VISIONEVAL: A MODEL SYSTEM AND SOFTWARE FRAMEWORK FOR THE COLLABORATIVE DEVELOPMENT OF REGIONAL STRATEGIC PLANNING MODELS

Brian Gregor (corresponding author)
Oregon Systems Analytics LLC
PO Box 2211, Salem, OR 97308
Tel: 503-877-0349 Fax: 503-385-0352; Email: gregor@or-analytics.com

Tara Weidner
Oregon Department of Transportation
555 13th Street NE, Suite 2, Salem, OR 97301
Tel: 503-986-4226 Fax: 503-986-4174; Email: tara.j.weidner@odot.state.or.us

Jeremy Raw
Federal Highway Administration, Office of Planning
1200 New Jersey Avenue, SE, Washington, DC 20590
Tel: 202-366-0986; Email: jeremy.raw@dot.gov

Word count: 5,980 words text + 2 tables/figures x 250 words (each) = 6,480 words
ABSTRACT

Strategic planning is becoming increasingly important as a means to help state and metropolitan area governments select policies and actions to address pressing issues that are fraught with uncertainty. Strategic planning models allow exploration of many scenarios to assess policy/investment tradeoffs about complex systems enabling us to think better about intended and unintended consequences of our actions. A lineage of strategic planning models has been developed in recent years to help transportation planners meet strategic planning analysis needs, including GreenSTEP, RSPM, EERPAT and SmartGAP (now RPAT). Although development practices (open source licensing, modularity, continuous development chain using R) with the original GreenSTEP model facilitated agile and adaptable development of the other models, the structure falls short of what is needed to enable a fully open and collaborative development process which enables functionality to be easily shared between models. Those deficiencies have been addressed with the VisionEval open source modeling system and software framework presented here. The model system and software framework are built to support the creation of strategic planning models composed of a set of loosely coupled sub-models that can be combined in a ‘plug-and-play’ manner, enabling model development to be carried out in a cooperative distributed manner. This paper examines how the technical structure of VisionEval supports those goals.

OBJECTIVES AND MOTIVATIONS

Strategic planning is becoming increasingly important as a means to help state and metropolitan area governments prioritize policies and actions to address pressing issues that are fraught with uncertainty. Regions are faced with many concerns related to the development of sustainable transportation systems such as energy consumption, air quality, water, agricultural lands preservation, public health, and economic development, as well as uncertainties about the future. Moreover uncertainties regarding the potential effects of changing demographic, modal options, prices, and other things that affect travel behavior, makes it increasingly difficult to conduct traditional long term planning.

To help decision-makers and the public consider the implications of a highly uncertain future, models are more valuable as tools to explore many possible scenarios than to provide precise numerical forecasts. Model runtimes need to be short to enable many future scenarios to be modeled. Data exploration tools such as interactive data viewers are needed to help make sense of the results of modeling hundreds or thousands of scenarios and to improve understanding of the impacts and tradeoffs of different policy combinations and their resilience under a range of conditions. Strategic planning models and data visualizations allow exploration of many scenarios to assess policy/investment tradeoffs about complex systems, enabling us to think better about intended and unintended consequences of our actions, and to avoiding inconsistencies between stated community goals and the strategies that are pursued to achieve those goals.

Strategic models when used in a visioning process can be used to set goals and objectives to guide transportation plans and programs over a long time horizon. This is a key component of the performance-based planning and programming process being promulgated by the USDOT (1). The strategic visioning process does more than establish vaguely worded goals and objectives that may be used to substantiate any plan or project. It establishes a well-defined future vision which also identifies performance measures and targets and preferred strategies for meeting those targets, along with triggers that identify alternative actions or review if trends depart substantially from what was anticipated. Once established, the long term vision can serve as guideposts for monitoring progress over time, adjusting planned actions or the overall vision as the future uncertainties play out. There are substantial analytical challenges to developing a strategic transportation vision. First, the interactions of many factors (e.g. energy types, vehicle types, transportation costs, land use patterns, etc.) must be analyzed. Second, plans need to be robust across a broad range of future conditions. Third, measuring the adequacy of plans must encompass a wide range of performance measures of community values, from economic and equity concerns to public health and the environment; with traditional mobility largely an interim milestone in reaching these goals. Finally, the purpose is not to predict a future that must be accommodated, but rather to identify strategies for creating a future that is desired. These challenges call for a new type of model that emphasizes breadth rather than depth, that runs fast enough to enable the exploration of a large
decision space that may consist of hundreds of scenarios, and that can evaluate the consequences of
diverse qualitative and quantitative assumptions regarding future travel behavior.

A lineage of strategic planning models has been developed in recent years to help transportation
planners meet such strategic needs. The first of these, GreenSTEP, was developed by the Oregon
Department of Transportation (ODOT) to assist with the development of statewide strategies for reducing
greenhouse gas (GHG) emissions from the transportation sector (2, 3). GreenSTEP models travel, vehicle
ownership and other characteristics at the household level using simplified models built on a rich
disaggregate synthetic household dataset. Subsequently GreenSTEP became the basis for the FHWA’s
Energy and Emissions Reduction Policy Analysis Tool (EERPAT), ODOT’s Regional Strategic Planning
Model (RSPM), and the Rapid Policy Assessment Tool (RPAT, formerly SmartGAP, a product of the
SHRP 2 C16 project) (4, 5, 6).

Although GreenSTEP development practices (open source licensing, modularity, continuous
development chain using R) facilitated agile and adaptable development of the other models, the structure
falls short of what is needed to enable a fully open and collaborative development process which enables
functionality to be easily shared between models. Deficiencies include the lack of a formal API for
modules, overreliance on the global namespace to glue modules together, an insufficient data model for
persistent storage, and incomplete functionality and documentation to enable model re-estimation.

Open-source development is viewed by ODOT and the FHWA as a key technique for constructing
and improving strategic planning models in a cost-effective manner. The complex set of planning
considerations before us today suggest the need to focus on conceptualization and exploration. Thinking
about planning models in “open source” places the emphasis on the questions we need to ask, rather than
on model outputs computed based on inscrutable and perhaps obsolete assumptions. Open-source
development enables many people to share and compare their work and, by so doing, improves their
models’ responsiveness to contemporary policy needs.

To that end, these agencies commissioned work to develop an improved open-source model system
and software framework for strategic planning models. This VisionEval model system and framework
establishes an API and specifications for model modules, provides services to coordinate module
execution and to interact efficiently with a common datastore, and specifies suitable packaging of
modules to ensure complete functionality and documentation. This paper describes the model system and
software framework and its usefulness for the practice of strategic modeling of transportation futures
under uncertainty.

MODEL SYSTEM OBJECTIVES

The VisionEval model system and software framework is built to support the creation of strategic
planning models composed of a set of loosely coupled sub-models. Each sub-model is formulated as a
module which contains all of the functionality, parameters, and data specifications that are needed to
operate within the software framework. The modules are loosely coupled in that they interact with each
other only through the sharing of data contained in a common datastore, and they are estimated separately
so that modification or replacement of one module does not require changes to be made to other modules.

Objectives for the model system and software framework include:

• **Modularity**: The model system allows new capabilities to be added in a plug-and-play fashion so
  models can be improved and extended and so improvements developed for one model can be
easily shared with other models. Models are composed of modules that contain all of the data and
  functionality needed to calculate what they are intended to calculate.

• **Loose Coupling**: This objective is closely related to the *modularity* objective. Loose coupling is
  necessary if modules are to be added to or removed from models in a plug-and-play fashion.
  Loose coupling means that the parameter estimation for a sub-model is independent of the
  parameter estimation of any other sub-model. It also means that there is no direct communication
  between modules. All communication between modules is carried out through the transfer of data
  that is mediated by the software framework.
• **Openness**: The VisionEval software framework and all modules developed to operate in the framework will be completely open. Being open means more than sharing one’s work. It means completely revealing the concepts and algorithms so that others can assess how the module works. All module code, parameters, data, and specifications developed within VisionEval will be open to inspection and licensed to allow users to use, modify, and redistribute them as they see fit. In addition, modules will provide access to data and code to estimate the model that the module implements. Finally, a module will contain complete documentation that users may use to document the model that the module is a part of.

• **Geographic Scalability**: The model system enables models to be applied at a variety of geographic scales including metropolitan areas of various sizes, states of various sizes, and multi-state regions. Although models are applied at different scales, they share common geographic definitions to enable modules to be more readily shared.

• **Data Accessibility**: Model results will be saved in a datastore that is easy to query. Results can be filtered, aggregated, and post-processed to compute desired performance measures.

• **Regionalization**: Modules will have built in capabilities for estimating and calibrating sub-model parameters from regional data as needed.

• **Speed and Simplicity**: Since the intent of the model system is to support the development of strategic planning models, it is important that the models be able to address a large number of factors and be able to model a large number of scenarios. For this to occur, the framework needs to run efficiently and modules need to be simple and run quickly.

• **Preemptive Error Checking**: The model system should incorporate extensive data checking to clearly identify errors in the model setup and inputs before the model is run. The objective of early error checking is to avoid model runtime errors that waste model execution time and are difficult to debug.

• **Documentation**: Modules will include complete documentation so that they may be reviewed and understood by others. The calculations for estimating a module’s parameters will be included with the module.

• **Operating System Independence**: The model system should run on any of the 3 major operating systems; Windows, Apple, or Linux.

**SOFTWARE ENVIRONMENT**

As is the case with GreenSTEP and related models, the VisionEval model system is written in the R software environment for statistical computing and graphics (7). Well-supported and easily installed R implementations exist for all major operating systems. VisionEval modules will be distributed as standard R packages that can be compiled on all operating systems. Code that is written in another language may be included in a module package as long as it can be compiled in an R package that is usable on all 3 operating systems. There are several reasons for the choice of using the R language:

1. The existing code base for the GreenSTEP model and related models is written in R. Writing the VisionEval software framework in R enables this code base to be moved to the new framework with much less effort than would be required otherwise.

2. R is open-source software that is available on all major operating systems so VisonEval will be system independent.

3. R has a very good package system that is well supported with tools to design and build both code and documentation, and they fully support a “literate programming” development model in which documentation matters as much as working code. This simplifies the development of the software framework and simplifies the process for module developers to produce complete and well documented modules.
4. R has the most extensive set of statistical and other data analysis packages available. Because of this, almost any type of model can be estimated using R and therefore, modules can contain not only full documentation of model estimation, but also scripts that allow model estimation to be replicated and rerun using regional data.

5. R is well supported with capable (and free) integrated development environments. Because the state of objects can be easily queried, the process of building and testing models is simplified. This makes it easier for modelers who don't come from a computer science background to develop models.

6. Although R is an interpreted language (and in this respect resembles many workhorse modern programming languages such as Java, Python and C#), most of its core functions are written in very efficient lower-level languages such as C. This means that most R programs can carry out many operations very quickly. In addition, it is relatively easy to develop and call functions written in compiled languages such as C++, C, and Fortran from R. When pure R code is not fast enough, it is straightforward to move key functions into a compiled language.

7. R has a large user base and extensive formal and informal documentation so it is relatively easy for users to get answers to programming questions.

**MODEL SYSTEM OVERVIEW**

The VisionEval model system is composed of 3 layers (Figure 1):

1. **Model**: The model layer organizes all of the modules into a coherent model. The model layer also includes a common datastore that is used by all modules.

2. **Modules**: The module layer manages the individual modules that contain all of the code and parameters to implement the model.

3. **Software Framework**: The software framework layer provides the functionality for controlling a model run, running individual modules, and interacting with the common datastore.
A VisionEval model is built from a set of compatible modules, a set of specifications for the model (including the geography over which it operates), a set of scenario input files, and a simple R script that initializes and runs the model. Following is a simple example of a model script from a demo that is included in the GitHub repository for the project (http://github.com/gerorgbj/VisionEval). The lines in the script after the ‘runModule’ section show simple queries of the Datastore.

```r
#=============  
#run_model.R  
#=============  

#Load VisionEval package  
library(visioneval)  

#Initialize model  
initializeModel(Dir = "defs", ParamFile = "parameters.json", GeoFile = "geo.csv")  

#Run all demo modules for all years  
for (Year in getYears()) {
```
runModule("CreateHouseholds", "vedemo1", Year = Year)
runModule("CreateBzones", "vedemo1", Year = Year)
runModule("CreateBzoneDev", "vedemo1", Year = Year)
}

# Datastore queries on results
# Display household size distribution in Household table
table(readFromTable("HhSize", "Household", "2010"))
table(readFromTable("HhSize", "Household", "2050"))
# Check Bzone density distribution in Bzone table
plot(density(readFromTable("PopDen", "Bzone", "2010"), bw=200))
lines(density(readFromTable("PopDen", "Bzone", "2050"), bw=200), col="red")

The script calls two functions that are defined by the software framework; ‘initializeModel’ and ‘runModule’. The ‘initializeModel’ function sets up the model environment and model datastore, checks that all necessary modules are installed, and checks whether all module data dependencies can be satisfied. The arguments of the ‘initializeModel’ function identify where key model definition data are found. The ‘runModule’ function invokes a specific module. The arguments of the ‘runModule’ function identify the name of the module to be run, the R package that contains the module, and the forecast year. There are several other optional arguments that are discussed below. This approach makes it easy for users to combine modules in a 'plug-and-play' fashion. One simply identifies the modules that will be run and the sequence that they will be run in. This is possible in large part for the following reasons.

1. The modules are loosely coupled. Modules only communicate to one another by passing information to and from the datastore.
2. The framework establishes standards for key shared aspects of modules including how data attributes are specified and how geography is represented.
3. Every module includes detailed specifications for input and output data that it manipulates. These data specifications serve as contracts between modules which the datastore keeps track of and the framework software enforces.

**MODEL SYSTEM GEOGRAPHY**

The design of the model system includes the specification of a flexible standard for model geography in order to fulfill the objectives of modularity and geographic scalability. It specifies levels of geographical units, their names, their relative sizes, and the hierarchical relationships between them. It is flexible in that it allows geographical boundaries to be determined by the user and it allows the units in some geographical levels to be simulated rather than being tied to actual physical locations. Allowing simulation of one or more geographic levels enables modules to be shared between models that operate at different scales. For example a statewide model and a metropolitan area model could use the same module for assigning households to land development types even though the statewide model lacks the fine scale of geography of the metropolitan model.

Following is the definition of the geographic structure of the VisionEval model system:

- **Region**: The region is the entire model area. Large-scale characteristics that don't vary across the region are specified at the region level. Examples include fuel prices and the carbon intensities of fuels.

- **Azones**: These are large subdivisions of the region containing populations that are similar in size to those of counties or Census Public Use Microdata Areas (PUMAs). The counties used in the GreenSTEP and EERPAT models and metropolitan divisions used in the RSPM are examples. Azones are used to represent population and economic characteristics that vary across the region such as demographic forecasts of persons by age group and average per capita income. Azones are the only level of geography that is required to represent actual geographic areas and may not be simulated.
**Bzones:** These are subdivisions of Azones that are similar in size to Census Tracts and Census Block Groups. Bzones are used to represent neighborhood characteristics and policies that may be applied differently by neighborhood, for example in the RSPM:

- District population density is a variable used in several sub-models;
- An inventory of housing units by type by district is a land use input; and,
- Carsharing inputs are specified by district.

In rural areas, Bzones can be used to distinguish small cities from unincorporated areas. Bzones may correspond to actual geographic areas or may be simulated. For example, a sub-model of the GreenSTEP model estimates the likely distribution of Census Tract densities for a metropolitan area given an overall population density for the metropolitan area. This sub-model will be converted in the model system so that it creates a set of simulated Bzones having densities that represent the likely distribution of Census Tract densities.

**Czones:** These are subdivisions of Bzones that describe more detailed land use characteristics such as whether an area is typified by residential development, commercial development, or mixed-use development. These characteristics may be described using a classification system such as the development type system, used by the GreenSTEP and RSPM models, or the place type system used by the RPAT model. A Czone may be any size, up to and including the entire Bzone that it is located within. As with Bzones, Czones may correspond to actual geographic areas such as the traffic analysis zones defined for an urban travel demand model. Perhaps more commonly, Czones will be synthesized based on scenario inputs and attributes of the Azones and Bzones they are situated within.

**Mareas:** These denote an association with urbanized portions of metropolitan areas. Mareas are used to specify and model urbanized area transportation characteristics such as overall transportation supply (transit, highways) and congestion. A Marea does not exist in a strict hierarchical relationship with Azones and Bzones because the Marea boundary is likely to change over time as the urbanized area population grows. While a Marea will be associated with one or more Azones and Bzones, its boundary may be located inside or outside the boundaries of the Azones and Bzones it is associated with. Czones are used to link the portions Bzones and Azones located within Mareas.

**MODEL DATASTORE**

Currently GreenSTEP/RSPM and related models store data in R binary files (".rda" files). The largest of these files are the simulated household files which store all of the information for all simulated households in an Azone (e.g. counties in GreenSTEP). All the data for households in the Azone are stored in a data frame where each row corresponds to a record of an individual household and the columns are household attributes. Vehicle data for households are stored as lists in the data frame. This approach has had some advantages:

- Storage and retrieval are part of the language: one line of code stores a data frame and one line of code retrieves it;
- It is easy to apply models to data frames; and,
- Vehicle data can be stored as lists within a household data frame, eliminating the need to join tables.

The simplicity of this approach helped move GreenSTEP from a concept into an operational model quickly. However, several limitations have emerged as GreenSTEP and related models have been used in subsequent applications, including:
• Large amounts of computer memory are required when modeling Azones that have large populations. This necessitates either expanding computer memory or limiting the size of Azones.

• It is not easy to produce summary statistics from the simulated household files for a region.

• The number of non-household data files has proliferated in order to store various aggregations for use in the model and for later summarization.

The VisionEval model system uses the HDF5 file format for storing model data, rather than using R binary files. The HDF5 file format was developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois and other contributors to handle extremely large and complex data collections. For example, it is used to store data from particle simulations and climate models. It also is the basis for the new open matrix standard for transportation modeling, OMX (http://github.com/osPlanning/omx) (8). The HDF5 format was chosen over SQL databases because it reads and writes data faster than SQL alternatives that were tested. It also provides random data access in a way that is analogous to how data in R data structures can be indexed. Large complex data sets can be randomly accessed by organizing datasets in a hierarchy of groups and by accessing portions of datasets by using indexing. In addition, metadata can be stored as attributes of any data group or data item.

An HDF5 file is composed of groups and datasets. Groups provide the overall structure for organizing data, just as a file system provides the structure for organizing files on a computer disk. The structure of the VisionEval datastore is shown in Figure 2 which is organized like a directory tree. The first level of the tree denotes groups that are collections of tables. There are two types of groups at this level: the 'Global' group, and 'forecast year' groups (e.g. 2010, 2050). The 'Global' group is used to store data that is not organized by forecast year and that applies to the entire model region. The vehicle characteristics scenario inputs used in the GreenSTEP, RSPM, and EERPAT models are examples. These data are organized by vehicle model year rather than forecast year. The 'forecast year' groups store datasets that are organized by forecast year. For example, regional cost inputs vary by forecast year. The second level of the tree denotes groups that represent data tables. These groups hold collections of datasets (vectors of data) that hold different types of data but all have the same lengths so that the corresponding position in each dataset (i.e. vector) relates to the same record in a data table. Tables in the Global group are organized by 'topic' (e.g. HEV vehicle characteristics, EV vehicle characteristics). Tables in 'forecast year' groups include tables for each level of geography as well as 'Households' and 'Vehicles' tables which store attributes of the simulated households and the vehicles they own.
This structure is adequate to store all of the data that are used by the GreenSTEP/RSPM models and their offshoots. It can also be easily expanded to serve new modeling capabilities. For example if a module is added to model building stock, a 'Buildings' table could be added to each 'forecast year' group. In addition, since HDF5 files can store matrix data as well as vector data, if a future module makes use of a distance matrix, that matrix could be added to either the 'Global' group or the 'forecast years' groups.

**INTERACTION OF MODULES AND THE SOFTWARE FRAMEWORK**

Modules and the software framework have distinct responsibilities. Modules encapsulate the functionality, parameters and specifications required to compute specific quantities. They describe what is to be accomplished (e.g. predicting auto ownership) and how it is to be accomplished. Each module is responsible for completely specifying the data it uses, including scenario input files, and information coming from the datastore. The framework uses these specifications to check that the data every module needs will be available when the module needs it and that the data will have the correct structure and content. This is done before any of the modules are run to avoid runtime errors. The framework is responsible for setting up the runtime environment and datastore, managing the state of the model run, interacting with the datastore, and managing the execution of modules. The framework also manages the logging of messages that framework functions or module functions produce. Modules can work in a plug and play fashion in the VisionEval model system because they adhere to a set of specifications for how they are structured.

For example when the ‘runModule’ function is called as in the example above –

```python
runModule("CreateHouseholds", "vedemo1", Year = Year) -
```

the specifications enable the software framework to run the model as follows:
The ‘CreateHouseholds’ module is contained in a standard R package named “vedemo1”. Because it is a standard R package, the ‘runModule’ function can load the namespace for the package, making all of the contents addressable. The ‘runModule’ function loads the data specifications for the ‘CreateHouseholds’ module. These specifications are contained in a list named ‘CreateHouseholdsSpecifications’. The name of the data specifications for a module is always the name of the module concatenated with ‘Specifications’.

The data specifications for a module are contained in a list that has 4 named components: ‘RunBy’, ‘Inp’, ‘Get’, and ‘Set’. The ‘RunBy’ specification identifies the level of geography that the module is to be run at. For example, a congestion module would be run at the Marea level. The ‘Inp’ specifications identify all of the scenario input items that must be processed and saved in the datastore. The ‘Get’ specifications identify all of the data items that must be retrieved from the datastore as inputs for the module. The ‘Set’ specifications identify all of the data items produced by the module that the ‘runModule’ function is to save in the datastore. These specifications are described in more detail below.

The ‘runModule’ function follows the directions in the ‘Inp’ specifications to load all of the scenario inputs that the module depends on into the datastore. The correctness of the inputs has been checked previously when the ‘initializeModel’ function is executed. Therefore, when the ‘runModule’ function is invoked, it is assured that the specified inputs can be loaded and are correct.

The ‘runModule’ function reads in the main module function that controls all of the module calculations and assigns it to a local function. The main module function has the same name as the module so the ‘runModule’ function can load it using the module name. All other resources that the main function needs are included in the package that it is contained in. These resources are accessible to it and to the local function that it is assigned to. This includes any sub-model parameters that it uses and any other functions that it calls.

The ‘runModule’ function follows the ‘Get’ specifications to load from the datastore all of the input data required by the module into a list. It then calls the local function, to which the main module function has been assigned, and passes to it the list of inputs.

The return value for the local function is assigned to another list. That list includes all of the outputs of the module that are to be saved to the datastore. The ‘runModule’ function follows the ‘Set’ specifications to determine what data are to be saved in the datastore and how they are to be saved.

The specifications establish ‘contracts’ that enable modules to work together. The framework software enforces those contracts. To be enforceable, the contracts need to meet specifications as well. The following code extract from the ‘CreateHouseholds’ demonstration module show an example of specifications included in the ‘Inp’ component:

```r
Inp = items(
  item(
    NAME = "Population",
    FILE = "azone_population.csv",
    TABLE = "Azone",
    TYPE = "integer",
    UNITS = "persons",
    NAVALUE = -1,
    SIZE = 0,
    PROHIBIT = c("NA", "< 0"),
    ISELEMENTOF = "",
    UNLIKELY = "",
  ))
```
TOTAL = ""

These specifications have the following meanings:
• FILE: the name of the file where the data are located;
• NAME: the name of the field in the input file containing the data and the name the dataset will be
given in the datastore;
• TABLE: the name of the table in the datastore where the dataset will be placed;
• TYPE: the data type (i.e. “integer”, “double”, “character”, “logical”);
• UNITS: the units of measure for the data;
• NAVALUE: value to be used to represent missing values in the datastore;
• SIZE: the maximum number of characters for character data;
• PROHIBIT: conditions that identify prohibited values for the data (if specified);
• ISELEMENTOF: allowed elements for categorical data (if specified);
• UNLIKELY: conditions that identify values that are unlikely (triggering a warning rather than an
error if specified); and,
• TOTAL: Required sum of all values (if specified).

The model system specifications for modules, by enabling modules to work together in a
“plug-and-play” fashion, also enables model developers to more easily collaborate on model
development. Because developers know that their sub-models will work with other sub-models if they
meet the model system specifications, they can collaborate on building a model even without any formal
structure for collaboration. The model system enables this collaboration in several additional ways. First,
all the code contained in a module must be licensed under an open source license which allows others to
use, modify, and redistribute the code. Second, all modules are required to be documented and
accompanied by code which shows how the sub-model was specified and its parameters estimated. Third,
all modules must include tests to evaluate whether they work as intended. These requirements help ensure
that developers can build upon the work of other developers. They also help to create an environment
where the quality of models is continuously improved over time because all modules are open to review,
testing, and improvement. This approach is similar to that of the R software environment for statistical
computing and graphics, which has produced a wide range of very high quality statistical software.

MAJOR RESULTS
Phase 1 of the VisionEval model system development has been completed. This includes documenting the
model system design, developing the software framework code, and developing a demonstration package
of modules and demonstration model setup. All work products are available in a public GitHub repository
(https://github.com/gregorbj/VisionEval). Phase 2, to convert the existing models to the model system,
will proceed through the course of 2016.

IMPLICATIONS FOR PRACTICE OF TRAVEL MODELING
There are several implications of this work for the practice of travel modeling. This new framework:
1. Improves robustness, usability, and documentation for existing strategic planning models
   (GreenSTEP, EERPAT, RSPM, RPAT);
2. Makes it much easier to share functionality between models and to build new models from
   existing modules; and,
3. Enables collaboration through open-source resources to increase the scope and quality of strategic
   planning models.

The first and most immediate result of the strategic planning model framework will be to improve the
robustness, usability and documentation of existing models. The data checking functionality will
eliminate practically all of the runtime errors that occur with the existing models. The logging functionality will also help users to identify inputs that, while not causing errors, might not fall within acceptable ranges. The regionalization functionality will greatly simplify the effort of deploying these models in new areas. In addition, the datastore and functions to query it will make the calculation of performance measures much simpler. Finally, improved documentation will make each component module easier to understand and will simplify the development of overall model documentation.

Second, the deployment of a framework which enables modules to be used in a plug-and-play fashion will make it much easier to share functionality between models and to build new models from the components of existing models. For example, the RSPM includes a housing type choice model that was developed to improve the matching of household characteristics with land use characteristics (e.g., development density). The framework will enable this model to be used in the GreenSTEP and EERPAT models as well. The structure and capabilities of the framework will enable agencies and consultants to modify existing models to meet their strategic planning analysis needs. They can start with an existing model, add modules from other models, modify existing modules, and create new modules. Researchers can more easily extend code to use for their own purposes, leading to tool innovation across multiple disciplines.

Finally, the framework establishes an open source modeling system that enables collaboration, investment efficiency, and quality control benefits. Public funds are not spent doing something more than once, flexibility with consultants is increased because the tool is not proprietary, potential reuse provides incentives for the development team to follow best practices (e.g., thorough documentation and portability), and the codebase as well as the collaborative process can serve as a reference and help to introduce the project’s benefits to the larger community. This supports a transparent process of maintaining a credible tool with a code base that is maintained and updated by a community of active users.

ACKNOWLEDGEMENTS
The work described in this paper was supported through funding from the Oregon Department of Transportation; the Federal Highway Administration Office of Planning, Environment and Realty; and the Strategic Highway Research Program (SHRP 2).

REFERENCES