Continuous Integration of a Dynamic Multiple Agency Activity-Based Travel Modeling System

Ben Stabler
Mark Bradley
Peter Andrews

RSG
the science of insight

EXECUTIVE SUMMARY

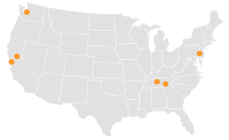
- In can be argued that activity-based (AB) travel modeling software has suffered from prototypical implementation problems such as correctness, stability, and comprehensiveness, which have helped to slow their industry adoption as well.
- We integrated an open source activity-based (AB) travel modeling systems for multiple United States transportation planning agencies under a Continuous Integration (CI) system built upon state-of-the-art software development technologies and practices
- CI is broadly defined as a collaboration practice where team members use a shared content management repository for tracking issues, software source code management, and testing. Each software revision is verified by an automated build and test system to detect problems early, in order to ultimately improve overall quality.
- Developing and maintaining advanced travel modeling tools in this online open source and collaborative workflow provides everyone in the team (developers, users, etc.) with a more robust, complete, and collaborative experience.
- We believe the multiple agency CI system developed here is a significant step toward resolving many of the implementation and maintenance issues that have plagued our industry.

1 NEED

AB microsimulation travel models are becoming the preferred modeling system for many agencies' diverse transportation and policy questions. However, many of the AB models developed to date have been singular development efforts and often suffer from typical prototype problems such as correctness, stability, and comprehensiveness. As more agencies adopt AB models, and as AB models move from research into practice, their implementation and maintenance requires improvement.

Our CI system tests each revision for compatibility in the following regional travel demand model systems:

- PSRC – Seattle area MPO
- SACOG – Sacramento area MPO
- SFCTA – San Francisco County Transportation Authority
- Nashville Area MPO
- CHCRPA – Chattanooga area MPO
- DVRPC – Philadelphia area MPO



While the core modeling system is similar in each of these regions, it also has a number of differences, including:

- Interfaces to different network modeling software
- Transit path choice modeling (especially for FTA New Starts applications)
- District to district flow balancing
- Traditional or stop-based transit access modeling
- Multiple spatial zone systems for land use and destinations
- Toll choice modeling
- Park-and-ride lot choice
- Methods for calculating short distance network level-of-service measures

Developing and maintaining a common software platform with these needs is challenging.

2 SYSTEM COMPONENTS

Our CI system makes use of a number of online open source technologies such as Jenkins, GitHub, Slack, Python, and R. Our CI system has three core components:

1. Online project site and repository
2. Continuous Integration server
3. Online agency specific repositories for testing

The entry point to the CI system is an online GitHub repository for the overall open source AB modeling software, called Daysim. DaySim simulates a day of activity and travel for each person in each household of a synthetic population distributed throughout a region. Each region implements a localized version of Daysim for synthesizing travel diaries, along with a network supply model for estimating network level-of-service. Only the Daysim travel demand model is managed by the CI system since each region's network supply model is implemented with third party technologies.

The Daysim project site does three key things:

1. source code management
2. management of issues for bugs, features, etc.
3. project and software user documentation

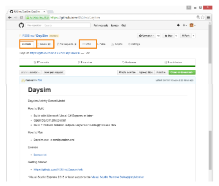


Figure 1. Online Daysim Project Site

The Jenkins CI server automatically checks-out the latest version of the software and the region-specific tests, run each regional model, compare the results to the expected results, and issues a success or failure notice. The server is run locally since the size of all model inputs and outputs for all models is 200+ GB.

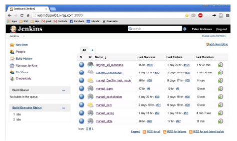


Figure 2. Jenkins CI Server

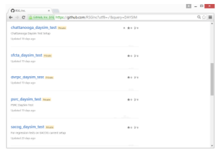


Figure 3. Agency Test Repositories

4 BENEFITS AND TRADEOFFS

The main **BENEFITS** of this CI system are:

- Users get more stable software as it has been tested in multiple regions with different configurations, travel patterns, demographics, land use, modal options, etc.
- Developers have more confidence in revisions since they test across diverse cases.
- Everyone is better informed of issues, bug fixes, and the level of effort required to develop and maintain a complex AB demand modeling software framework.
- Everyone benefits from a more efficient and repeatable workflow, which reduces the time and cost for revisions.

Key **TRADE-OFFS** made in the system setup include:

- Because a large amount of data is required, the system needed to use a desktop CI server on our network, as opposed to making use of a web-based CI service, which would likely better integrate with the other tools in the system.
- In order to reduce test run times, the region specific models were configured to run a subset of regional households. The downside of this is that the likelihood of testing low probability travel patterns decreases, which decreases code coverage.

3 SYSTEM WORKFLOW

To better understand how the collaborative CI system works, two key tasks are illustrated:

3A ADDING A NEW MODEL

In order to add a new regional model to system, the following is done:

- Create an online repository for the new regional model test data
- Run two tests
  - Compare outputs (file differences)
  - Summarize results (mode shares, etc.)
- Process outcomes
  - If pass, commit and version
  - If fail, restart process as needed

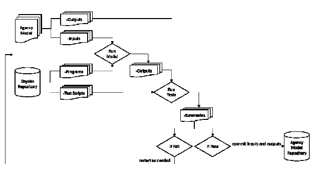


Figure 4. Add a New Regional Model

3B ADDING A NEW FEATURE

In order to add a new feature or fix a bug, the following is done:

- Create issue
- Revise software (& test data as needed)
  - Commit to develop branch
- Run two tests
  - Compare outputs (file differences)
  - Summarize results (mode shares, etc.)
- Process outcomes
  - If pass, commit to master and version
  - If fail, restart process as needed

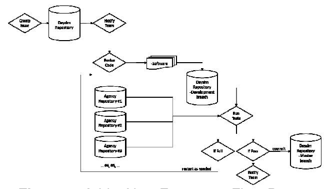


Figure 5. Add a New Feature or Fix a Bug

3C BRANCHES

A repository is a collection of files and their history. A branch is a fork in the history of a select set of files. Branches are useful for managing two versions of an agency's model inputs and outputs – the approved (i.e. tested) version and a new yet-to-be-approved (i.e. untested) version. Branches are used in both the agency specific test repositories and the Daysim project repository.

In the agency repositories, branches are used to manage the currently approved model configuration for testing. This allows the agency modelers to stage new versions of their model setup for testing, while giving the Daysim developers control over when new test data (i.e. model inputs, settings, and outputs) are introduced into the CI system. This is important since model results are often expected to change as a result of revisions to the software.

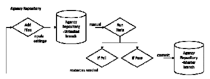


Figure 6. Agency Branches

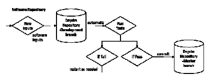


Figure 7. Software Branches

In the Daysim project repository, branches are used in a more traditional software engineering manner – i.e. for development of new features and/or correcting bugs. After developing a new feature or fixing a bug, the developer commits their revisions into a development branch, which is then tested in each region and if PASS, can then be merged into the master. If FAIL, then the developer resolves issues and restarts as needed.