1.3.2 Tool Manager

The Tool Manager will focus on the management and execution of the wide variety of tools to be developed for AZ-SMART. The ArcToolbox/ModelBuilder framework will be used 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.

Jesse: insert appropriate screenshot and reference here. Can you flesh out an initial list of the tools you think will be needed in this in order to accomplish what we need? I believe we need more detail here and you should be able to do this from the information you have on SAM and your understanding of how functionality could be accessed via ArcTools.

1.3.3 Land Use Editor

When SAM-IM was develped in the ArcView environment, there was considerable functionality not present in the underlying GIS platform that needed to be developed in SAM-IM, such as editing of polygons. In the development of AZ-SMART on the ArcGIS 9.x platform, there is no need to develop such functionality, since editing tools for spatial data are built-in. The initial plan for AZ-SMART is to maximize the use of built-in functionality and minimize the amount of custom-code development which will need to be maintained and synchronized with evolving ArcGIS functionality and interfaces. If it becomes clear that there is some need for customization of the built-in functionality for editing after the AZ-SMART system is put into testing and use, specifications for this could be developed and the task prioritized along with other tasks in the project.

1.3.4 Project Manager

The Project Manager will focus on managing the model components of the system. The Project Manager will leverage and manage modeling infrastructure embeded in the OPUS system, and will take advantage of the run management capabilities of OPUS such as model specification, estimation, execution, and generation of indicators. The Project Manager user interface would be accessible from within ArcGIS, and will open a windowed application which we refer to as the OPUS GUI. The OPUS GUI, developed in Python and using the Envisage tools from Enthought, could also be used outside of ArcGIS, providing flexibility for a range of use cases. The Project Manager will be the central application where simulation runs are configured, managed, started and stopped, and will interoperate seamlessly with the Data Manager and Tol Manager components of AZ-SMART.

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 System Security

There are many situations in which there may be concerns related to restricting access to data or functions to different users and/or machines. We propose to exploit existing infrastructure wherever possible to address

security requirements. Several security points are available using existing system functionality:

- Database Access: Access to the Geodatabase may be protected at the user or group level, and can be applied to databases or possibly to the table level, for managing read and write access to data. This would provide a lightweight mechanism for managing user access, for example by restricting which users or groups have access to a database containing the database containing the core data for the AZ-SMART application. More fine-grained control would also be possibe, for example protecting employment data from read access outside a set of users within the agency that have been cleared for this access.
- Login Access: Each machine that has AZ-SMART installed will have login access managament, which can be used as a simple but effective way to prevent unauthorized access to the AZ-SMART system.
- Machine Access: A machine may be enabled for access by setting environment variables that can store username and password information, and this can be loaded by the az-smart system as needed for security enforcement. A generic az-smart user could be defined, or a group account, or an individual user account and a password assigned to each. These usernames and passwords could be stored in an az-smart-security database that only an AZ-SMART Administrator is enabled to access and maintain.

1.5 Data

1.5.1 Data Models

Insert narrative and E-R diagrams.

1.5.2 Storage

Insert stuff about Geodatabase and OPUS Cache.

1.5.3 Data Diagnostics, Validation, and Imputation

Insert stuff about data quality.

1.6 Models

The model architecture is adapted from the functional design of SAM, and also informed by the development of OPUS and UrbanSim. The functionality in SAM focuses on the allocation of land use by sector to grid cells, from aggregate information at a mid-level geography such as MPAs. By reference to the UrbanSim model system, this functionality is approximately equivalent in purpose to the real estate development model component of UrbanSim, with some key differences. UrbanSim attempts to include a complete representation of the real estate market, with occupants (consumers: households and jobs), buildings and land (suppliers: developers and property owners), and prices (markets: hedonic regressions representing the interaction of suppliers and consumers).

The architecture for the model system proposed for AZ-SMART is based on a 3-year plan, and incorporates the complete market representation as in UrbanSim, and a multi-level geography and model system. We describe this in the Full Model System subsection below, and then focus on a Phase 1 Model System in the subsequent section.

1.6.1 Full Model System

The full model system proposed for AZ-SMART involves some hybridization and extension of elements of UrbanSim and SAM-IM. Below we itemize the core elements of the full model system architecture.

- *Land-Structure-Occupant Accounting*: The full market representation and explicit representation of and accounting of Land-building-occupant objects and their relationship is proposed for the full model architecture.
- *Parcels and Buildings*: Land could be represented by parcels, land use polygons, or cells, but it is expected that the parcel concept would be used as the principal representation of land, and that in areas that do not have parcel data available, land use polygons could be used in a way that treats them

as equivalent to (possibly large) parcels. Note that there are many to one relationships from buildings to land and from occupants to building. That is, a building may contain multiple occupants, and a parcel (or land use polygon) may contain multiple buildings. In the event that building data is not available, it could be imputed, to preserve a consistent data model.

- *Development Projects, Sites and Templates*: Development Projects are proposed as a higher-level construct to represent one or more parcels that form a coherent single development project, such as single-family housing subdivision, or a shopping center complex. These development projects will deal both with known *development projects* which the user wishes to incorporate into the simulation, and also development projects predicted by the simulation and assigned to *development sites*. For predicted development projects, a set of pre-defined *development templates* provide a set of configurations of development that include at a minimum the mix of building types (land uses), density, size and timetable for development.

- *Multi-level Model*: The full model system would use a multi-level approach, incorporating parcels as the lowest level (buildings are linked to parcels), and for the forseeable future, one higher level geography to represent an intermediate geography between the county and the parcel. Traffic Analysis Zones (TAZ) are proposed as the basis for this mid-level geography in the full model, simplifying the interface with the travel model system. There are behavioral and practical reasons for using a two-level geography in the model system. Behaviorally, it is based on the expectation that consumers looking for a location (e.g. a household searching for a house) compare neighborhoods, and select properties to examine based on their assessment of the neighborhoods. In practical terms, the two-level geography provides a more convenient way to interface models in a modular way, for example to interface the travel model system, or to run the model system for corridor or area studies where more detail is needed in a subset of the planning region and less is needed outside of this focus area.

- *Microsimulation*: The proposed architecture is based on explicit representation of the agents and objects being modeled at a microscopic level. Parcels, buildings, businesses (or jobs), households, and eventually, persons (for supporting activity-based travel models and workplace choice models and individual-level accessibility calculations).

- *Temoral Dynamics*: The model system would be able to use a specified time interval, such as 5-year or 1-year steps, between which it would simulate changes to the state of each object and agent in the system (construction of new buildings or conversion of existing ones, movement of households and businesses from one location to another, creation of new households or businesses).

- *Models and Interface*: A set of models will be interfaced through a common data store, and managed by a Model Coordinator that controls their execution and implements events (changes to the data) proposed by models. The individual models would be the following:

  - Demographic Transition (Region)
  - Employment Transition (Region)
  - Development Project Transition (Region)
  - Household Relocation (Region)
  - Business Relocation (Region)
  - Household Location (TAZ and Parcel)
  - Business Location (TAZ and Parcel)
  - Parcel Subdivision (Parcel)
  - Parcel Aggregation (Parcel)
  - Demolition (Building)
  - Development Project Location (Development Site)
  - Building Development Model (Parcel)
  - Real Estate Price Model (Building)
  - Simplified Travel Model (TAZ)

1.6.2 Phase 1 Model System

In the draft below, we develop the architecture for the general allocation model that would apply to each land use sector, with some differences in the configuration of the model for each sector. As an example we discuss the single family residential land use sector. This might not be the preferred sector to test initially in implementation, but provides some useful features for describing the model architecture.

The model architecture has the following steps, using a 5-year time interval:

=== Determine Quantity of Development Needed ===

* Translate mid-level model predictions of population and employment by RAZ (or other mid-level geography) into predicted demands for housing units and land area of non-residential uses by type. ** This would be done using average density and occupancy assumptions.

=== Determine Development from Active Projects ===

* Compute the quantity of development expected in a RAZ (or other mid-level geography) for each land use sector, based on the velocity within Active Development projects.

* Assign the development generated by Active Development Projects to the cells within those developments.

=== Determine Eligible Development Sites ===

* Evaluate cells with vacant land to determine what kinds of development would be permitted on them according to the development constraints represented by Planned Land Use and other constraints (slopes, etc). This applies development constraints in order to produce Eligible Vacant Lands for development.

* Evaluate potential availability of Redevelopment Districts for new development.

* Generate 'Development Sites' from 'patches' of available land suitable for development. These are areas formed by contiguous cells eligible for development for a particular land use sector, and would serve as a counterpart to the polygons digitized by users for planned and active development projects. They would allow the comparable representation of Development Sites from all sources.

=== Assign Development Projects to Development Sites ===

* Use 'Development Sites' as the candidate set of locations for new development within the RAZ.

* Generate a set of proposed 'Development Projects' that would fill the gap in the RAZ between the development that is generated by evaluating the velocity of Active Development Projects and the quantity required to meet the RAZ control totals.

* Compute logit model probability that a Development Project in a RAZ will choose one of the available candidate Development Sites.

** Compute variables used in utility (scoring function) ** Multiply variables by their coefficients ** Compute utility ** Compute probability

* 'Choose' a Development Site for each Development Project to be located based on the logit probabilites. Specifically, draw a random number and compare it to the cumulative probability distribution across the alternative Development Sites. Choose the site that the random number falls within (the choice pattern will be proportional to the logit probabilities). Ensure that the capacities of the sites are respected: if a Development Project is too big for a site, exclude it from consideration.

* Once a Development Site is chosen for a Development Project, assign the Development Project to the Development Site, and set the Development Project status to 'Active', so it becomes part of the set of Active Development Projects that will be used to generate development at the beginning of each period.

* Generate initial development of the newly selected project sites for the current time period based on the predicted or stored development velocity.

* Compare generated development to that required to meet control totals. Add more development if necessary.

=== Iterate over RAZs (or other geography) and Allocate Unplaced Development Projects ===

* Once allocation of all development in a RAZ or other mid-level geography is completed, process the next RAZ.

* If not all development could be allocated in a RAZ, accumulate unmet demand within a higher level geography to be processed in a final iteration.

* If needed, allocate unmet demand from higher level (MPA for example), to Development Sites as above.

=== Process Submodels and Interface to Other Models ===

* Once the core allocation model is finished,

** Run any needed 'Sub-models' for post-processing to create inputs to Travel Model ** Launch Travel Model run if this is configured for a scenario ** Resume model at next time step if this is configured for a scenario.

=== Generate Indicators and DataSets === As needed, once the model is completed on a Scenario:

* Configure indicators to produce from results, and the format for them: ** Tables ** Maps ** Charts (Petya says:)" Can we use the indicators to generate or edit feature classes and then use ArcMap to see the results? I would like whenever possible to organize the indicators in appropriate feature classes instead of keeping them in non-spatial columns of tables"""

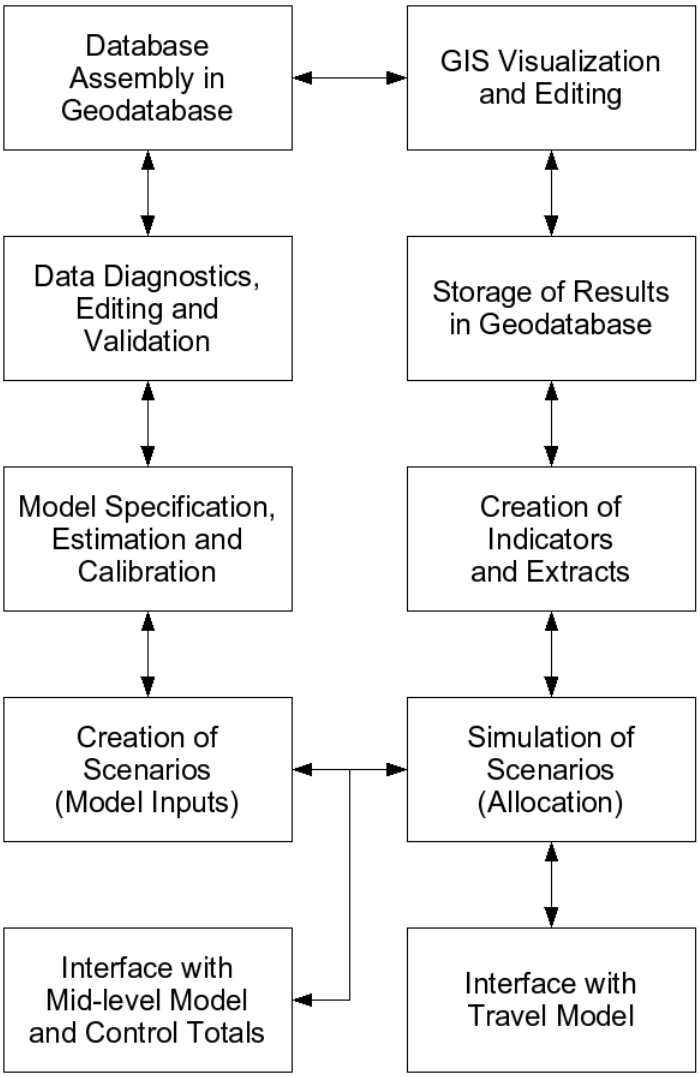


Figure 1: AZ-SMART Workflow Diagram

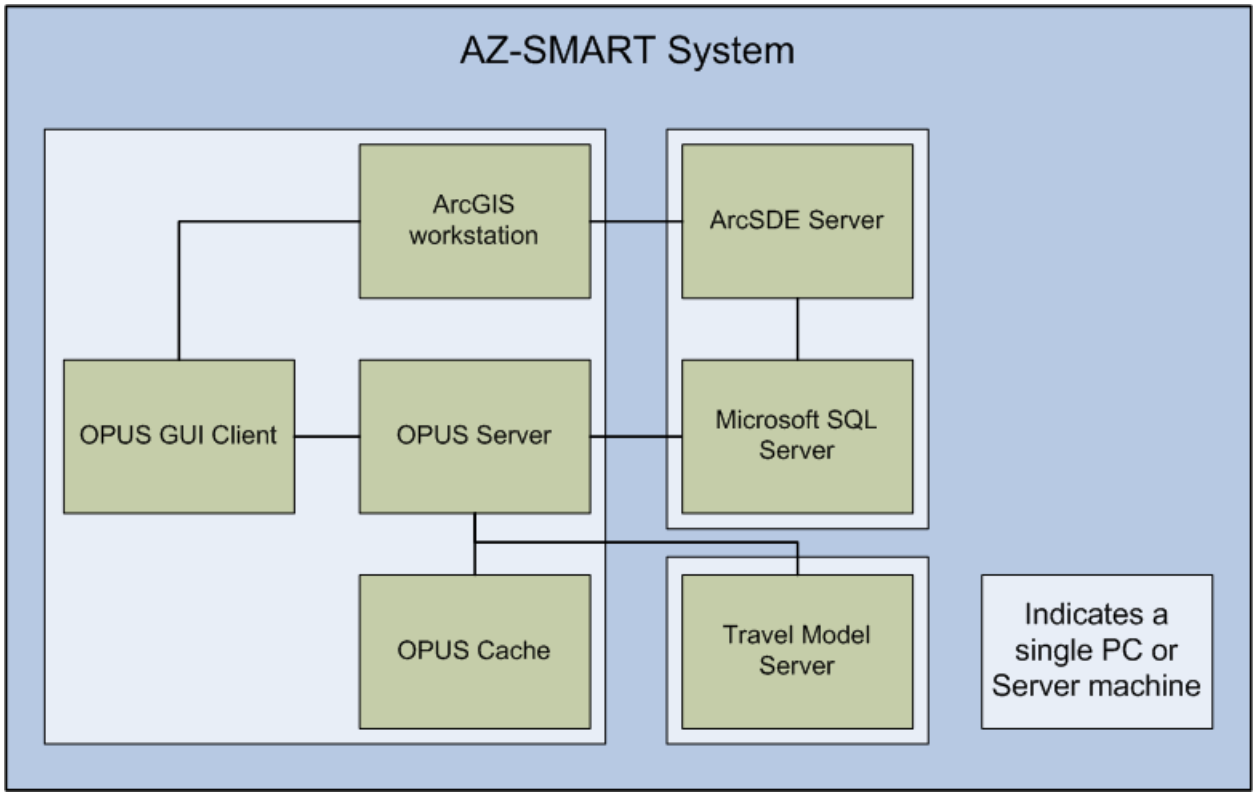


Figure 2: AZ-SMART System Architecture

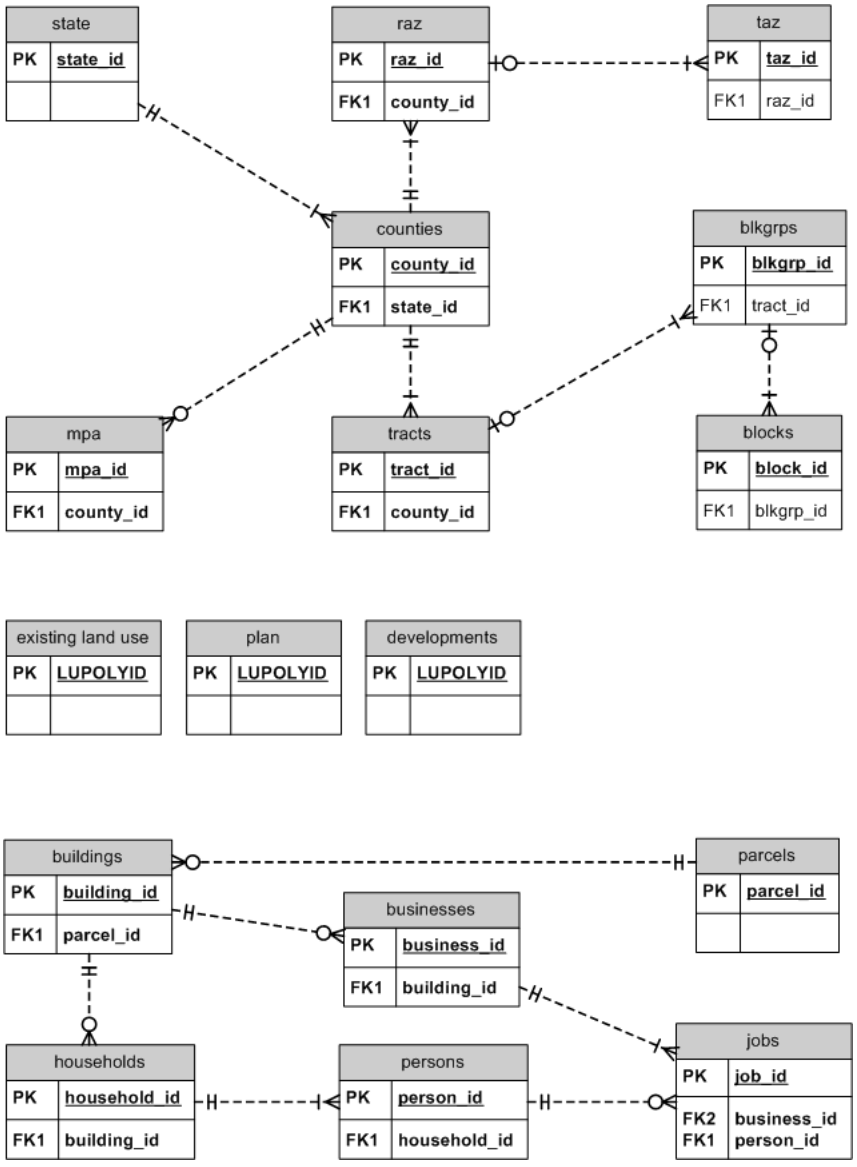


Figure 3: AZ-SMART Data Model