DISCRETE EVENT SIMULATION

1. [**Is it possible to increase throughput and reduce waiting time in a system at the same time for the case described?**](https://www.researchgate.net/post/Is_it_possible_to_increase_throughput_and_reduce_waiting_time_in_a_system_at_the_same_time_for_the_case_described)

Generally, yes. From little's law, it is possible to increase throughput and decrease waiting time by increasing service rate. It can be achieved by improving process efficiency or by investing in better technology. You may also want to look into the system utilization. The system may not perfectly utilized due to specific customer arrival behavior. For example most of customers may tend to arrive in early morning which result in longer waiting time.

1. **Which methods are used to determine replications (run number) for discrete event simulation? Which one outperforms than the others?**

Regarding simulation, the number of replications determines the width of your confidence intervals (reduces variance). The limiting factors will be computing time and expense. Assuming that performing N replications achieves a satisfactory estimate of mean performance as required by the user, performing more than N replications may be an unnecessary use of computer time and considerable expense. However, performing fewer than N replications could lead to inaccurate results and thus to incorrect decisions being made.

There are three main methods found in the literature for choosing N: Rule of Thumb (Law sInterval (with Specified Precision) Method (Robinson 2004, Law 2007, Banks et al. 2005). Law and McComas (1990) recommend running at least 3 to 5 replications. This rule of thumb is useful for telling users that relying upon the results of only one run is unwise. However, it makes no allowance for the characteristics of a model’s output, so although running 3 to 5 replications may suffice for one model it may be woefully inadequate for another.

In general, The Confidence Interval (with Specified Precision) Method is the one chosen both statistics wise as well as its capability to be adapted into an algorithm for automation for testing using artificial and real models. This method runs increasing numbers of replications until the confidence intervals constructed around the chosen output variable (e.g. mean queue length) using the t-statistic, are within a (user) specified precision. This allows the user to tailor the accuracy of output results to their particular requirement or purpose for that model and result. This method assumes that the cumulative mean has a normal distribution, (which is true under the Central Limit Theorem when the number of replications is large).

1. **How do you determine the correct length of a simulation run?**

Preliminary estimates of required run lengths can be obtained by approximating the stochastic model of interest by a more elementary Markov model that can be analyzed analytically. When steady-state quantities are to be estimated by sample means, we often can estimate required run lengths by calculating the asymptotic variance and the asymptotic bias of the sample mean in the Markov model.

1. **List Open Source Software for discrete event simulation.**

Following are software’s:

1. CPN TOOLS
2. DESMO-J
3. Facsimile
4. PowerDEVS
5. Ptolemy ii
6. SIM.JS
7. Simply
8. Simula
9. SystemC
10. **What is Event Scheduling?**

* Basic building block is the event
* Model program's code segments consist of event routines waiting to be executed
* Event routine associated with each event type -- performs required operations for that type
* Simulation executive moves from event to event executing the corresponding event routines

1. **What is Processes, Coroutines, Objects, and Events in Discrete Event Simulation?**

A simulation can use different implementation techniques (+ indicates advantages, − indicates drawbacks in the following lists):

* *Processes*   
  + "natural" translation from real process descriptions to code, e.g. sequential behavior translates to sequential code.  
  − overhead of process switching,  
  − avoiding unintended interactions through shared variables.
* *Coroutines*  
  + similar to processes but lighter-weight  
  + programmer controls context switching (when simulation time advances), avoiding some problems of true concurrency.  
  − coroutines may not be available  
  − overhead of stacks and other state.
* *Objects*  
  + lighter-weight than coroutines.  
  − activity phases must be implemented separately  
  − object must store state indicating the next activity since there is only one entry point where the object is activated.
* *Events*+ still lighter-weight   
  + avoid the need to explicitly allocate/deallocate storage to represent a process, coroutine, or object.  
  − all simulated process state must be passed through event parameters since there is no object or other entity that can retain state information.