Deliverable 2

SYSC 4005 Project Group # 7

Thomas de Haan Carriere – 100947281

Benjamin Palko – 100964652

## Model Translation

As determined in deliverable 1, the simulation of the factory was found to be a physical, dynamic and stochastic model. From the histograms of the given data sets for the inspection times and workstation processing times, it was visually determined that they follow an exponential distribution with decay. This is further supported by the physical basis of the distribution being *“time between independent events, or a process time that is memoryless”*, mentioned in lecture 4, slide 8.

Python was chosen as the programming language with which to perform the simulation of the model. Python is a versatile and easy to use language which we are familiar with. MATLAB was considered, but it is an unconventional software that we are less accustomed to. “Simpy” is a Python module for process-based discrete-event simulation and was used to facilitate the simulation. “Simpy” was very easy to learn and is very flexible.

In order to supply the simulation with random variables, a Python script called “Model.py” was created. There are methods in this file pertaining to each inspector and their relevant components and each workstation, whereby random values can be generated. For example, the method “workstation3()” imports the “ws3.dat” file and reads the values into a list. The method then calls and returns the output of the function “calc\_rand\_list(list)”, passing it the list of data values. This function will calculate the sum of all values in the list and divide it by 300 in order to derive the mean. Since the distribution of the model is exponential, the mean is then the value. Using the Python module “numpy”, a module for performing scientific computing, a random value can be computed using the expression “numpy.random.exponential(mean, 1)”, where mean is the calculated value from the data, and 1 is the number of values to return. This is then returned to the method which will return it to the function call in “main.py”, where the simulation is taking place. See attached source code.

The simulation uses a producer and consumer model, whereby inspectors are the consumers

In order to implement an alternate design, the goal is the simulate having an additional inspector which will solely focus on the inspection of component 3 and removing inspector 2’s responsibility of inspecting component 3. This will provide one inspector for each component. This may improve processing times for the workstations but may increase the idle time of the inspectors. At the present, inspectors are represented as Python classes, each dealing with their respective components. To implement an additional inspector, an additional class would be created to represent the new inspector, with its behaviour requiring slight modifications. The behaviour of inspector 2 would also be modified so it no longer accepts component 2. “Model.py” would also need to be altered considering the new inspector will need random values generated. This would be achieved by altering the method which generates values for the service time of inspector 2 working on component 3. This method would instead be used to pass values to the new inspector, as it would represent the time needed to service component 3.