

SHRP 2 Capacity Project C10A

Start-up Guide for the Dynamic, Integrated Model System: Burlington-Area and Jacksonville- Area Applications

SHRP 2 Capacity Project C10A

Start-up Guide for the Dynamic, Integrated Model System: Burlington-Area and Jacksonville-Area Applications

RSG

AECOM

Mark Bradley Research & Consulting

John Bowman Research & Consulting

Mohammed Hadi, FIU

Ram Pendyala, ASU

Chandra Bhat, UT Austin

Travis Waller, UT Austin

TRANSPORTATION RESEARCH BOARD

Washington, D.C.

2014

www.TRB.org

© 2014 National Academy of Sciences. All rights reserved.

ACKNOWLEDGMENT

This work was sponsored by the Federal Highway Administration in cooperation with the American Association of State Highway and Transportation Officials. It was conducted in the second Strategic Highway Research Program, which is administered by the Transportation Research Board of the National Academies.

COPYRIGHT INFORMATION

Authors herein are responsible for the authenticity of their materials and for obtaining written permissions from publishers or persons who own the copyright to any previously published or copyrighted material used herein.

The second Strategic Highway Research Program grants permission to reproduce material in this publication for classroom and not-for-profit purposes. Permission is given with the understanding that none of the material will be used to imply TRB, AASHTO, or FHWA endorsement of a particular product, method, or practice. It is expected that those reproducing material in this document for educational and not-for-profit purposes will give appropriate acknowledgment of the source of any reprinted or reproduced material. For other uses of the material, request permission from SHRP 2.

NOTICE

The project that is the subject of this document was a part of the second Strategic Highway Research Program, conducted by the Transportation Research Board with the approval of the Governing Board of the National Research Council.

The Transportation Research Board of the National Academies, the National Research Council, and the sponsors of the second Strategic Highway Research Program do not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of the report.

DISCLAIMER

The opinions and conclusions expressed or implied in this document are those of the researchers who performed the research. They are not necessarily those of the second Strategic Highway Research Program, the Transportation Research Board, the National Research Council, or the program sponsors. The information contained in this document was taken directly from the submission of the authors. This material has not been edited by the Transportation Research Board.

SPECIAL NOTE: This document IS NOT an official publication of the second Strategic Highway Research Program, the Transportation Research Board, the National Research Council, or the National Academies.

THE NATIONAL ACADEMIES

Advisers to the Nation on Science, Engineering, and Medicine

The **National Academy of Sciences** is a private, nonprofit, self-perpetuating society of distinguished scholars engaged in scientific and engineering research, dedicated to the furtherance of science and technology and to their use for the general welfare. On the authority of the charter granted to it by Congress in 1863, the Academy has a mandate that requires it to advise the federal government on scientific and technical matters. Dr. Ralph J. Cicerone is president of the National Academy of Sciences.

The **National Academy of Engineering** was established in 1964, under the charter of the National Academy of Sciences, as a parallel organization of outstanding engineers. It is autonomous in its administration and in the selection of its members, sharing with the National Academy of Sciences the responsibility for advising the federal government. The National Academy of Engineering also sponsors engineering programs aimed at meeting national needs, encourages education and research, and recognizes the superior achievements of engineers. Dr. C. D. (Dan) Mote, Jr., is president of the National Academy of Engineering.

The **Institute of Medicine** was established in 1970 by the National Academy of Sciences to secure the services of eminent members of appropriate professions in the examination of policy matters pertaining to the health of the public. The Institute acts under the responsibility given to the National Academy of Sciences by its congressional charter to be an adviser to the federal government and, upon its own initiative, to identify issues of medical care, research, and education. Dr. Victor J. Dzau is president of the Institute of Medicine.

The **National Research Council** was organized by the National Academy of Sciences in 1916 to associate the broad community of science and technology with the Academy's purposes of furthering knowledge and advising the federal government. Functioning in accordance with general policies determined by the Academy, the Council has become the principal operating agency of both the National Academy of Sciences and the National Academy of Engineering in providing services to the government, the public, and the scientific and engineering communities. The Council is administered jointly by both Academies and the Institute of Medicine. Dr. Ralph J. Cicerone and Dr. C.D. (Dan) Mote, Jr., are chair and vice chair, respectively, of the National Research Council.

The **Transportation Research Board** is one of six major divisions of the National Research Council. The mission of the Transportation Research Board is to provide leadership in transportation innovation and progress through research and information exchange, conducted within a setting that is objective, interdisciplinary, and multimodal. The Board's varied activities annually engage about 7,000 engineers, scientists, and other transportation researchers and practitioners from the public and private sectors and academia, all of whom contribute their expertise in the public interest. The program is supported by state transportation departments, federal agencies including the component administrations of the U.S. Department of Transportation, and other organizations and individuals interested in the development of transportation. **www.TRB.org**

www.national-academies.org

About This Guide

Project C10A developed an application of the DaySim activity-based demand model and a TRANSIMS network for Burlington, Vermont, to test linking the demand and network models in a relatively small environment. The research team did the design and troubleshooting work on the Burlington model set to take advantage of quicker run times. Once the integrated model system worked in Burlington, the model structure was transferred to the larger Jacksonville, Florida, area.

This document contains two start-up guides, one for the Burlington model set and one for the Jacksonville-area model set. They are similar for the reasons described above, but they are not the same. The start-up guides describe the file structures and provide step-by-step instruction to get started on each set.

The model files are a “snapshot” of the final files developed under the project. The files provided by SHRP 2 and FHWA for the project should be used rather than the latest versions on the market. All of the software components continue to change in the open source environment, so the final versions provided should be used as a starting point. See the reports on the project—*SHRP 2 C10A: Dynamic, Integrated Model System: Jacksonville-Area Application* and *SHRP 2 C10A: Transferability of Activity-Based Model Parameters*—for more information.—*Stephen J. Andrlle, SHRP 2 Deputy Director*

Contents for Part 1: Burlington, Vermont

2	CHAPTER 1	Introduction
3	CHAPTER 2	Model Directory Structure
3	2.1	SHRP2_C10A_v3-12-21-2012_Base
5	2.2	Burlington_Model Directory
8	2.3	RTE Directory
10	2.4	TRANSIMS
11	CHAPTER 3	Operating Systems and Hardware
11	3.1	Specifying Operating System
12	3.2	Specifying Partitioning Variables
14	CHAPTER 4	Running the Integrated Model System
19	4.1	Model Runtimes
19	4.2	File Naming Convention

Contents for Part 2: Jacksonville, Florida

22	CHAPTER 1	Introduction
23	CHAPTER 2	Model Directory Structure
23	2.1	Jacksonville_Burlington_Daysim_TransiMs_V5_Studio_Model_V1
25	2.2	Burlington_Model And Jacksonville_Planning_Model Directories
29	2.3	Rte_Burl And Rt_Jax Directories
30	2.4	1_Software
32	CHAPTER 3	Operating Systems And Hardware
32	3.1	Specifying Operating System
33	3.2	Specifying Partitioning Variables
35	CHAPTER 4	Running The Integrated Model System
40	4.1	Model Runtimes
40	4.2	File Naming Convention

PART 1: Burlington, Vermont

CHAPTER 1

Introduction

The primary objective of the C10A project is to make operational a dynamic, integrated model—that is, an integrated, advanced travel demand model with a fine-grained, time-sensitive network—and to demonstrate the model’s performance through validation tests and policy analyses. This integrated model system is necessary because most current travel models are not sufficiently sensitive to the dynamic interplay between travel behavior and network conditions and are unable to reasonably represent the effects of transportation policies such as variable road pricing and travel demand management strategies. The model system was designed to capture changes in demand, such as time-of-day choice (i.e., peak spreading), destination, mode, and route choice in response to capacity and operational improvements such as signal coordination, freeway management, and variable tolls.

The purpose of this start-up guide is to provide information necessary for an end user to develop a working knowledge of the DaySim-TRANSIMS integrated model system (referred to as “the model”). The end-user start-up guide is a technical document for travel modeling practitioners who are interested in studying the configuration and specification of the system and ultimately running the model.

The start-up guide does *not* represent and/or address the following:

- Technical DaySim software documentation,
- Technical TRANSIMS software documentation,
- Technical TRANSIMS Studio software documentation,
- Technical Python software documentation, and
- Technical integrated DaySim-TRANSIMS model system software documentation.

Detailed software and technical documentation can be found elsewhere and has been submitted as other stand-alone SHRP 2 C10A research project products. The start-up guide does not require expert-level experience with the software used in the model system. However, some familiarity with the software and, more important, the underlying concepts associated with these tools is essential.

CHAPTER 2

Model Directory Structure

This section of the start-up guide describes the model system directory structure and briefly explains the contents of each directory and subdirectory.

2.1 SHRP2_C10A_v3-12-21-2012_Base

When the zip-archive is unpacked, the main model directory is labeled as SHRP2_C10A_v3-12-21-2012_Base (Figure 2.1). This archive contains DaySim Version 1.8.4 and TRANSIMS Version 5. Older versions of the integrated model system used earlier releases of the DaySim and TRANSIMS software.

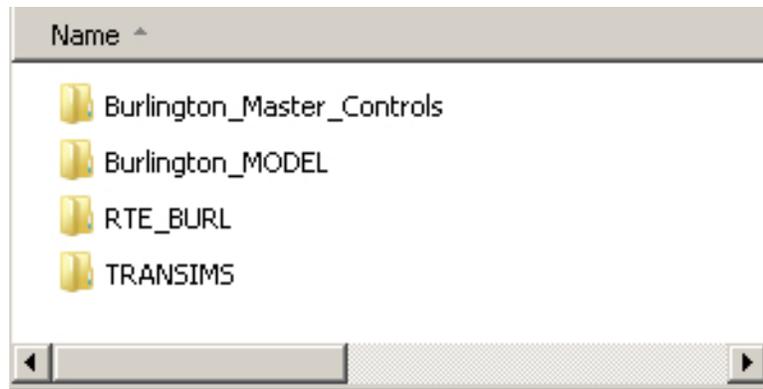


Figure 2.1. Main model directory: SHRP2_C10A_v3-12-21-2012_Base.

2.1.1 Burlington_Master_Controls

The *Burlington_Master_Controls* directory contains 70 control files (*.ctl) which specify the input and output files for each individual TRANSIMS tool used during the execution of the integrated model system (Figure 2.2). The control files can be opened with any text editor. The control files will look familiar to a practitioner familiar with the TRANSIMS software. The control files contain a listing of the “keys” used by the program followed by an input file, output file, parameter, and/or variable value.

Global (G) and assignment (N) iteration-specific control files are created on the fly during the model execution from these master control files. As such the user will find that most of the master control files contain “wildcards” that indicate where a variable replacement is performed. For instance, *results/@NEW@.@ALT@.Performance* in the master control file includes two variables (NEW and ALT)—identified by the @ symbols—which will be replaced by actual values during the execution. In this case NEW could be replaced by the value 1.50 and ALT could be replaced by the string *Activity_Model*.

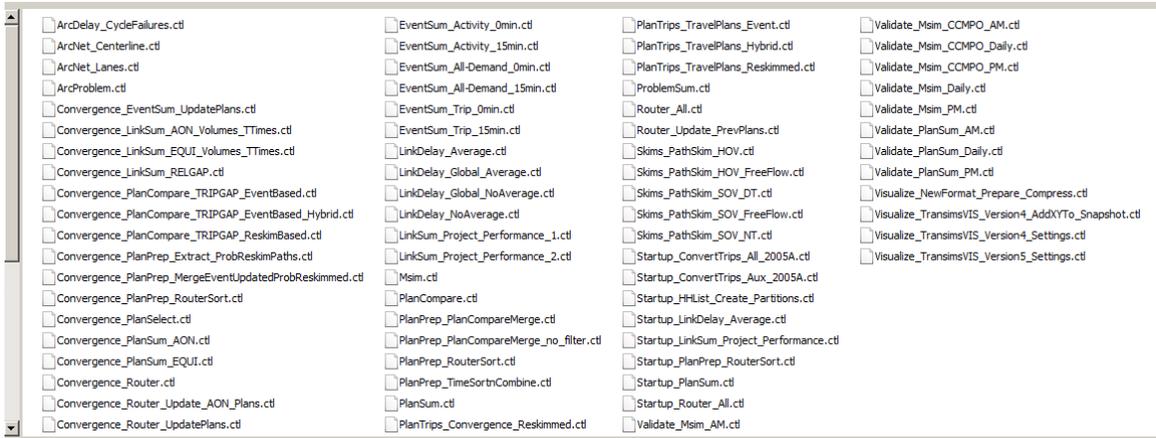


Figure 2.2. Burlington_Master_Controls directory.

2.1.2 Burlington_Model

The *Burlington_Model* directory is the principal directory associated with the integrated model system (Figure 2.3). This directory contains all the inputs, control files, and scripts used during the model execution and is the location where final outputs are written. The contents of this directory are described in more detail in Section 2.2.

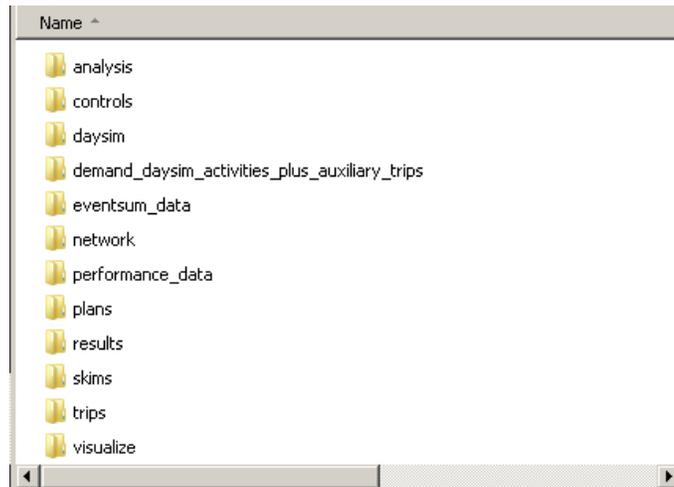


Figure 2.3. Burlington_Model directory.

2.1.3 RTE_BURL Directory

The runtime-environment (*RTE*) directory contains the Python scripts which control the integrated model system execution as well as model log files (Figure 2.4). The contents of this directory are described in more detail in Section 2.3.

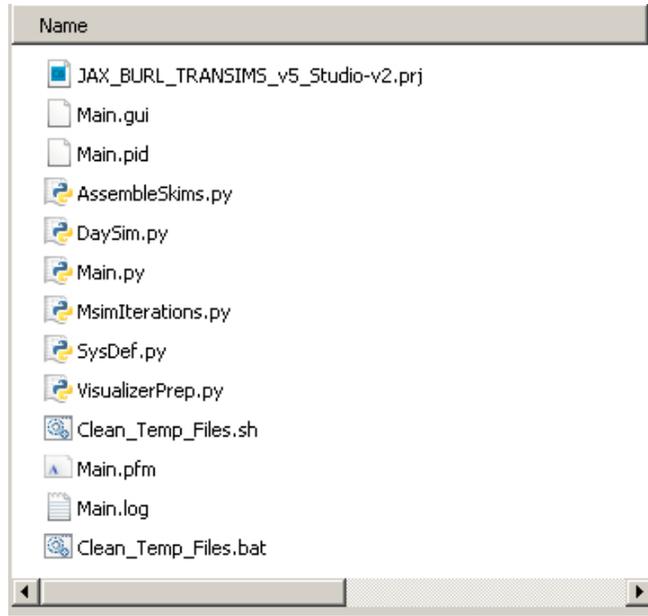


Figure 2.4. RTE_BURL directory.

2.1.4 TRANSIMS

The *TRANSIMS* directory contains the TRANSIMS executables (Figure 2.5). The contents of this directory are described in more detail in Section 2.4.



Figure 2.5. TRANSIMS directory.

2.2 Burlington_Model Directory

The *Burlington_Model* directory contains 12 individual subdirectories in which input, intermediate, and final output files from the integrated model system run are stored (Figure 2.6). Each individual subdirectory is described below.

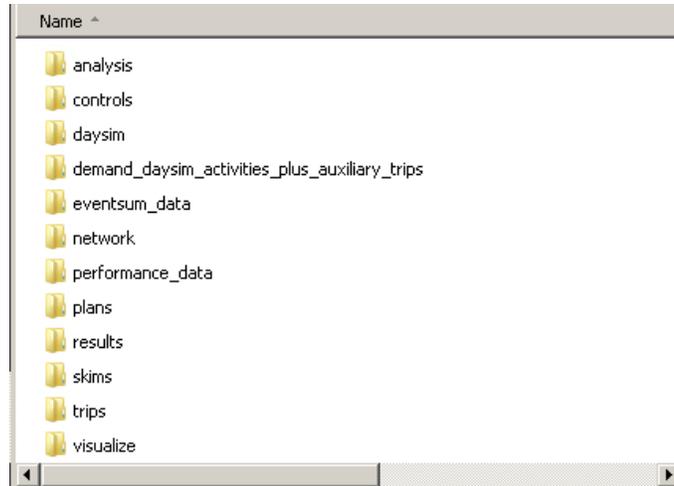


Figure 2.6. Burlington_Model directory.

2.2.1. Analysis Directory

The *analysis* directory contains subdirectories associated with model pre- and postprocessing. The *analysis* directory subdirectories are as follows:

- *arcview*\ contains shape file representations of the TRANSIMS network files.
- *eventsum*\ contains eventsum postprocessing outputs (*schedule consistency measures*).
- *eventsum_tod*\ contains eventsum by time-of-day postprocessing outputs (*schedule consistency measures*).
- *gap_extraction*\ contains trip gap and relative gap postprocessing outputs.
- *RMSD_demand*\ contains root mean square difference postprocessing outputs.

2.2.2 Controls Directory

The *controls* directory contains the run and TRANSIMS tool-specific control files which are generated on the fly during the model execution from the inputs contained in the *Master_Controls* directory.

2.2.3 DaySim Directory

The *daysim* directory contains the files, parameters, and controls required to run the disaggregate demand estimation model, DaySim. The *daysim* directory contains three subdirectories: *input*\, *output*\, and *working*\. For more information and technical documentation details, the user should review the DaySim online documentation.

2.2.4 Demand_daysim_activities_plus_auxiliary_trips Directory

The *demand_daysim_activities_plus_auxiliary_trips* directory contains the disaggregate demand estimate by DaySim as well as the auxiliary demand files (*trucks*, *externals*). The

directory contains three subdirectories: *demand*, *households*, and *vehicles*, which is how TRANSIMS inputs are typically split for input into the Router, Microsimulator, and related toolbox tools.

Important demand files include the following:

- *\Daysim\demand\activity.tab* is the daily regional household-person activity list estimated by DaySim.
- *\Daysim\demand\auxiliary_trips* is the daily regional truck and external trips not estimated by DaySim.
- *\Daysim\household\partitioned_all_households.t*** is the regional household list which is partitioned during model execution to support parallelization of the model run.

2.2.5 Eventsum_data Directory

The *eventsum_data* directory is the location where outputs from the TRANSIMS tool EventSum are written.

2.2.6 Network Directory

The *network* directory is the location where the TRANSIMS input network files are located. A TRANSIMS network comprises a number of important input files: *Activity Locations*, *Nodes*, *Links*, *Lanes*, *Lane_Connectivity*, *Pocket_Lane*, *Parking_Locations*, *Process_Link*, *Signal_Coordinator*, *Signalized_Nodes*, *Unsignalized_Nodes*, *Timing_Plan*, *Phasing_Plan*, and *Zone_Table*.

The user should refer to the TRANSIMS online documentation for more information about network files.

2.2.7 Performance_data Directory

The *performance* directory contains the link performance output file produced by the TRANSIMS Microsimulator. The performance file includes link-level attributes such as volumes and congested travel times.

2.2.8 Plans Directory

The *plans* directory contains the travel plan output files produced by the TRANSIMS Router. The plans represent all of the regional time-dependent shortest paths (TDSP) identified by the Router for all the input regional travel demand, the daily activities from DaySim, and the auxiliary truck and external trips estimated exogenously.

2.2.9 Results Directory

The *results* directory contains the *Link_Delay* output files produced by the TRANSIMS PlanSum and Microsimulator tools. The link delay files contain key metrics used during the assignment iteration process and used to assess model system performance.

2.2.10 Skims Directory

The *skims* directory contains the zone-to-zone travel impedance skims produced by the TRANSIMS PathSkim tool. Note that PathSkim is the only TRANSIMS Version 5 tool used in the integrated model system. The skims are a critical input to the DaySim demand estimation model.

2.2.11 Trips Directory

The *trips* directory contains the rescheduled, regional-demand trip files. Rescheduling sensitivity testing was conducted as part of ongoing research efforts. The directory will not be filled with output files when the model package is executed as delivered.

2.2.12 Visualize Directory

The *visualize* directory contains the output files associated with running the *VisualizerPrep.py* script. The output files produced by the TRANSIMS tool ArcSnapshot reside in this directory location. The user should refer to both the TRANSIMS and TRANSIMS visualizer (VIS) online documentation for more information.

2.3 RTE Directory

The *RTE* directory contains the Python scripts that control the integrated model system execution as well as model log files (Figure 2.7).

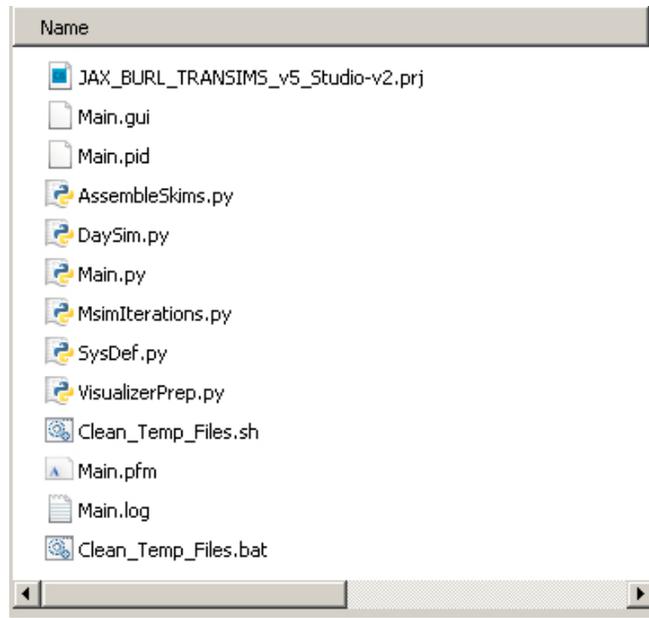


Figure 2.7. RTE directory.

2.3.1. Python (*.py) Files

The *RTE* directory contains the seven Python scripts that control the integrated model system execution:

- *AssembleSkims.py* generates the travel impedance skims which are critical inputs to the DaySim demand model.
- *DaySim.py* runs the DaySim demand model in addition to some pre- and postprocessing associated with disaggregate demand estimation.
- *Main.py* controls the step-wise execution of the entire model system—specifically, which tools are executed and when. The script contains important model run specification variables such as the number of global iteration loops (*MaxFeedBackLoops*) and assignment iteration loops (*MaxAssignmentIterations*).
- *MsimIterations.py* runs the supply-side iteration scheme that has been developed to achieve a dynamic traffic assignment user equilibrium using the TRANSIMS toolbox with input disaggregate demand estimated by DaySim.
- *SysDef.py* specifies a number of important system definitions. The relative paths to software binaries, critical input files, geography-specific inputs, and system control variables are specified.
- *VisualizerPrep.py* runs the visualizer preparation process so that regional simulations can be viewed using TRANSIMSVIS, the TRANSIMS visualizer software packaged with TRANSIMS Studio.

2.3.2 Main (Main.*) Files

The *RTE* directory also contains the 10 text files created during the model execution. Two of the *Main.** files of particular interest to end users are the following:

- *Main.log* is the main log file which documents the status of the model run. This can be a useful file for debugging model execution failures.
- *Main.res* is the main “resume” file. Depending on how it is configured, the integrated model system may have relatively long runtimes. Therefore, functionality was added to support resuming the model at the point desired by the end user. The model can be resumed from the last model step which was executed successfully or from a user-specified step (assuming all the intermediate files needed to launch from that point are present.) When the model is executed for the very first time, this file must be removed or renamed.

2.4 TRANSIMS

2.4.1 TRANSIMS_Studio

The *Transims_Studio* directory contains the installer for the TRANSIMS Studio v0.9.9 software (TRANSIMS-Studio-0.9.9-Win.exe). The TRANSIMS Studio application is an integrated development environment for the Transportation Analysis and Simulation System (TRANSIMS). Components include a runtime environment to execute TRANSIMS in parallel, as well as a full-featured graphical user interface (GUI). TRANSIMS Studio serves as the runtime controller and GUI for the model system. This software will need to be installed by the user. Note that the installer will also install the required version of Python.

2.4.2 TRANSIMS_v4_Windows_Executables_64bit

This directory contains the Version 4 TRANSIMS toolbox—specifically, the 22 Windows 64-bit compiled TRANSIMS tools which are used by the model system. Again, note that each individual TRANSIMS executable has an associated version number (e.g., Microsimulator, Version 4.0.79).

2.4.3 TRANSIMS_v5_Windows_Executables_64bit

This directory contains the Version 5 TRANSIMS toolbox—specifically, the 17 Windows 64-bit compiled TRANSIMS tools which are used by the model system. Note that each individual TRANSIMS executable has an associated version number (e.g., Microsimulator, Version 4.0.79).

2.4.4 TRANSIMVIS_Visualizer

This directory contains the files to support the use of the TRANSIMS-VIS visualizer.

CHAPTER 3

Operating Systems and Hardware

The current model system is configured to run on Windows operating systems, although it can also be set up to run in Linux. Runtimes for the model system are influenced by a number of factors, including

- The computing resources available to run the simulation (such as the number and performance of processors, the amount of memory available, and the speed of storage drive input/output);
- The degree of convergence required for a given model application;
- The size of the synthetic population used as the basis for all choice making in the model; and
- The level of detail of segmentation employed in the model system (such as the number of time periods).

The Windows implementation of the model system typically is configured to use between eight and 32 processing cores, although this can be flexibly set to exploit TRANSIMS partitioning capabilities. The project team tested a number of alternative configurations of the model system with respect to temporal detail and market segmentation of the skims. As few as four time periods and as many as 22 time periods were used in the model system; some configurations employed no value-of-time segmentation while other configurations included up to 50 value-of-time segments. The simplest of these schemes could be run with only 2 GB of RAM, while the most elaborate of the schemes required approximately 20 GB of RAM. Data storage and access influence runtimes; so in addition to having sufficient RAM, the model system (specifically, TRANSIMS) also benefits greatly from the use of fast hard drives for storage—ideally, solid state drives (SSD). The project team tested the model system using both the Transportation Research and Analysis Computing Center (TRACC) Linux computing cluster located at Argonne National Laboratory and local Windows-based servers.

3.1 Specifying Operating System

The integrated model system can be operating-system agnostic, although the current distribution is configured for Windows only. The specification of the operating system can be configured by the user by reviewing and/or modifying lines in system definition Python script, *SysDef.py*. Lines 14 through 38 of the *SysDef.py* file are presented in Figure 3.1.

The operating system will be automatically identified using the `os.name` function. The model package as delivered is configured to run in Windows and the software paths are specified properly (`var.BINDIR`, `var.BINDIR_v5`, `var.DAYSIM.EXE`, and `var.DAYSIM_CTL`).

If the user wishes to run the integrated model system in Linux, a number of paths in this section of the system definition Python file (Lines 26–31) need to be modified. Specifically, the lines colored in green will need to be revised to point to the correct paths for the Linux TRANSIMS binaries and the location of the TRANSIMS RTE where the Studio installation places the *TRANSIMS RTE* directory. In addition, the Windows-compiled version of the DaySim executable will need to be run through the WINE emulator.

```

14  #== BEGIN - Operating System Dependent Paths ==#
15  if os.name == 'nt':  # Windows Operating System
16
17      sys.path.append ('../../TransimsRTE')
18      from TransimsRTE import *
19      var.BINDIR                = '../../TRANSIMS_Software/TRANSIMS_v4_Windows_Executables_64bit'
20      var.BINDIR_v5             = '../../TRANSIMS_Software/TRANSIMS_v5_Windows_Executables_64bit'
21      var.DAYSIM_EXE            = 'DAYSIM09v16_BURL22.exe'
22      var.DAYSIM_CTL            = 'daysim_22periods.ctf'
23
24  elif os.name == 'posix':  # LINUX Operating System
25
26      sys.path.append ('/home/ac.ereztz/Custom_Software/TRANSIMS-Studio-0.9.9-Python/TransimsRTE')
27      from TransimsRTE import *
28      var.BINDIR                = '/soft/transims/alpha/bin'
29      var.BINDIR_v5             = '/home/ac.ereztz/Custom_Software/Transims50/BinBoost/BinBoost'
30      var.DAYSIM_EXE            = '/soft/transims/wine/bin/wine_DAYSIM09v16_BURL22.exe'
31      var.DAYSIM_CTL            = 'daysim_22periods.ctf'
32
33  if (os.path.exists(var.BINDIR + '/Router.exe') or os.path.exists(var.BINDIR + '/Router')) == False:
34
35      Event ('TRANSIMS Executables Not Found! Check Paths and Restart. EXECUTION TERMINATED!')
36      sys.exit(1)
37
38  #== END - Operating System Dependent Paths ==#

```

Figure 3.1. SysDef.py lines for operating system specification.

3.2 Specifying Partitioning Variables

A number of tools in the TRANSIMS toolbox support partitioning to reduce model system runtime. In particular, the TRANSIMS Router can be run with a partitioned regional household list to expedite the processing using the computing resources available on the modeling machine being used.

The number of partitions can be selected and set by the user on Line 64 of the system definition, *SysDef.py*, Python script. The model package as delivered has the *NUM_PARTITIONS* variable set to a value of eight (8). Therefore, the input household list will be split into eight parts to facilitate eight simultaneous Router instances, each working on a separate portion of the input demand files (activities and trips). Note that *NUM_PATHSKIM_THREADS* should not be set higher than eight, as testing to date has revealed instability with the program beyond eight threads. Thus,

- (Line 64) `var.NUM_PARTITIONS = 8`, and
- (Line 65) `var.NUM_PATHSKIM_THREADS = 8`.

CHAPTER 4

Running the Integrated Model System

This section provides a list of instructions to run the integrated model system. The model system requires at least 4 GB of RAM, although if additional spatial or temporal detail is required this requirement will increase. The model does not explicitly require multiple processing cores, though the availability and speed of multiple cores will reduce model runtimes. The steps should be performed in the order they are presented.

1. Unzip the compressed archive SHRP2_C10A_v3-12-21-2012_Base.zip. The model files can be unpacked anywhere on the user's local machine (e.g., C:\Travel_Models).
2. Run the TRANSIMS Studio installer provided in the directory ..\SHRP2_C10A_v3-12-21-2012_Base\TRANSIMS\TRANSIMS_Studio\TRANSIMS-Studio-0.9.9-Win.exe. Simply double-click on the executable, click "run," and follow the installation instructions (Figure 4.1).

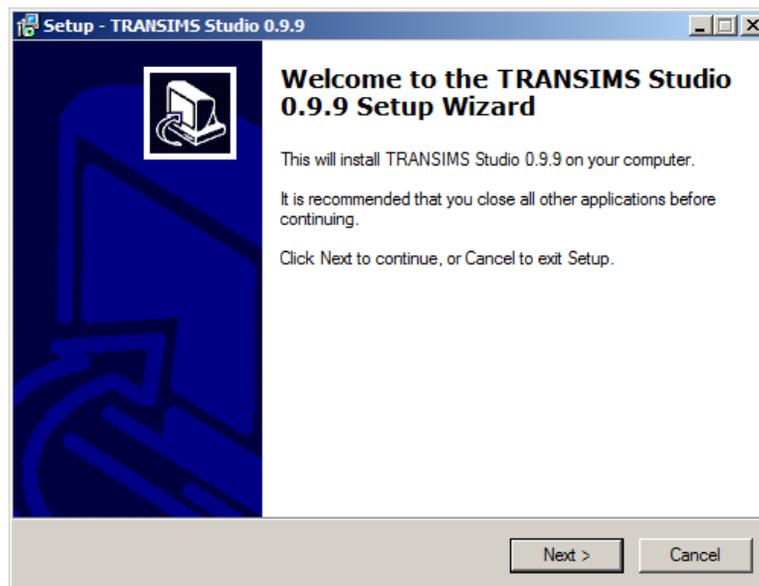


Figure 4.1. TRANSIMS Studio 0.9.9 setup wizard.

3. Locate the TRANSIMS Studio executable. Note that a desktop shortcut icon may have been created during the installation. Double-click the shortcut icon to launch the program (Figure 4.2).

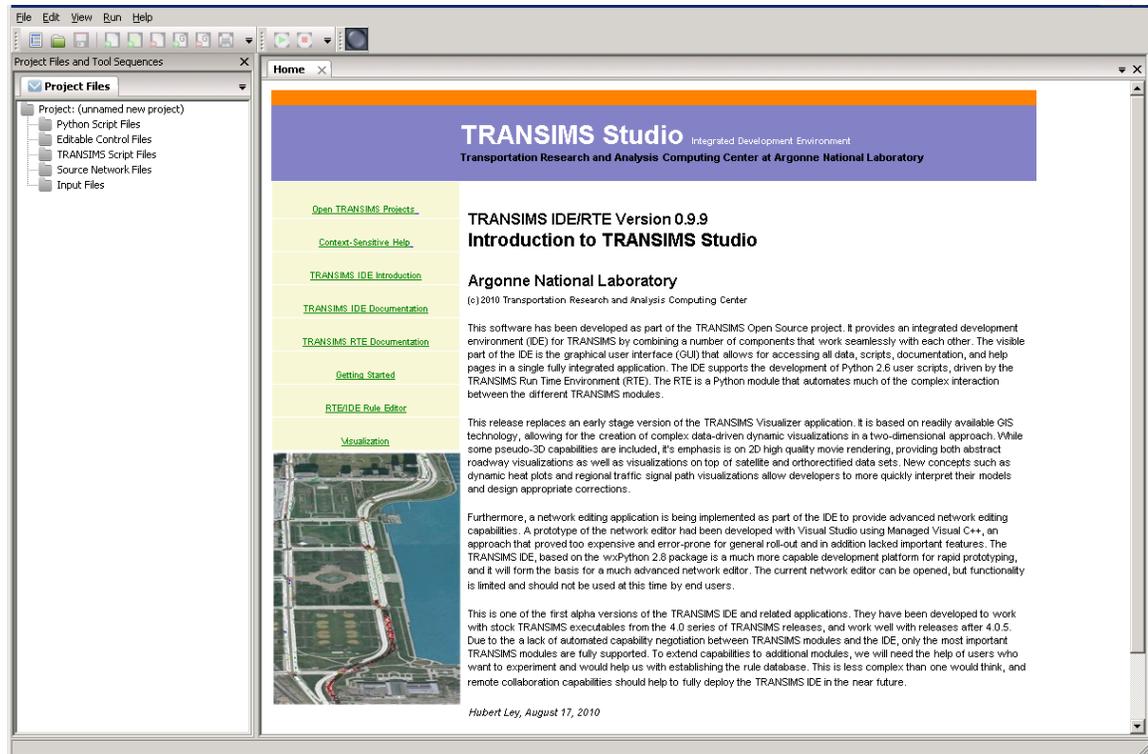


Figure 4.2. TRANSIMS Studio.

4. Locate the TRANSIMS Studio project file (.prj) and open the file. Select File → Open and select the file located at *..\SHRP2_C10A_v3-12-21-2012_Base\RTE_BURL\C10A_DaySim_TRANSIMS_v5_Studio.prj*.
5. When the *C10A_DaySim_TRANSIMS_v5_Studio.prj* file is opened, note that the files related to the project will have been added in the Project Files window which is positioned in the left-side panel of the GUI.
6. Click the plus sign (+) next to the Python Script Files to expand the list of the Python script files. At least six Python scripts should be displayed: *AssembleSkims.py*, *DaySim.py*, *Main.py*, *MsimIterations.py*, *SysDef.py*, and *VisualizerPrep.py* (Figure 4.3).

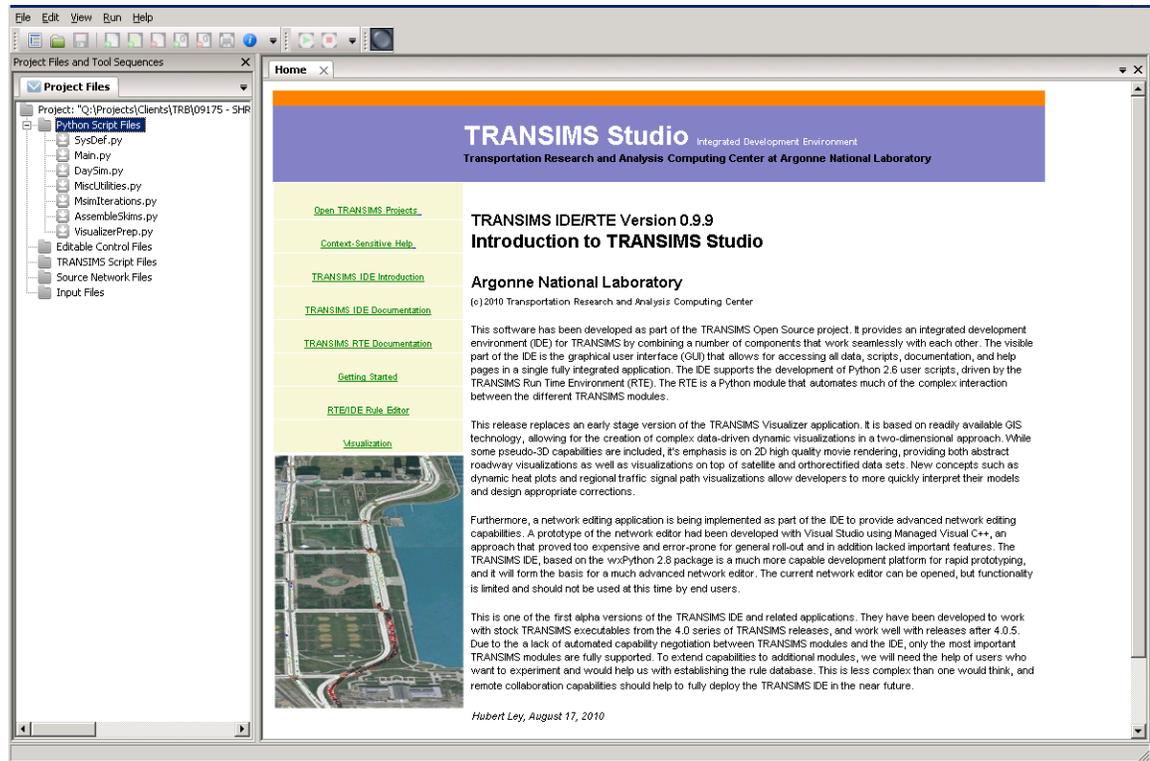


Figure 4.3. TRANSIMS Studio with project files open (*.prj).

7. Select the *Main.py* script in the list of Python Script Files by clicking the file name once to highlight the *Main.py* file. Then double-click the *Main.py* file.
8. A new tab containing the contents of the *Main.py* script file will be opened in the main GUI window (Figure 4.4).

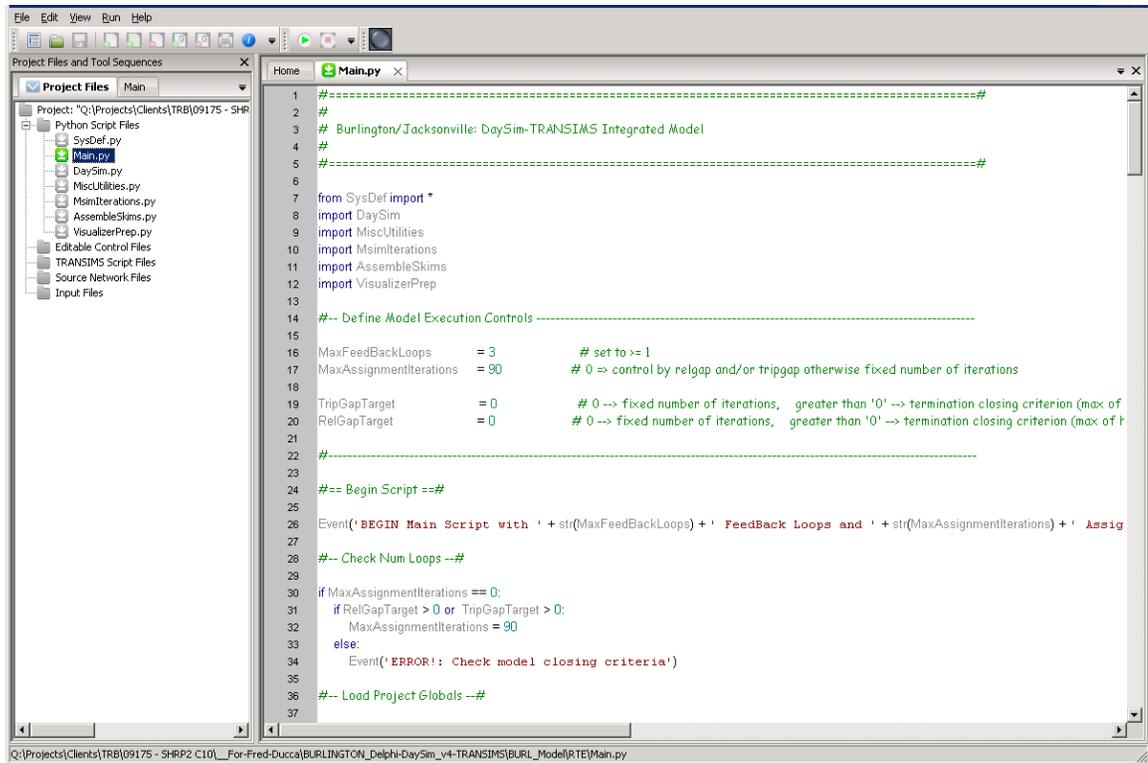


Figure 4.4. TRANSIMS Studio with Main.py displayed in GUI.

9. Define the model execution controls. The integrated model system can be run one of two ways: (1) using a fixed number of global (G) iterations with a fixed number of assignment (N) iterations, or (2) using relative gap and trip-gap convergence closure criteria.
10. To run the integrated model system with a fixed number of iterations, the user must set two variables:
 - a. *MaxFeedBackLoops*—the number of global feedback iterations, and
 - b. *MaxAssignmentIterations*—the number of assignment iterations performed within each global iteration.

The integrated model was typically run with $\text{MaxFeedBackLoops} = 3$ ($G = 3$) and $\text{MaxAssignmentIterations} = 90$ ($N = 90$). For initial testing by a new user it is recommended that the number of iterations be reduced to first achieve a successful complete run (e.g., $G = 1$ and $N = 15$).
11. To run the integrated model system with a gap-based quit closure criteria, the user must set four variables:
 - a. *MaxFeedBackLoops*—the number of global feedback iterations.
 - b. *MaxAssignmentIterations*—the number of assignment iterations performed within each global iteration. This variable should be set to zero ($N = 0$).

- c. *TripGapTarget*—the assignment process will terminate when the maximum hourly trip-gap value is less than the *TripGapTarget*.
- d. *RelGapTarget*—the assignment process will terminate when the maximum hourly relative-gap value is less than the *RelGapTarget*.

The integrated model was typically run with a fixed number of assignment iterations; therefore, the gap-based quit closure criteria were not often used once this functionality was configured. Some user-sensitivity testing will be required to select appropriate values for *TripGapTarget* and *RelGapTarget*, though values of 0.01 could be appropriate. New users are recommended to specify a fixed number of assignment iterations before attempting to use the gap-based quit closure criteria variables.

12. Remove or rename the *Main.res* file, which resumes the integrated model run from an intermediate step. The *Main.res file* is located at `..\RTE_BURL \Main.res`. A new resume file will be generated every time the model is executed, so this file must be removed as part of the model run preflight checklist if the user desires the model to start from the first step in the model flow.
13. To run the model, make sure the *Main.py* script has been selected and is displayed in the Studio GUI window. Double-check the model execution controls (iteration values specified on Line 16 and Line 17 of the *Main.py* script).
14. To initiate the run, click the green “play” arrow on the GUI menu bar

(Figure 4.5). Click  to start the run, and click  to stop the run.

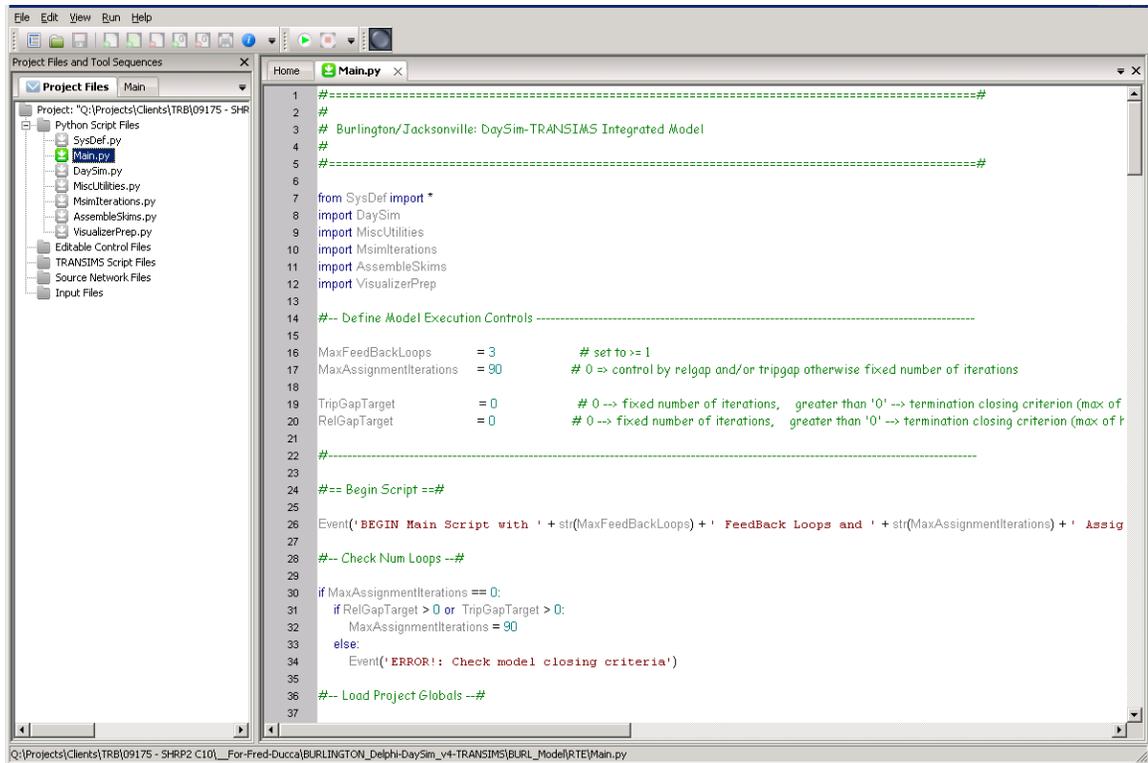


Figure 4.5. Run model by clicking green “play” button.

4.1 Model Runtimes

The project team extensively tested various iterative schemes to evaluate trade-offs between network assignment and model system convergence and overall model system runtimes. Generally, greater degrees of convergence are required for more spatially, temporally, or behaviorally detailed analysis. However, the level of convergence required for any given analysis is context specific, and the user is afforded many potential means with which to configure the model system and investigate performance and runtime trade-offs. Keys and parameters which affect DaySim and TRANSIMS runtimes can be easily configured by the user via two primary control files.

Model runtimes are highly influenced by the computing resources (both the memory and the number of processing cores). Runtimes in excess of 6 days were observed to run the full integrated model system with three global iterations and 90 assignment iterations using 16 processing cores. As described earlier, the user is encouraged to specify fewer global and assignment iterations to complete a successful run of the full integrated system. Local runtimes can then be estimated by scaling the observed runtime accordingly if additional iterations are desired.

4.2 File Naming Convention

Output files generated by the integrated model use a prefix in the file name to identify the global iteration, assignment iteration, and model step for which the file was generated. For

example, the file *2.50.Activity_Model.Performance* is the Microsimulator performance output file produced at iteration $G = 2$ and $N = 50$ for the activity model. The wildcards *@NEW@*.*@ALT@.Performance* file specified in the master control file were replaced with $NEW = 2.50$ and $ALT = Activity_Model$.

Note that the integrated model system can also use a demand trip file developed from the Burlington regional four-step model. The four-step model demand can be used instead of the activity list estimated by DaySim by modifying a variable in the system definition *SysDef.py* Python script file. Thus,

(Line 48) `var.ALT = 'Activity_Model'`

can be adjusted to

(Line 48) `var.ALT = 'Trip_Model'`.

With this change, the integrated model system will use the trip demand converted from the four-step model. Output files will have file names with a different prefix, for instance *2.50.Trip_Model.Performance*. As stated earlier, the trip model was only used during the early stages of the research to help guide the development and configuration of the supply-side model while the DaySim model was still under development.

PART 2: Jacksonville, Florida

CHAPTER 1

Introduction

The primary objective of the C10A project is to make operational a dynamic, integrated model—an integrated, advanced travel-demand model with a fine-grained, time-dependent network—and to demonstrate the model’s performance through validation tests and policy analyses. This integrated model system is necessary because most current travel models are not sufficiently sensitive to the dynamic interplay between travel behavior and network conditions and are unable to reasonably represent the effects of transportation policies, such as variable road pricing and travel demand management strategies. The model system was designed to capture changes in demand, such as time of day choice (i.e., peak spreading), destination, mode, and route choice, in response to capacity and operational improvements such as signal coordination, freeway management, and variable tolls.

The purpose of this start-up guide is to provide information necessary for an end user to develop a working knowledge of the DaySim-TRANSIMS Integrated Model System (“the model”). The start-up guide is a technical document for travel modeling practitioners who are interested in studying the configuration and specification of the system and ultimately running the model.

The end-user start-up guide does *not* represent and/or address the following:

- Technical DaySim software documentation;
- Technical Transportation Analysis Simulation System (TRANSIMS) software documentation;
- Technical TRANSIMS Studio software documentation;
- Technical Python software documentation; and
- Technical Integrated DaySim-TRANSIMS Model System software documentation.

Detailed software and technical documentation can be found elsewhere and has been submitted as other stand-alone SHRP 2 C10A research project products. The end-user start-up guide does not require expert-level experience with the software utilized in the model system. However, some familiarity with the software and, more importantly, the underlying concepts associated with these tools is essential.

CHAPTER 2

Model Directory Structure

This section of the start-up guide describes the model system directory structure and briefly explains the contents of each directory and subdirectory.

2.1 Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v1

When the zip-archive is unpacked, the main model directory (Figure 2.1) is labeled as *Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v1*. This archive contains DaySim version 1.8.3 and TRANSIMS version 5. Older versions of the integrated model system used earlier releases of the DaySim and TRANSIMS software, which are also included in the archive.

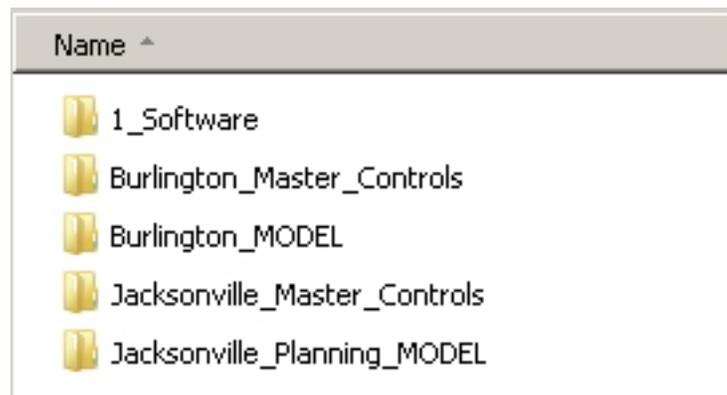


Figure 2.1. Main model directory: SHRP2_C10A_v3-12-21-2012_Base.

2.1.1 Burlington_Master_Controls and Jacksonville_Master_Controls

The *Burlington_Master_Controls* and *Jacksonville_Master_Controls* directories (Figure 2.2) contain 62 control files (*.ctl) which specify the input and output files for each individual TRANSIMS tool used during the execution of the integrated model system. The control files can be opened with any text editor. The control files will look familiar to a practitioner used to working with the TRANSIMS software. The control files contain a listing of the keys used by the program, followed by an input file, output file, parameter, and/or variable value.

Global (G) and Assignment (N) iteration-specific control files are created on the fly during the model execution from these master control files. As such, the user will find that most of the master control files contain wildcards that indicate where a variable replacement is performed. For instance, *results/@NEW@.@ALT@.Performance* in the master control file includes two variables (NEW and ALT) identified by the @ symbols, which will be replaced by actual values during the execution. In this case NEW could be replaced by the value 1.50 and ALT could be replaced by the string *Activity_Model*.

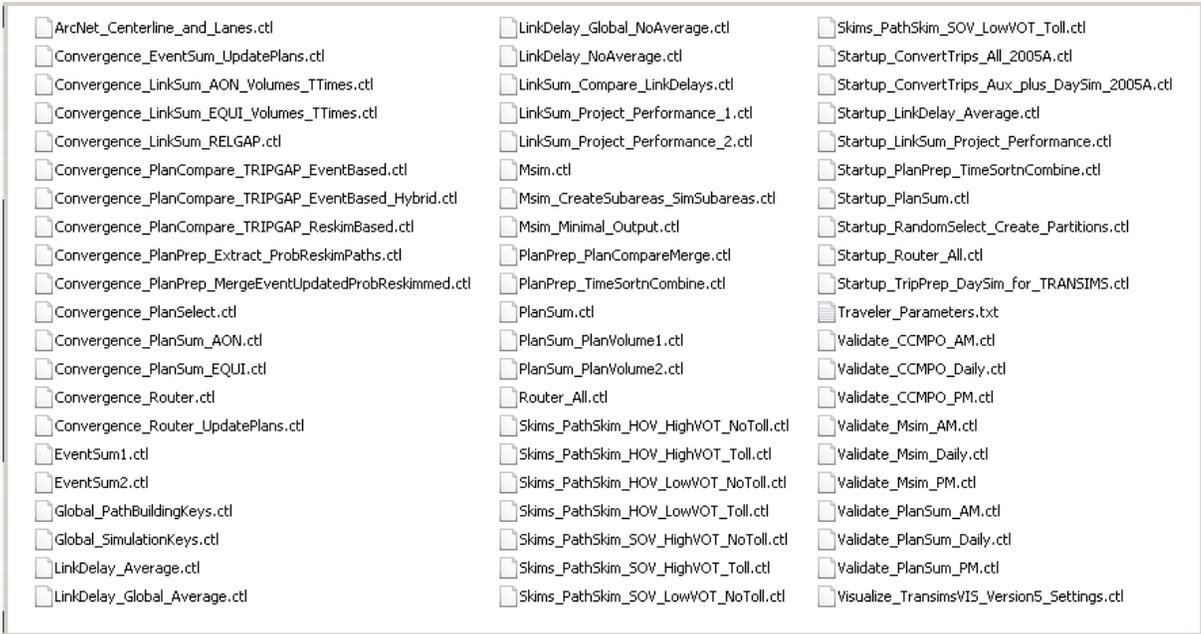


Figure 2.2. Burlington_Master_Controls and Jacksonville_Master_Control directories.

2.1.2 Burlington_Model and Jacksonville_Planning_Model

The *Burlington_Model* and *Jacksonville_Planning_Model* directories (Figure 2.3) are the principal directories associated with the integrated model system. These directories contain all the inputs, control files, and scripts used during the model execution and is the location where final outputs are written. The contents of this directory are described in more detail in Section 2.2.

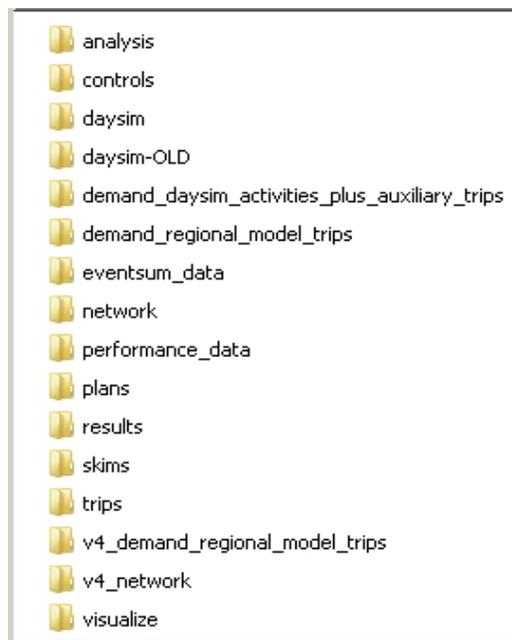


Figure 2.3. Burlington_Model and Jacksonville_Planning_Model directories.

2.1.3 RTE_BURL and RTE_JAX Directories

The *RTE* (runtime environment) directory contains the Python scripts that control the integrated model system execution as well as model log files (Figure 2.4). The contents of this directory are described in more detail in Section 2.3.

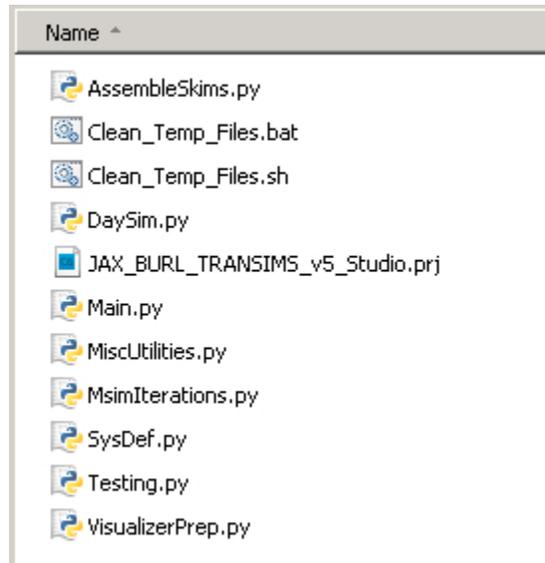


Figure 2.4. RTE_BURL and RTE_JAX directories.

2.1.4 1_Software

The *1_Software* directory contains the TRANSIMS executables (Figure 2.5). The contents of this directory are described in more detail in Section 2.4.

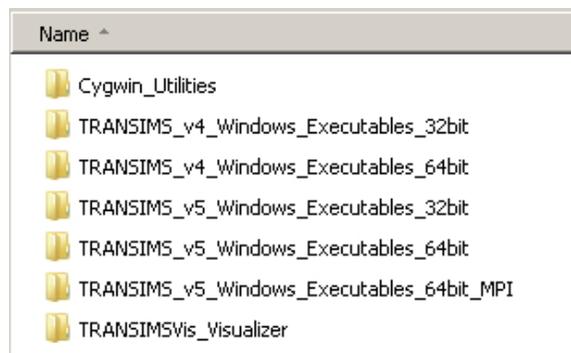


Figure 2.5. 1_Software directory

2.2 Burlington_Model and Jacksonville_Planning_Model Directories

The *Burlington_Model* and *Jacksonville_Planning_Model* directories contain 16 individual subdirectories where input, intermediate, and final output files from the integrated model system run are stored (Figure 2.6). Each individual subdirectory is described below.

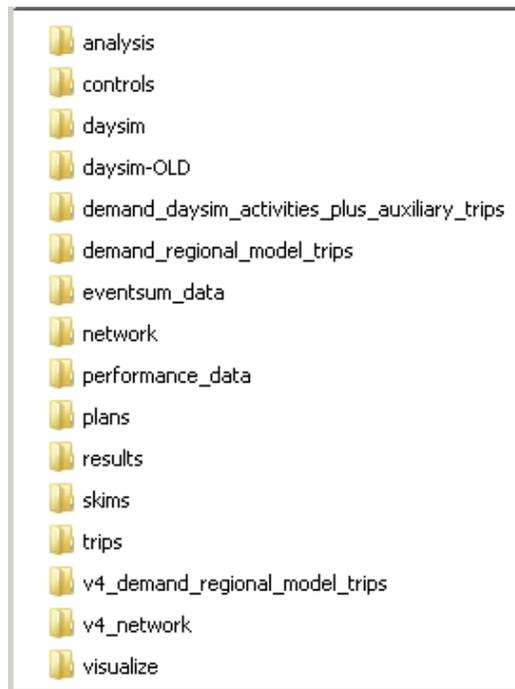


Figure 2.6. Burlington_Model and Jacksonville_Planning_Model Directories

2.2.1 Analysis Directory

The *analysis* directory contains subdirectories associated with model pre- and postprocessing. The *analysis* directory contains these subdirectories:

- *arcview*\ contains shapefile representations of the TRANSIMS network files.
- *eventsum*\ contains eventsum postprocessing outputs (*schedule consistency measures*).
- *eventsum_tod*\ contains eventsum by time of day postprocessing outputs (*schedule consistency measures*).
- *gap_extraction*\ contains trip gap and relative gap postprocessing outputs.
- *RMSD_demand*\ contains root mean square difference postprocessing outputs.

2.2.2 Controls Directory

The *controls* directory contains the run and TRANSIMS tool-specific control files, which are generated on the fly during the model execution from the inputs contained in the *Master_Controls* directory.

2.2.3 DaySim Directory

The *daysim* directory contains the files, parameters, and controls required to run the disaggregate demand estimation model, DaySim. The *daysim* directory contains four

subdirectories: `\Daysim_1_8_3`, `\output`, `\skims`, and `\working`. For more information and technical documentation details, please review the DaySim documentation.

2.2.4 DaySim-OLD Directory

The *daysim-OLD* directory contains the files, parameters, and controls required to run the older version of the disaggregate demand estimation model, DaySim. The *daysim-OLD* directory contains four subdirectories: *input*\, *output*\, *working*\, and *PopGen*. For more information and technical documentation details, please review the DaySim documentation.

2.2.5 Demand_daysim_activities_plus_auxiliary_trips Directory

The *demand_daysim_activities_plus_auxiliary_trips* directory contains the disaggregate demand estimate by DaySim, as well as the auxiliary demand files (*trucks*, *externals*). The directory contains three subdirectories: *demand*\, *households*\, and *vehicles*\, which is how TRANSIMS inputs are typically split for input into the router, Microsimulator, and related toolbox tools.

Important demand files include

- `\Daysim\demand\activity.tab`: the daily regional household-person activity list estimated by DaySim;
- `\Daysim\demand\auxiliary_trips`: the daily regional truck and external trips not estimated by DaySim; and
- `\Daysim\household\partitioned_all_households.t**`: the regional household list which is partitioned during model execution to support parallelization of the model run.

2.2.6 Demand_regional_model_trips Directory

The *demand_regional_model_trips* directory contains the original regional trip-based model demand, including the auxiliary demand files (*trucks*, *externals*). The directory contains three subdirectories: *demand*\, *households*\, and *vehicles*\, which is how TRANSIMS inputs are typically split for input into the router, Microsimulator, and related toolbox tools.

2.2.7 Eventsum_data Directory

The *eventsum_data* directory is the location where outputs from the TRANSIMS tool EventSum are written.

2.2.8 Network Directory

The *network* directory is the location where the TRANSIMS input network files are located. A TRANSIMS network is composed of a number of important input files: *Activity Locations*, *Nodes*, *Links*, *Lanes*, *Lane_Connectivity*, *Pocket_Lane*, *Parking_Locations*, *Process_Link*, *Signal_Coordinator*, *Signalized_Nodes*, *Unsignalized_Nodes*, *Timing_Plan*, *Phasing_Plan*, and *Zone_Table*.

The user should refer to the TRANSIMS online documentation for more information about network files.

2.2.9 Performance_data Directory

The *performance* directory contains the link performance output file produced by the TRANSIMS Microsimulator. The *performance* file includes link-level attributes such as volumes and congested travel times.

2.2.10 Plans Directory

The *plans* directory contains the travel plan output files produced by the TRANSIMS router. The plans represent all of the regional time-dependent shortest paths (TDSP) identified by the router for all the input regional travel demand, the daily activities from DaySim, and the auxiliary truck and external trips estimated exogenously.

2.2.11 Results Directory

The *results* directory contains the *Link_Delay* output files produced by the TRANSIMS PlanSum and Microsimulator tools. The link delay files contain key metrics utilized during the assignment iteration process and used to assess model system performance.

2.2.12 Skims Directory

The *skims* directory contains the zone-to-zone travel impedance skims produced by the TRANSIMS PathSkim tool. Note that PathSkim is the only TRANSIMS Version 5 tool utilized in the integrated model system. The skims are a critical input to the DaySim demand estimation model.

2.2.13 Trips Directory

The *trips* directory contains the rescheduled regional demand trip files. Rescheduling sensitivity testing was conducted as part of ongoing research efforts. The directory will not be filled with output files when the model package is executed as delivered.

2.2.14 V4_demand_regional_model_trips Directory

The *v4_demand_regional_model_trips* directory contains the demand from the regional trip-based model in TRANSIMS v4 format.

2.2.15 V4_network Directory

The *v4_network* directory contains the original TRANSIMS v4 networks developed as part of the project, which were succeeded by the TRANSIMS v5 networks.

2.2.16 Visualize Directory

The *visualize* directory contains the output files associated with running the *VisualizerPrep.py* script. The output files produced by the TRANSIMS tool ArcSnapshot reside in this directory location. The user should refer to both the TRANSIMS and TRANSIMS-VIS online documentation for more information.

2.3 RTE_BURL and RT_JAX Directories

The *RTE* (runtime environment) directory (Figure 2.7) contains the Python scripts that control the integrated model system execution, as well as model log files.

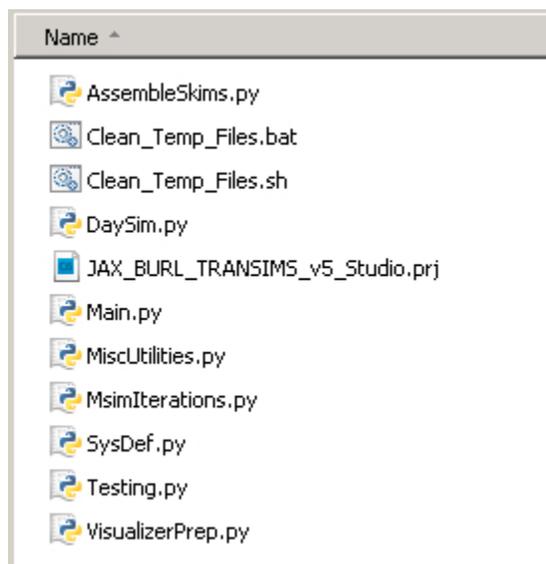


Figure 2.7. RTE directory.

2.3.1 Python (*.py) Files

The *RTE* (runtime environment) directory contains the seven Python scripts that control the integrated model system execution. Each Python script is described briefly below.

- *AssembleSkims.py* generates the travel impedance skims which are critical inputs to the DaySim demand model.
- *Daysim.py* runs the DaySim demand model in addition to some pre- and postprocessing associated with disaggregate demand estimation.
- *Main.py* controls the step-wise execution of the entire model system—specifically, which tools are executed and when. The script contains important model run specification variables, such as the number of global iteration loops (*MaxFeedBackLoops*) and assignment iteration loops (*MaxAssignmentIterations*).

- *MsimIterations.py* runs the supply-side iteration scheme that has been developed to achieve a dynamic traffic assignment user equilibrium using the TRANSIMS toolbox with input disaggregate demand estimated by DaySim.
- *SysDef.py* specifies a number of important system definitions. The relative paths to software binaries, critical input files, geography-specific inputs, and system control variables are specified.
- *VisualizerPrep.py* runs the visualizer preparation process, so that regional simulations can be viewed using TRANSIMS-VIS, the TRANSIMS visualizer software packaged with TRANSIMS Studio.

2.3.2 Main (Main.*) Files

The *RTE* (runtime environment) directory contains the 10 text files created during the model execution. Two of the *Main.** files of particular interest to end users are these:

- *Main.log* is the main logfile which documents the status of the model run. This can be a useful file for debugging model execution failures.
- *Main.res* is the main resume file. Depending on how it is configured, the integrated model system may have relatively long runtimes. Therefore, functionality was added to support resuming the model where desired by the end user. The model can be resumed from the last model step which was executed successfully or from a user-specified step (assuming all the intermediate files needed to launch from that point are present.) When the model is executed for the very first time, this file must be removed or renamed.

2.4 1_Software

2.4.1 TRANSIMS_Studio

The *Transims_Studio* directory contains the installer for the TRANSIMS Studio v0.9.9 software (*TRANSIMS-Studio-0.9.9-Win.exe*). The TRANSIMS Studio application is an integrated development environment for the Transportation Analysis and Simulation System (TRANSIMS). Components include a runtime environment to execute TRANSIMS in parallel, as well as a full-featured graphical user interface (GUI). TRANSIMS Studio serves as the runtime controller and graphical user interface for the model system. This software will need to be installed by the user. Note that the installer will also install the required version of Python.

2.4.2 TRANSIMS_v4_Windows_Executables_32bit

This directory contains the Version 4 TRANSIMS toolbox—specifically, the 22 Windows 32-bit compiled TRANSIMS tools which are utilized by the model system. Again, note that

each individual TRANSIMS executable has an associated version number (e.g., Microsimulator, Version 4.0.79)

2.4.3 TRANSIMS_v4_Windows_Executables_64bit

This directory contains the Version 4 TRANSIMS toolbox—specifically, the 22 Windows 64-bit compiled TRANSIMS tools, which are utilized by the model system. Again, note that each individual TRANSIMS executable has an associated version number (e.g., Microsimulator, Version 4.0.79).

2.4.4 TRANSIMS_v5_Windows_Executables_32bit

This directory contains the Version5 TRANSIMS toolbox—specifically, the 17 Windows 32-bit compiled TRANSIMS tools, which are utilized by the model system. Note that each individual TRANSIMS executable has an associated version number (e.g., Microsimulator, Version 4.0.79).

2.4.5 TRANSIMS_v5_Windows_Executables_64bit

This directory contains the Version5 TRANSIMS toolbox—specifically, the 17 Windows 64-bit compiled TRANSIMS tools, which are utilized by the model system. Note that each individual TRANSIMS executable has an associated version number (e.g. Microsimulator, Version 4.0.79).

2.4.6 TRANSIM-VIS_Visualizer

This directory contains the files to support the use of the TRANSIMS-VIS visualizer.

CHAPTER 3

Operating Systems and Hardware

The current model system is configured to run on Windows operating systems, although it can also be set up to run on Linux as well. Runtimes for the model system are influenced by a number of factors, including

- The computing resources available to run the simulation (such as the number and performance of processors, the amount of memory available, and the speed of storage drive input/output);
- The degree of convergence required for a given model application;
- The size of the synthetic population used as the basis for all choice-making in the model; and
- The level of detail of segmentation employed in the model system (such as the number of time periods).

The Windows implementation of the model system typically was configured to use between 8 and 32 processing cores, although this can be flexibly set to exploit the partitioning capabilities of TRANSIMS. The project team tested a number of alternative configurations of the model system with respect to temporal detail and market segmentation of the skims. As few as four time periods and as many as 22 time periods were used in the model system; while some configurations employed no value of time segmentation, other configurations included up to 50 value of time segments. The simplest of these schemes could be run with only 2 GB of RAM, while the most elaborate of the schemes required approximately 20 GB of RAM. Data storage and access influence runtimes, so in addition to having sufficient RAM, the model system (specifically, TRANSIMS) also benefits greatly from the use of fast hard drives for storage—ideally, solid state drives (SSD). The project tested the model system using the TRACC Linux computing cluster located at Argonne National Laboratory, as well as using local Windows-based servers.

3.1 Specifying Operating System

The integrated model system can be operating-system agnostic, although the current distribution is configured for Windows only. The specification of the operating system can be configured by the user by reviewing and/or modifying lines in system definition Python script, *SysDef.py*. Lines 14 through 38 of the *SysDef.py* file are presented in Figure 3.1.

The operating system will be automatically identified using the `os.name` function. The model package as delivered is configured to run in Windows, and the software paths are specified properly (`var.BINDIR`, `var.BINDIR_v5`, `var.DAYSIM.EXE`, and `var.DAYSIM_CTL`).

If the user wishes to run the integrated model system in Linux, a number of paths in this section of the system definition Python file (Lines 26–31) need to be modified. Specifically, the lines colored in green will need to be revised to point to the correct paths for the Linux TRANSIMS binaries and the location of the TRANSIMS RTE where the Studio installation places the *TRANSIMS RTE* directory. In addition, the Windows-compiled version of the DaySim executable will need to be run through the WINE emulator.

```

14  #== BEGIN - Operating System Dependent Paths ==#
15  if os.name == 'nt':  # Windows Operating System
16
17      sys.path.append ('../../TransimsRTE')
18      from TransimsRTE import *
19      var.BINDIR                = '../../TRANSIMS_Software/TRANSIMS_v4_Windows_Executables_64bit'
20      var.BINDIR_v5            = '../../TRANSIMS_Software/TRANSIMS_v5_Windows_Executables_64bit'
21      var.DAYSIM_EXE           = 'DAYSIM09v16_BURL22.exe'
22      var.DAYSIM_CTL           = 'daysim_22periods.ctf'
23
24  elif os.name == 'posix':  # LINUX Operating System
25
26      sys.path.append ('/home/ac.ereztz/Custom_Software/TRANSIMS-Studio-0.9.9-Python/TransimsRTE')
27      from TransimsRTE import *
28      var.BINDIR                = '/soft/transims/alpha/bin'
29      var.BINDIR_v5            = '/home/ac.ereztz/Custom_Software/Transims50/BinBoost/BinBoost'
30      var.DAYSIM_EXE           = '/soft/transims/wine/bin/wine_DAYSIM09v16_BURL22.exe'
31      var.DAYSIM_CTL           = 'daysim_22periods.ctf'
32
33  if (os.path.exists(var.BINDIR + '/Router.exe') or os.path.exists(var.BINDIR + '/Router')) == False:
34
35      Event ('TRANSIMS Executables Not Found! Check Paths and Restart. EXECUTION TERMINATED!')
36      sys.exit(1)
37
38  #== END - Operating System Dependent Paths ==#

```

Figure 3.1. SysDef.py lines for operating system specification.

3.2 Specifying Partitioning Variables

A number of tools in the TRANSIMS toolbox support partitioning in order to reduce model system runtime. In particular, the TRANSIMS router can be run with a partitioned regional household list to expedite the processing, using the computing resources available on the modeling machine being used.

The number of partitions can be selected and set by the user on Line 64 of the system definition *SysDef.py Python* script. The model package as delivered has the *NUM_PARTITIONS* variable set to a value of eight (8). Therefore, the input household list will be split into eight parts to facilitate eight simultaneous router instances, each working on a separate portion of the input demand files (activities and trips). Note that *NUM_PATHSKIM_THREADS* should not be set higher than eight, as testing to date has revealed instability with the program beyond eight threads.

(Line 64) var.NUM_PARTITIONS = 8

(Line 65) var.NUM_PATHSKIM_THREADS = 8

CHAPTER 4

Running the Integrated Model System

This section provides a list of instructions to run the integrated model system. The model system requires at least 4 GB of RAM, although if additional spatial or temporal detail is required, this requirement will increase. The model does not explicitly require multiple processing cores, though the availability and speed of multiple cores will reduce model runtimes. The steps should be performed in the order they are presented.

1. Unzip the compressed archive

Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v_1_1.zip. The model files can be unpacked anywhere on your local machine (e.g., C:\Travel_Models).

2. Run the TRANSIMS Studio installer provided in the directory

..\Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v_1_1\1_Software\TRANSIMS_Studio\TRANSIMS-Studio-0.9.9-Win.exe. Simply double-click on the executable, click “run,” and follow the installation instructions (Figure 4.1).

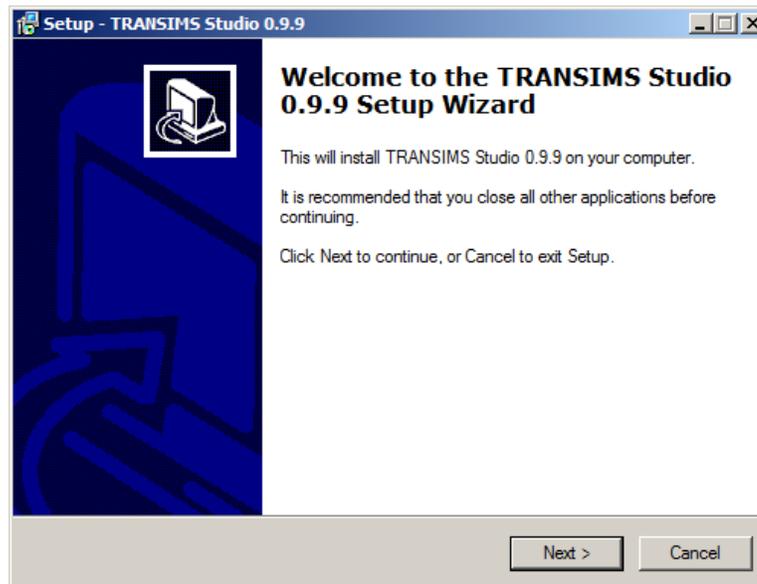


Figure 4.1. TRANSIMS Studio 0.9.9 setup wizard.

3. Locate the TRANSIMS Studio executable. Note that a desktop shortcut icon may have been created during the installation. Double-click the shortcut icon to launch the program (Figure 4.2).

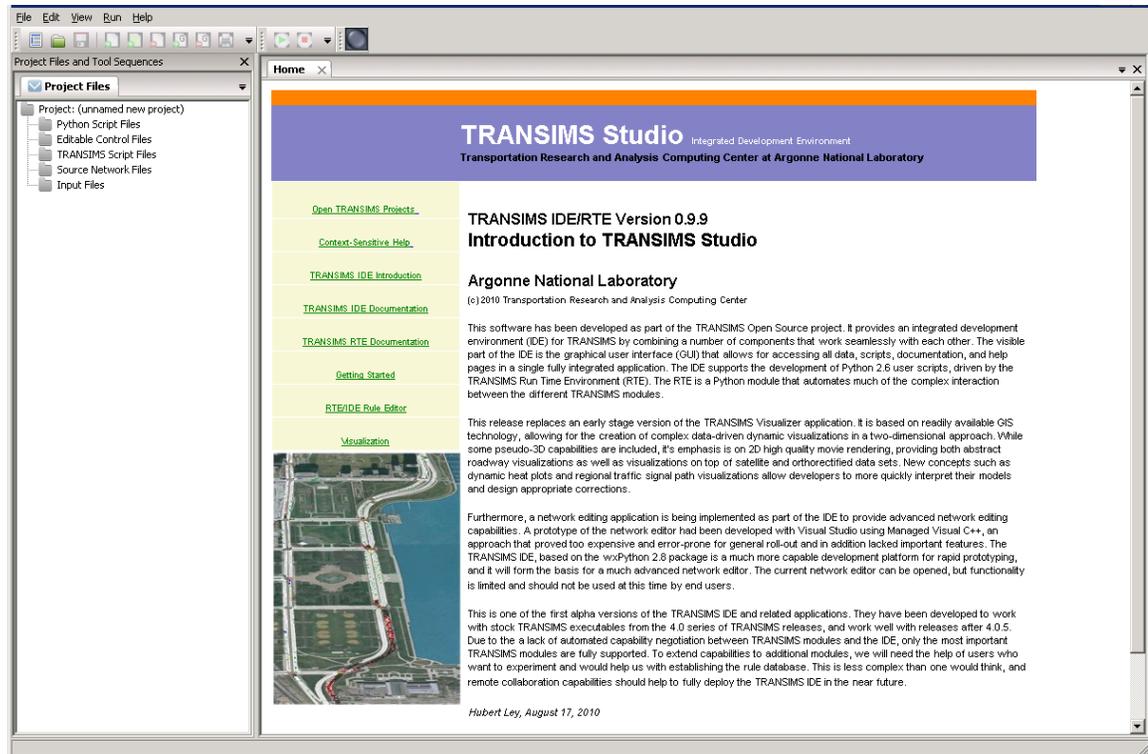


Figure 4.2. TRANSIMS Studio.

4. Locate the TRANSIMS Studio project file (.prj) and open the file. Select File → Open and select the file located at ..\
Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v_1_1\RTE_BURL\JAX_BURL_TRANSIMS_v5_Studio.prj or
 ..\
Jacksonville_Burlington_DaySim_TRANSIMS_v5_Studio_Model_v_1_1\RTE_JAX\JAX_BURL_TRANSIMS_v5_Studio.prj
5. When the *JAX_BURL_TRANSIMS_v5_Studio.prj* file is opened, note that the files related to the project will have been added in the Project Files window, which is positioned in the left-side panel of the GUI (Figure 4.3).
6. Click the '+' sign next to the Python script files to expand the list of the Python script files. At least six Python scripts should be displayed: *AssembleSkims.py*, *DaySim.py*, *Main.py*, *MsimIterations.py*, *SysDef.py* and *VisualizerPrep.py*.

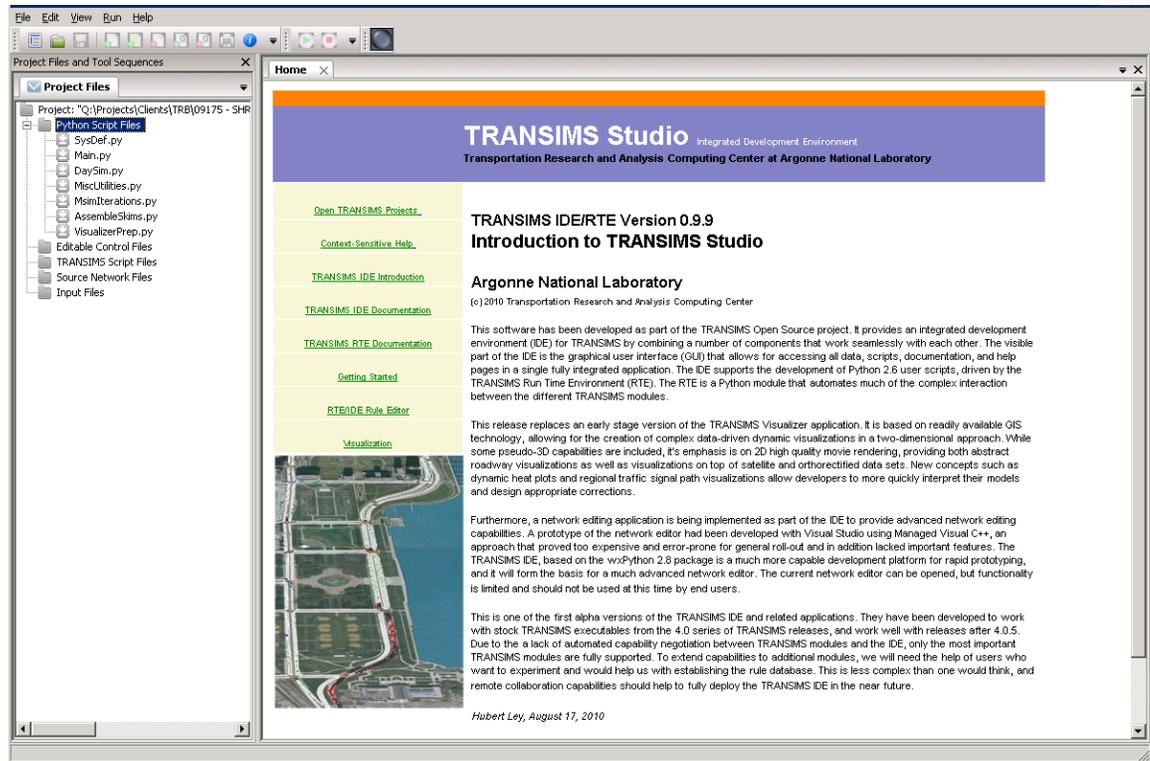


Figure 4.3. TRANSIMS Studio with project file open (*.prj).

7. Select the *Main.py* script in the list of Python script files by clicking the filename once. This will highlight the *Main.py* file. Then double-click the *Main.py* file.
8. A new tab containing the contents of the *Main.py* script file will be opened in the main GUI window (Figure 4.4).

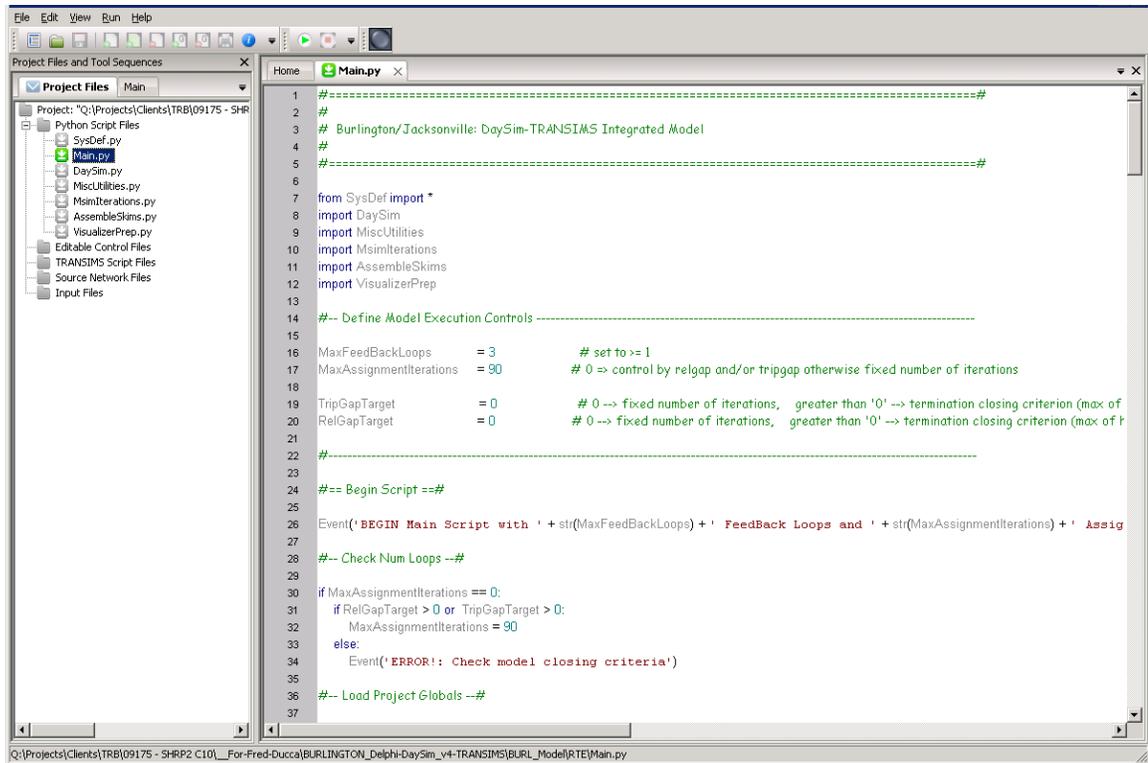


Figure 4.4. TRANSIMS Studio with Main.py displayed in the GUI.

9. Define the model execution controls. The integrated model system can be run in one of two ways: (1) using a fixed number of global (G) iterations with a fixed number of assignment (N) iterations or (2) using relative gap and tripgap convergence closure criteria.

10. To run the integrated model system with a fixed number of iterations the user must set two variables:

- a. *MaxFeedBackLoops*: the number of global feedback iterations, and
- b. *MaxAssignmentIterations*: the number of assignment iterations performed within each global iteration.

The integrated model was typically run with *MaxFeedBackLoops*=3 (G=3) and *MaxAssignmentIterations*=90 (N=90). For initial testing by a new user, it is recommended that the number of iterations be reduced to first achieve a successful complete run (e.g. G=1 and N=15).

11. To run the integrated model system with a gap-based quit closure criteria, the user must set four variables:

- a. *MaxFeedBackLoops*: the number of global feedback iterations.
- b. *MaxAssignmentIterations*: the number of assignment iterations performed within each global iteration. This variable should be set to zero (N=0).
- c. *TripGapTarget*: the assignment process will terminate when the maximum hourly trip gap value is less than the *TripGapTarget*.

d. *RelGapTarget*: the assignment process will terminate when the maximum hourly relative gap value is less than the *RelGapTarget*.

The integrated model was typically run with a fixed number of assignment iterations, and therefore the gap-based quit closure criteria was not often utilized, once this functionality was configured. Some user sensitivity testing will be required in order to select appropriate values for *TripGapTarget* and *RelGapTarget*, though values of 0.01 could be appropriate. New users are recommended to specify a fixed number of assignment iterations before attempting to use the gap-based quit closure criteria variables.

12. Remove or rename the *Main.res* file, which resumes the integrated model run from an intermediate step. The *Main.res* file is located at `..\RTE_BURL \Main.res`. A new resume file will be generated every time the model is executed. So this file must be removed as part of the model run preflight checklist, if the user desires the model to start from the very first step in the model flow.

13. To run the model, make sure the *Main.py* script has been selected and is displayed in the Studio GUI window. Double-check the model execution controls (iteration values specified on Line 16 and Line 17 of the *Main.py* script).

14. To initiate the run, click the green “play” arrow on the GUI menu bar

(Figure 4.5). Click  to start the run.  Click to stop the run.

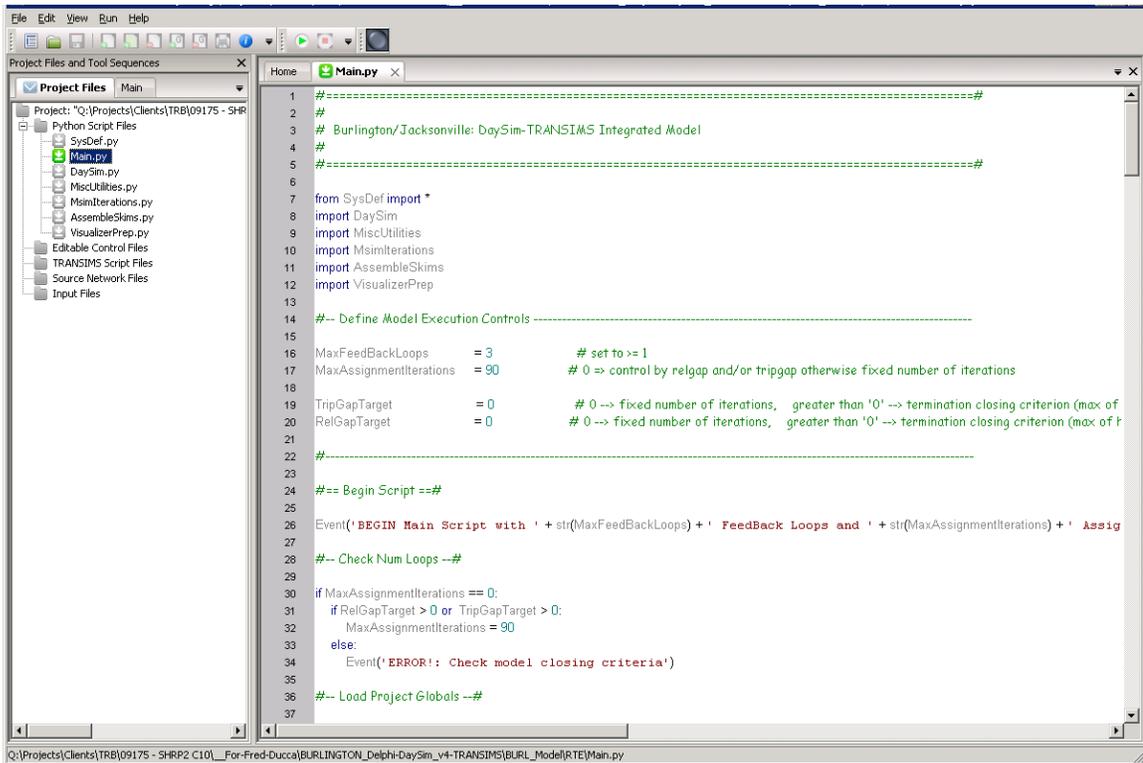


Figure 4.5. Run the model by clicking the green “play” button.

4.1 Model Runtimes

The project team extensively tested various iterative schemes to evaluate trade-offs between network assignment and model system convergence and overall model system runtimes. Generally, greater degrees of convergence are required for more spatially, temporally, or behaviorally detailed analysis. However, the level of convergence required for any given analysis is context specific, and the user is afforded many potential means with which to configure the model system and investigate performance and runtime tradeoffs. Keys and parameters which affect DaySim and TRANSIMS runtimes are easily user-configurable via two primary control files.

Model runtimes are highly influenced by the computing resources (both the memory and the number of processing cores). Runtimes in excess of 6 days were observed to run the full integrated model system with three global iterations and 90 assignment iterations using 16 processing cores. As described earlier, the user is encouraged to specify fewer global and assignment iterations in order to complete a successful run of the full integrated system. Local runtimes can then be estimated by scaling the observed runtime accordingly, if additional iterations are desired.

4.2 File Naming Convention

Output files generated by the integrated model use a prefix in the filename to identify the global iteration, assignment iteration, and model step for which the file was generated. For example, the file *2.50.Activity_Model.Performance* is the Microsimulator performance output file produced at iteration G=2 and N=50 for the Activity Model. The wildcards *@NEW@.@ALT@.Performance* file specified in the master control file were replaced with *NEW=2.50* and *ALT=Activity_Model*.

Note that the integrated model system can also use a demand trip file developed from the Burlington regional four-step model. The four-step model demand can be used instead of the activity list estimated by DaySim by modifying a variable in the system definition *SysDef.py* Python script file. Thus,

(Line 48) `var.ALT = 'Activity_Model'`

can be adjusted to

(Line 48) `var.ALT = 'Trip_Model'`.

With this change, the integrated model system will use the Trip demand converted from the four-step model. Output files will have filenames with a different prefix—for instance, *2.50.Trip_Model.Performance*. As stated earlier, the Trip Model was only utilized during the early stages of the research to help guide the development and configuration of the supply-side model while the DaySim model was still under development.