SACSIM/05
Activity-Based Travel Forecasting Model for SACOG
Featuring ***DaySim***—the Person Day Activity and Travel Simulator

Technical Memo Number 3

Design of Model System Application Software

July 31, 2006 – Draft 2

Prepared for

Sacramento Area Council of Governments

Prepared by

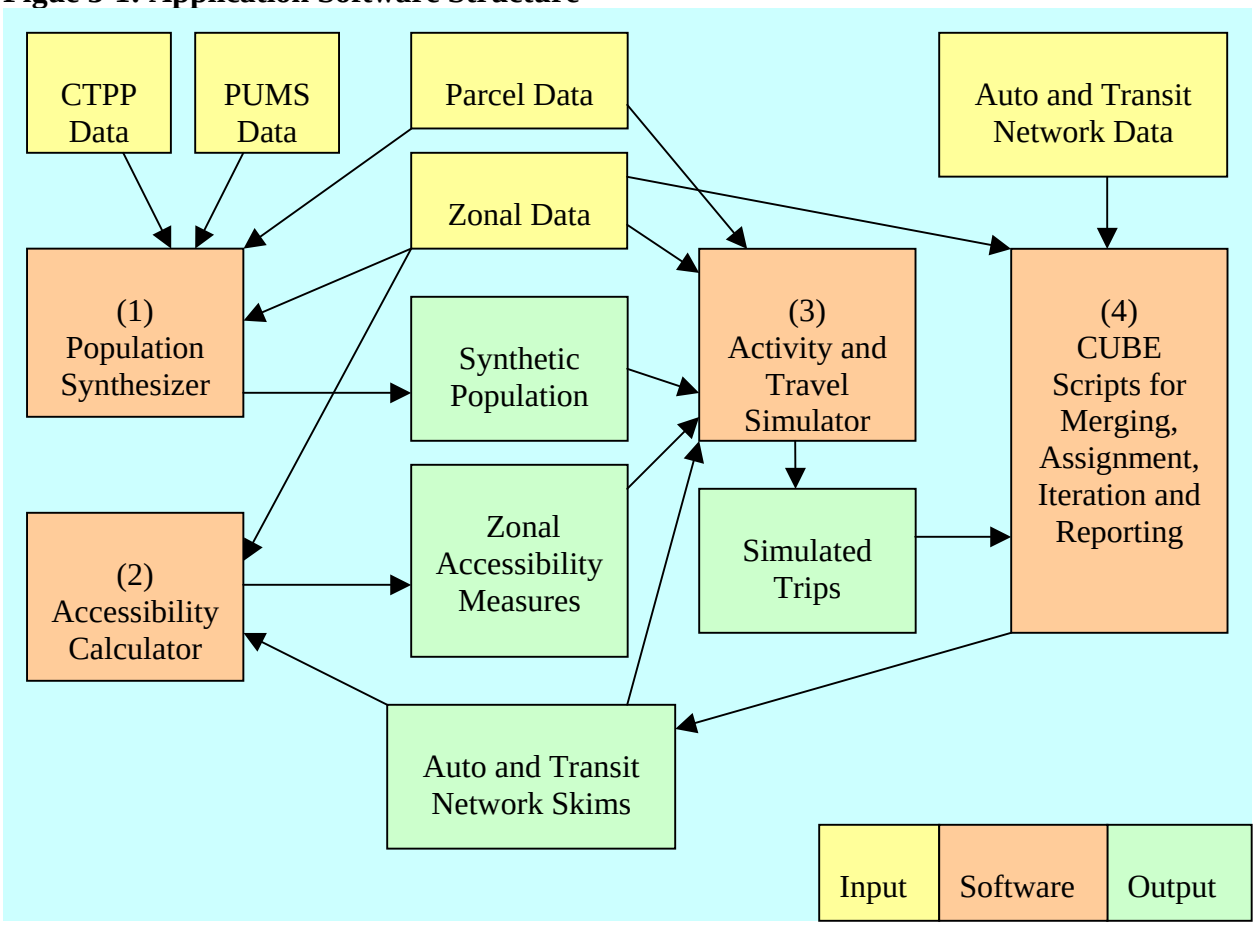
John L. Bowman, Ph. D.
Transportation Systems and Decision Sciences
28 Beals Street, Brookline, MA 02446 USA
+1-617-232-8189 John_L_Bowman@alum.mit.edu http://JBowman.net

Mark Bradley
Bradley Research & Consulting
524 Arroyo Ave., Santa Barbara, CA 93109, USA.
+1-805-564-3908 mark_bradley@cox.net

Introduction

The custom software for applying the SACSIM/05 model system is written in Pascal code compiled under the Borland Delphi 2005 compiler (which will also compile C code). The software will fit into the Cube Application framework as shown in Figure 3-1.

Figue 3-1: Application Software Structure



The first module, which is run only once outside the network assignment iteration loop, is the Population Synthesizer, as documented in Tech Memo 2. This program uses only exogenous input files (CTPP, PUMS, zonal, and parcel data) that do not change during the simulation. It produces the synthetic population file that is used in the Activity and Travel Simulator.

The second module, is the Accessibility Calculator, to pre-calculate accessibility measures at the zonal level using simple mode and destination choice models for various non-mandatory activity purposes. The reason for pre-calculating these measures outside of the main Activity and Travel Simulator population loop is to reduce run-time, as the same values can be used for all persons living in specific residence zones. The calculations use both zonal data and auto and transit skim

data, so must be rerun during each iteration of the assignment loop. More detail on this program is given in Tech Memo 11.

The third module is the core of the model system, the Activity and Travel Simulator. This module also needs to be rerun during each iteration of the assignment loop, because it uses the updated auto and transit network skims. The output of the Simulator is a list of day, tour and trip records for all persons in the synthetic population. The remaining sections of this memo describe this software in detail.

All three of these modules have been programmed into a single executable, DAYSIM05.EXE. Tech Memo 10 gives details on running this program. Briefly, the program relies on a short control file with a simple text format. The control files are used to:

- Specify the names of all input and output files used by the program.
- Specify values for all user-defined parameters and options used in the program.

The remaining code that is necessary is a series of scripts in Cube Voyager to:

- Convert the trip record file into trip matrices for all desired mode/time-of-day/purpose combinations.
- Perform network traffic assignment for the relevant auto trip matrices.
- Control the iteration of the entire model system to stabilize the congested auto flows and travel times.
- Perform transit assignment on the final transit trip matrices.
- Perform other desired reporting tasks at the trip, tour, person-day, and area levels, taking advantage of CUBE's new built-in table and graph generation facilities.

CUBE also functions as a shell to run this entire system of models, calling the stand-alone executable program when needed.

Logic and Structure of DaySim05

Figure 1 outlines the main structure of program:

The models and structure are divided into 3 main parts:

1. Longer term models
    1.1. Usual work location for workers
    1.2. Usual school location
    1.3. Usual work location for students
    1.4. Auto ownership
2. Person-day level models
    2.1. Activity pattern- presence of tours and stops by purpose
    2.2. Exact number of tours by purpose
3. Tour level models
    3.1. Primary destination choice
    3.2. Number of work-based subours (work tours only)
    3.3. Main mode choice
    3.4. Primary activity begin and end times (to 30 minute periods)
4. Trip level models
    4.1. Half tour number and purpose of intermediate stops
    4.2. Intermediate stop location choice
    4.3. Trip mode choice
    4.4. Intermediate stop departure time (to 30 minute periods)

Figure 1—DaySim models (numbered) within the program looping structure

```
Begin
  {Read run controls, model coefficients, TAZ data, LOS matrices,
                  population controls, and Parcel data into memory}
  {Draw a synthetic household sample if specified}
  {Pre-calculate destination sampling probabilities}
  {Pre-calculate (or read in) TAZ aggregate accessibility arrays}
  {Open other input and output files}
  {Main loop on households}
    {Loop on persons in HH}
       {Apply model 1.1 Work Location for workers}
       {Apply model 1.2 School Location for students}
       {Apply model 1.1 Work Location for students}
    {End loop on persons in HH}
    {Apply model 1.3 Household Auto Availability }
    {Loop on all persons within HH}
        {Apply model 2.1 Activity Pattern (0/1+ tours and 0/1+ stops)
          and model 2.2 Exact Number of Tours for 7 purposes}
       {Count total home-based tours and assign purposes}
       {Initialize tour and stop counters and time window for the person-day before looping on tours}
       {If there are tours, loop on home-based tours within person in tour priority sequence,
             with tour priority determined by purpose and person type}
          {Increment number of home-based tours simulated for tour purpose (including current)}
          {Apply model 3.1 Tour destination}
          {If work tour, apply model 3.2 Number and purpose of work-based subtours}
          {Loop on predicted work-based sub tours and insert then tour array after current tour}
          {Apply model 3.3 Tour mode}
          {Apply model 3.4 Tour primary destination arrival and departure times}
          {Loop on tour halves (before and after primary activity)}
             {Apply model 4.1Half tour stop frequency and purpose}
             {Loop on trips within home-based half tour (in reverse temporal order for 1st tour half)}
                {Increment number of stops simulated  for stop purpose (including current)}
                {Apply model 4.2 Intermediate stop location}
                {Apply model 4.3 Trip mode}
                {Apply model 4.4 Intermediate stop departure time}
                {Update the remaining time window}
             {End loop on trips within half tour}
          {End loop on tour halves}
       {End loop on tours within person}
       {Write output records for person-day and all tours and trips}
    {End loop on persons within household}
  {End loop on Households}
 {Close files}
 {Create usual work location flow validation statistics}
End.
```

This structure was first programmed to work with the household survey data, with observed variables in the data records representing the "chosen" alternative for each choice model. This approach had a few major advantages:

- It ensured that the structure of the choice models would agree with the data available to estimate the models, in terms of the data that is known at each point in the conditional hierarchy.

- The software was used to generate the estimation files for many of the models that were estimated.

- Model application code was written and added in a modular fashion. At a certain point in the program logic, instead of writing out an estimation file record, we can apply the model coefficients to that same data, and compare the predicted choices against the observed choices in the household survey data.

- Once the code is there to predict all activity and travel choice variables instead of observing them, we can then use the PUMS records in the synthetic sample instead of the household survey to provide the remaining input data on household and person characteristics.

In fact, the software has been programmed to easily substitute the household survey sample for the synthetic household sample, and, in that case, to either keep the observed choices for each model or to predict new choices. This facilitates validation checking of the models and code.