Tara Halsted

4/19/2023

Assigned to look into discrete event simulation for Week 2 meeting.

Here are my notes from researching it:

* Discrete-event simulation—often used to model queueing activities
* Watched this video (1hr, 48mins): <https://youtu.be/jXDjrWKcu6w>
  + Entities are the things flowing through (EX: patients) processes and are waiting for resources
  + Generators—things that create the entities
    - Can have more than one (EX: multiple ways that patients can come in, such as patients arriving themselves or through EMS)
  + Interarrival times
  + Activities/servers—stuff that happens to entities (EX: triage, treatment ward, admission)
  + Activity/server time—amount of time for an activity to happen
    - Typically model with distributions to account for variability
  + Resources—required for activities
    - Can be shared between activities (EX: nurses doing multiple things)
    - Queues—where entities are held until activity has capacity + resources necessary to start
      * Often physical (standing and waiting), but could be more abstract (waiting list over long period of time)
    - Sinks—how entities leave the model (left the part of system we are modeling)
  + Generator functions—python functions
    - EX: Count, start, infinite while loop, but uses “yield” count key word instead of “return” count
    - Remembers where you are when you left (unlike normal functions, which run as if never been run before)
  + Time usually described in minutes

4/24/2023

Here are notes for three of the discrete-event simulation methods I looked into:

* **1st simulation method: running python on desktop**
  + Pros:
    - Would be a good way to do this if it would work, but could not get it to work on a desktop. Next: can I get this to run online instead?
  + Cons:
    - Very hard to get path set up on my Mac computer and could not even get it set up
  + Here is what I tried:
  + Setting-up python
    - * Have to get python path set up
      * In environment variables on path
      * Path—list of folders
    - echo $SHELL
      * Echo prints out value of string
      * $=indicating an environment variable
      * SHELL=an environment variables
      * shell value of variable SHELL
      * 1st word is command, any words after separated by spaces are arguments to program
    - ~=your home directory (shortcut)
    - nano ~/.zshrc
      * zshrc=code that runs when you open terminal; what happens when you open a terminal
      * .=hidden file if starts with “.”
        + Doesn’t show up in file explorer
      * Nano=command line file editor
    - Download python on Visual Studio Code
      * <https://www.python.org/downloads/>
      * <file:///Library/Frameworks/Python.framework/Versions/3.11/Resources/English.lproj/Documentation/using/mac.html#getting-and-installing-macpython>
      * <https://www.makeuseof.com/how-to-install-python-on-mac/>
      * <https://code.visualstudio.com/>
  + Video tutorial: <https://www.youtube.com/watch?v=jXDjrWKcu6w&ab_channel=HSMA>
    - Entities are the things flowing through (EX: patients) processes and are waiting for resources
    - Generators—things that create the entities
      * Can have more than one (EX: multiple ways that patients can come in, such as patients arriving themselves or through EMS)
      * Interarrival times
    - Activities/servers—stuff that happens to entities (EX: triage, treatment ward, admission)
    - Activity/server time—amount of time for an activity to happen
      * Typically model with distributions to account for variability
    - Resources—required for activities
      * Can be shared between activities (EX: nurses doing multiple things)
      * Queues—where entities are held until activity has capacity + resources necessary to start
        + Often physical (standing and waiting), but could be more abstract (waiting list over long period of time)
    - Sinks—how entities leave the model (left the part of system we are modeling)
    - Generator functions—python functions
      * EX: Count, start, infinite while loop, but uses “yield” count key word instead of “return” count
      * Remembers where you are when you left (unlike normal functions, which run as if never been run before)
    - Time usually described in minutes

* **2nd simulation method: Python with SimPy libraries on Google Colab (an online software for running python)** <https://colab.research.google.com/>
  + Pros:
    - Google Colab works like Google docs and can be shared and edited by group of people simultaneously
    - Work is done online and works on Mac and Windows computers
    - Free
    - Could learn a lot about python by working on this project using this method
  + Cons:
    - Output is not as visual (does have visuals, but hard to get the animation library to work so far. Will continue to try)
    - Complex SimPy library that would be hard to learn in short amount of time
      * Is there someone that could help us learn or a tutorial that could speed this up?
  + Testing
    - Asked ChatGPT for examples of how to set up process flow so that we could learn what it would look like and if this is feasible for us to learn

I put this in colab to try it out:

test\_1 file: <https://colab.research.google.com/drive/1XLM-KBM7hqNm2_s9CgND7g3ZR34sac2n?usp=sharing>

test\_2 file: <https://colab.research.google.com/drive/1wiCcN4gSM0qCms4fmEZufwQoehAB92aI?usp=sharing>

test\_3 file (working on getting an animation visual, but have not yet gotten the animation to work): <https://colab.research.google.com/drive/1NhJZaxokifwtSbFz-_oH6Rfgw-DAEWs6?usp=sharing>

* **3rd simulation method: JaamSim** <https://jaamsim.com/>
  + Pros:
    - Free and open-source
    - Drag and drop style, so we would not have to code
    - More visual format of output
    - Seems to have some limitations with model (can get two lines to feed into one, but have not yet gotten a line to split into multiple)
  + Cons:
    - Works on Windows, but only on Macs that have Open GL support (newer macs have deprecated Open GL support, so this would not work on a new Mac)
  + Testing
    - <https://www.youtube.com/watch?v=doly56hk1U4&ab_channel=ChristofDefryn>
      * Followed this YouTube tutorial on making a single-server model