Learning simulation with PySim!

Abstract

We have selected Project # 10 and are interested in learning how to simulate the airport passenger waiting time using SimPy, a process-based discrete-event simulation framework that runs in standard Python environment, and compare it with the simulation process using ARENA. [Insert summary of the simulation model and major findings and conclusions]

How SimPy works

Created by Kalus G.Muller and Tony Vignaux and released in September 2002, SimPy is a discrete-event Python framework used for event simulation by creating a virtual environment to reflect a real-world system. SimPy requires three major steps:

1. Establish the environment

2. Pass in the parameters

3. Run the simulation

Installation & Setting up the Environment

SimPy runs on Python 3(>= 3.6 or better) and supports PyP3.

To install SimPy from the command line: type pip install simpy

$ pip install simpy

Alternatively, you can also manually download it and install manually: extract the archive and open a terminal window where it is extracted from and type: python setup.py install

$ python setup.py install

To check if the installation is completed successfully: type py.test –pyargs simply

$ py.test –pyargs simpy

Software Packages used in the *airport\_simulation.py*

We have imported the following packages to successfully execute the simulation process using SimPy in Python.

import simpy

from numpy import random

We need to import numpy.random() as it will let us use both the numpy.random.uniform() and numpy.random.exponential() distributions for generating random boarding check service time as well as the personal check service time respectively.

Background and Problem Description:

Our airport waiting time simulation model is based on 1 boarding check worker, and 1 personal check scanner station. Initially we set there are 5 passengers arriving within 1-minute interval on a *Poisson* distribution(, and we define the boarding check service rate to be 0.75 minute(45 seconds) on an *exponential* distribution(, and personal check service rate is ranging between ½ minute to 1 minute. After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner where time is *uniformly* distributed between ½ minute and 1 minute. The simulation is running on a 10-minute scale with 1 simulation per cycle run in a “create-process-dispose” sequence similarly to the ARENA model process.

In the main program we have created two major classes: *Airport* and *Passenger*. The Airport reclass has two process methods: boarding\_check\_service\_time(self) and personal\_check\_service\_time(self). In addition, a \_\_init\_(self, env) method is used to start the run process every time an instance is created for both the boarding\_check process and the personal\_check\_process.

In the *Passenger class* we have create the following functions:

go\_to\_airport(self): keep tracks of the passenger and records the arrival time at the boarding check and the personal check scanner in a chronological sequence. Under this function, there are additionally three private classes created to complete the boarding simulation process: 1) The \_create\_process\_dispose method automatically disposes the entity and releases resources when finished. This method initiates this will triggers the go\_to\_airport() function in the *Passenger* class to have the access to start counting the passenger arrival time and pass the decision block to determine whether the passenger should use the boarding check resource or the personal check scanner resource during a waiting queue. The Resource serves as Process block and sorts the check-in processes in a FIFO(first in first out) manner.2) The \_determine\_resource() class classifies the resources service time based on the personal check service or the boarding check service and records the timestamp. 3) Finally the \_decision\_block() class creates a decision block that determines …

[To be continuously updated and edited…]

References:

1. <https://realpython.com/simpy-simulating-with-python/#how-simulation-works>