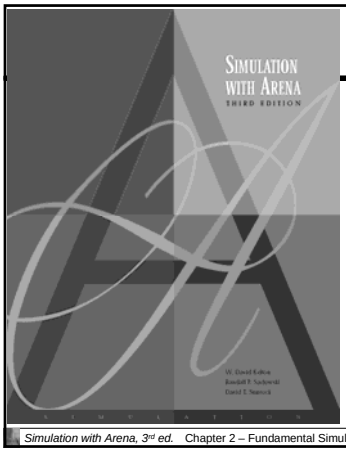


Fundamental Simulation Concepts

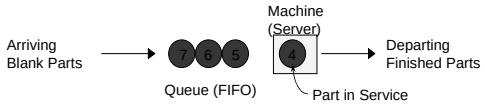
Chapter 2

Last revision June 7, 2003

What We'll Do ...

- **Underlying ideas, methods, and issues in simulation**
- **Software-independent (setting up for Arena)**
- **Centered around an example of a simple processing system**
  - Decompose the problem
  - Terminology
  - Simulation by hand
  - Some basic statistical issues
  - Overview of a simulation study

The System: A Simple Processing System



- **General intent:**
  - Estimate expected production
  - Waiting time in queue, queue length, proportion of time machine is busy
- **Time units**
  - Can use different units in different places … must declare
  - Be careful to check the units when specifying inputs
  - Declare *base time units* for internal calculations, outputs
  - Be reasonable (interpretation, roundoff error)

Model Specifics

- **Initially (time 0) empty and idle**
- **Base time units: minutes**
- **Input data (assume given for now …), in minutes:**

Part Number	Arrival Time	Interarrival Time	Service Time
1	0.00	1.73	2.90
2	1.73	1.35	1.76
3	3.08	0.71	3.39
4	3.79	0.62	4.52
5	4.41	14.28	4.46
6	18.69	0.70	4.36
7	19.39	15.52	2.07
8	34.91	3.15	3.36
9	38.06	1.76	2.37
10	39.82	1.00	5.38
11	40.82	.	.
.	.	.	.
.	.	.	.

- **Stop when 20 minutes of (simulated) time have passed**

Goals of the Study: Output Performance Measures

- ***Total production* of parts over the run (*P*)**
- ***Average waiting time* of parts in queue:**

$$\frac{\sum_{i=1}^{N} WQ_i}{N}$$

***N* = no. of parts completing queue wait**
***WQ*$_i$ = waiting time in queue of *i*th part**
Know: *WQ*$_1$ = 0 (why?)
***N* ≥ 1 (why?)**

- ***Maximum waiting time* of parts in queue:**

$$\max_{i=1,\ldots,N} WQ_i$$

Goals of the Study: Output Performance Measures (cont'd.)

- ***Time-average number of parts in queue*:**

$$\frac{\int_0^{20} Q(t)\,dt}{20}$$

Q*(*t*) = number of parts in queue at time *t

- ***Maximum number of parts in queue*:** $\max_{0 \le t \le 20} Q(t)$
- ***Average* and *maximum total time in system* of parts (a.k.a. *cycle time*):**

$$\frac{\sum_{i=1}^{P} TS_i}{P}, \quad \max_{i=1,\ldots,P} TS_i$$

TS*$_i$ = time in system of part *i

Goals of the Study: Output Performance Measures (cont'd.)

- ***Utilization*** **of the machine (proportion of time busy)**

$$\frac{\int_0^{20} B(t)\,dt}{20}, \quad B(t) = \begin{cases} 1 & \text{if the machine is busy at time } t \\ 0 & \text{if the machine is idle at time } t \end{cases}$$

- **Many others possible (information overload?)**

Analysis Options

- **Educated guessing**
  - Average interarrival time = 4.08 minutes
  - Average service time = 3.46 minutes
  - So (on average) parts are being processed faster than they arrive
    - System has a chance of operating in a stable way in the long run, i.e., might not "explode"
    - If all interarrivals and service times were exactly at their mean, there would never be a queue
    - But the data clearly exhibit variability, so a queue could form
  - If we'd had average interarrival < average service time, and this persisted, then queue would explode
  - Truth — between these extremes
  - Guessing has its limits …

Analysis Options (cont'd.)

- **Queueing theory**
  - Requires additional assumptions about the model
  - Popular, simple model: *M/M/1 queue*
    - Interarrival times ~ exponential
    - Service times ~ exponential, indep. of interarrivals
    - Must have E(service) < E(interarrival)
    - Steady-state (long-run, forever)
    - Exact analytic results; e.g., average waiting time in queue is

$$\frac{\mu_S^2}{\mu_A - \mu_S}, \quad \begin{matrix} \mu_A = \text{E(interarrival time)} \\ \mu_S = \text{E(service time)} \end{matrix}$$

  - Problems: validity, estimating means, time frame
  - Often useful as first-cut approximation

Mechanistic Simulation

- **Individual operations (arrivals, service times) will occur exactly as in reality**
- **Movements, changes occur at the right "time," in the right order**
- **Different pieces interact**
- **Install "observers" to get output performance measures**
- **Concrete, "brute-force" analysis approach**
- **Nothing mysterious or subtle**
  - But a lot of details, bookkeeping
  - Simulation software keeps track of things for you

Pieces of a Simulation Model

- ***Entities***
  - "Players" that move around, change status, affect and are affected by other entities
  - *Dynamic objects* — get created, move around, leave (maybe)
  - Usually represent "real" things
    - Our model: entities are the parts
  - Can have "fake" entities for modeling "tricks"
    - Breakdown demon, break angel
      Though Arena has built-in ways to model these examples directly
  - Usually have multiple *realizations* floating around
  - Can have different types of entities concurrently
  - Usually, identifying the types of entities is the first thing to do in building a model

Pieces of a Simulation Model (cont'd.)

- ***Attributes***
  - Characteristic of all entities: describe, differentiate
  - All entities have same attribute "slots" but different values for different entities, for example:
    - Time of arrival
    - Due date
    - Priority
    - Color
  - Attribute value tied to a specific entity
  - Like "local" (to entities) variables
  - Some automatic in Arena, some you define

Pieces of a Simulation Model (cont'd.)

- **(Global) *Variables***
  - Reflects a characteristic of the whole model, not of specific entities
  - Used for many different kinds of things
    - Travel time between all station pairs
    - Number of parts in system
    - Simulation clock (built-in Arena variable)
  - Name, value of which there's only one copy for the whole model
  - Not tied to entities
  - Entities can access, change variables
  - Writing on the wall (rewriteable)
  - Some built-in by Arena, you can define others

Pieces of a Simulation Model (cont'd.)

- ***Resources***
  - What entities compete for
    - People
    - Equipment
    - Space
  - Entity *seizes* a resource, uses it, *releases* it
  - Think of a *resource being assigned to an entity*, rather than an entity "belonging to" a resource
  - "A" resource can have several *units* of capacity
    - Seats at a table in a restaurant
    - Identical ticketing agents at an airline counter
  - Number of units of resource can be changed during the simulation

Pieces of a Simulation Model (cont'd.)

- ***Queues***
  - Place for entities to wait when they can't move on (maybe since the resource they want to seize is not available)
  - Have names, often tied to a corresponding resource
  - Can have a finite capacity to model limited space — have to model what to do if an entity shows up to a queue that's already full
  - Usually watch the length of a queue, waiting time in it

Pieces of a Simulation Model (cont'd.)

- ***Statistical accumulators***
  - Variables that "watch" what's happening
  - Depend on output performance measures desired
  - "Passive" in model — don't participate, just watch
  - Many are automatic in Arena, but some you may have to set up and maintain during the simulation
  - At end of simulation, used to compute final output performance measures

Pieces of a Simulation Model (cont'd.)

- **Statistical accumulators for the simple processing system**
  - Number of parts produced so far
  - Total of the waiting times spent in queue so far
  - No. of parts that have gone through the queue
  - Max time in queue we've seen so far
  - Total of times spent in system
  - Max time in system we've seen so far
  - Area so far under queue-length curve $Q(t)$
  - Max of $Q(t)$ so far
  - Area so far under server-busy curve $B(t)$

Simulation Dynamics: The Event-Scheduling "World View"

- **Identify characteristic *events***
- **Decide on *logic* for each type of event to**
  - Effect *state changes* for each event type
  - Observe statistics
  - Update times of future events (maybe of this type, other types)
- **Keep a simulation *clock*, future *event calendar***
- ***Jump* from one event to the next, process, observe statistics, update event calendar**
- **Must specify an appropriate *stopping rule***
- **Usually done with general-purpose programming language (C, FORTRAN, etc.)**

Events for the Simple Processing System

- ***Arrival* of a new part to the system**
  - Update time-persistent statistical accumulators (from last event to now)
    - Area under $Q(t)$
    - Max of $Q(t)$
    - Area under $B(t)$
  - "Mark" arriving part with current time (use later)
  - If machine is idle:
    - Start processing (schedule departure), Make machine busy, Tally waiting time in queue (0)
  - Else (machine is busy):
    - Put part at end of queue, increase queue-length variable
  - Schedule the next arrival event

Events for the Simple Processing System (cont'd.)

- ***Departure* (when a service is completed)**
  - Increment number-produced stat accumulator
  - Compute & tally time in system (now - time of arrival)
  - Update time-persistent statistics (as in arrival event)
  - If queue is non-empty:
    - Take first part out of queue, compute & tally its waiting time in queue, begin service (schedule departure event)
  - Else (queue is empty):
    - Make the machine idle (Note: there will be no departure event scheduled on the future events calendar, which is as desired)

Events for the Simple Processing System (cont'd.)

- ***The End***
  - Update time-persistent statistics (to end of the simulation)
  - Compute final output performance measures using current (= final) values of statistical accumulators
- **After each event, the event calendar's top record is removed to see what time it is, what to do**
- **Also must initialize everything**

Some Additional Specifics for the Simple Processing System

- **Simulation clock variable (internal in Arena)**
- **Event calendar: list of event *records*:**
  - [Entity No., Event Time, Event Type]
  - Keep *ranked* in increasing order on Event Time
  - Next event always in top record
  - Initially, schedule first Arrival, The End (Dep.?)
- **State variables: describe current status**
  - Server status $B(t)$ = 1 for busy, 0 for idle
  - Number of customers in queue $Q(t)$
  - Times of arrival of each customer now in queue (a list of random length)

Simulation by Hand

- **Manually track state variables, statistical accumulators**
- **Use "given" interarrival, service times**
- **Keep track of event calendar**
- **"Lurch" clock from one event to the next**
- **Will omit times in system, "max" computations here (see text for complete details)**

Simulation by Hand: Setup

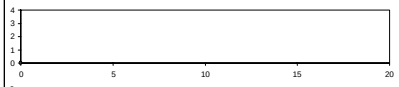
System	Clock	$B(t)$	$Q(t)$	Arrival times of custs. in queue	Event calendar
Number of completed waiting times in queue	Total of waiting times in queue		Area under $Q(t)$		Area under $B(t)$
$Q(t)$ graph $B(t)$ graph					
Interarrival times	1.73, 1.35, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...				
Service times	2.90, 1.76, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...				

Simulation by Hand: *t* = 0.00, Initialize

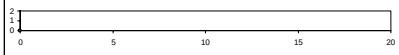
System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
	0.00	0	0	<empty>	[1, 0.00, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
0	0.00	0.00	0.00

Q(*t*) graph



B(*t*) graph



Time (Minutes)

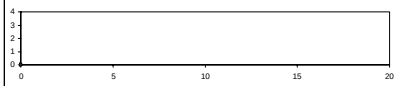
Interarrival times	1.73, 1.35, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	2.90, 1.76, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

Simulation by Hand: *t* = 0.00, Arrival of Part 1

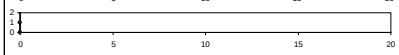
System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
1	0.00	1	0	<empty>	[2, 1.73, Arr] [1, 2.90, Dep] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
1	0.00	0.00	0.00

Q(*t*) graph



B(*t*) graph



Time (Minutes)

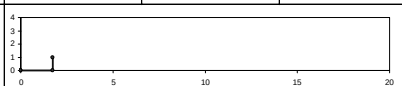
Interarrival times	~~1.73~~, 1.35, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.90~~, 1.76, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

Simulation by Hand: *t* = 1.73, Arrival of Part 2

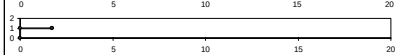
System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
2 1	1.73	1	1	(1.73)	[1, 2.90, Dep] [3, 3.08, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
1	0.00	0.00	1.73

Q(*t*) graph



B(*t*) graph



Time (Minutes)

Interarrival times	~~1.73~~, ~~1.35~~, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.90~~, 1.76, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

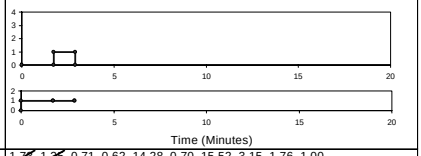
Simulation by Hand: *t* = 2.90, Departure of Part 1

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
2	2.90	1	0	<empty>	[3, 3.08, Arr] [2, 4.66, Dep] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
2	1.17	1.17	2.90

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.90~~, ~~1.76~~, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

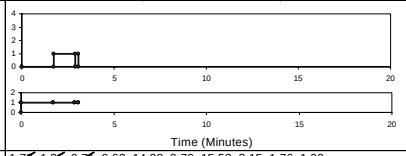
Simulation by Hand: *t* = 3.08, Arrival of Part 3

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
3 2	3.08	1	1	(3.08)	[4, 3.79, Arr] [2, 4.66, Dep] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
2	1.17	1.17	3.08

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.90~~, ~~1.76~~, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

Simulation by Hand: *t* = 3.79, Arrival of Part 4

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
4 3 2	3.79	1	2	(3.79, 3.08)	[5, 4.41, Arr] [2, 4.66, Dep] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
2	1.17	1.88	3.79

Q(*t*) graph

B(*t*) graph

Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.90~~, ~~1.76~~, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

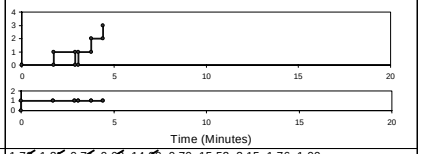
Simulation by Hand: *t* = 4.41, Arrival of Part 5

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
5 4 3 2	4.41	1	3	(4.41, 3.79, 3.08)	[2, 4.66, Dep] [6, 18.69, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
2	1.17	3.12	4.41

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.82~~, ~~1.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	2.90, ~~1.76~~, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

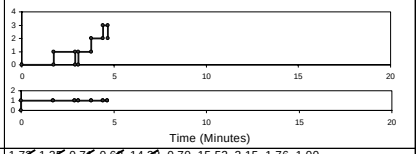
Simulation by Hand: *t* = 4.66, Departure of Part 2

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
5 4 3	4.66	1	2	(4.41, 3.79)	[3, 8.05, Dep] [6, 18.69, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
3	2.75	3.87	4.66

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.82~~, ~~1.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	2.90, ~~1.76~~, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

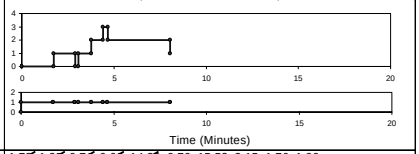
Simulation by Hand: *t* = 8.05, Departure of Part 3

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
5 4	8.05	1	1	(4.41)	[4, 12.57, Dep] [6, 18.69, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
4	7.01	10.65	8.05

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.82~~, ~~1.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	2.90, ~~1.76~~, ~~3.39~~, ~~4.52~~, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

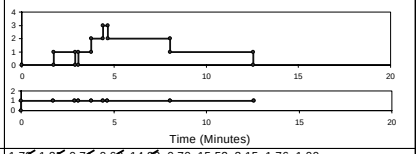
Simulation by Hand: *t* = 12.57, Departure of Part 4

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
5	12.57	1	0	()	[5, 17.03, Dep] [6, 18.69, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
5	15.17	15.17	12.57

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.76~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~1.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.80~~, ~~1.76~~, ~~3.35~~, ~~4.52~~, ~~4.46~~, 4.36, 2.07, 3.36, 2.37, 5.38, ...

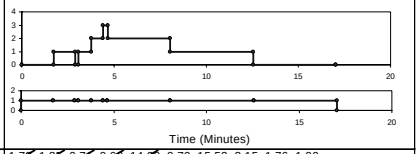
Simulation by Hand: *t* = 17.03, Departure of Part 5

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
	17.03	0	0	()	[6, 18.69, Arr] [–, 20.00, End]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
5	15.17	15.17	17.03

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.76~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~1.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.80~~, ~~1.76~~, ~~3.35~~, ~~4.52~~, ~~4.46~~, 4.36, 2.07, 3.36, 2.37, 5.38, ...

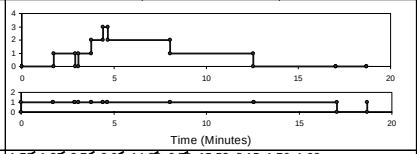
Simulation by Hand: *t* = 18.69, Arrival of Part 6

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
6	18.69	1	0	()	[7, 19.39, Arr] [–, 20.00, End] [6, 23.05, Dep]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
6	15.17	15.17	17.03

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.76~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~1.28~~, ~~0.70~~, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.80~~, ~~1.76~~, ~~3.35~~, ~~4.52~~, ~~4.46~~, ~~4.36~~, 2.07, 3.36, 2.37, 5.38, ...

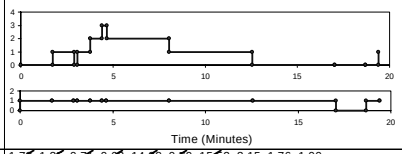
Simulation by Hand: *t* = 19.39, Arrival of Part 7

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
7 6	19.39	1	1	(19.39)	[–, 20.00, End] [6, 23.05, Dep] [8, 34.91, Arr]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
6	15.17	15.17	17.73

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~14.28~~, ~~0.70~~, ~~15.52~~, 3.15, 1.76, 1.00, ...
Service times	~~2.80~~, ~~1.76~~, ~~3.29~~, ~~4.52~~, ~~4.46~~, ~~4.36~~, 2.07, 3.36, 2.37, 5.38, ...

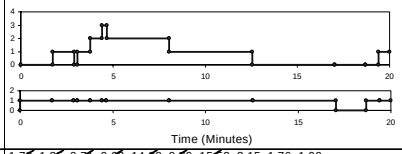
Simulation by Hand: *t* = 20.00, The End

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
7 6	20.00	1	1	(19.39)	[6, 23.05, Dep] [8, 34.91, Arr]

Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
6	15.17	15.78	18.34

Q(*t*) graph

B(*t*) graph



Interarrival times	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~14.28~~, ~~0.70~~, ~~15.52~~, 3.15, 1.76, 1.00, ...
Service times	~~2.80~~, ~~1.76~~, ~~3.29~~, ~~4.52~~, ~~4.46~~, ~~4.36~~, 2.07, 3.36, 2.37, 5.38, ...

Simulation by Hand: Finishing Up

- **Average waiting time in queue:**

$$\frac{\text{Total of times in queue}}{\text{No. of times in queue}} = \frac{15.17}{6} = 2.53 \text{ minutes per part}$$

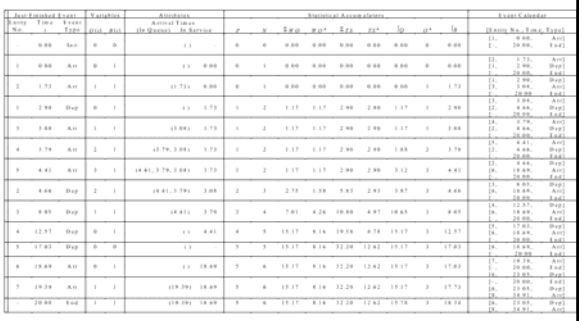
- **Time-average number in queue:**

$$\frac{\text{Area under } Q(t) \text{ curve}}{\text{Final clock value}} = \frac{15.78}{20} = 0.79 \text{ part}$$

- **Utilization of drill press:**

$$\frac{\text{Area under } B(t) \text{ curve}}{\text{Final clock value}} = \frac{18.34}{20} = 0.92 \text{ (dimensionless)}$$

Complete Record of the Hand Simulation



Event-Scheduling Logic via Programming

- **Clearly well suited to standard programming language**
- **Often use "utility" libraries for:**
  - List processing
  - Random-number generation
  - Random-variate generation
  - Statistics collection
  - Event-list and clock management
  - Summary and output
- **Main program ties it together, executes events in order**

Simulation Dynamics: The Process-Interaction World View

- **Identify characteristic *entities* in the system**
- **Multiple copies of entities co-exist, interact, compete**
- **"Code" is non-procedural**
- **Tell a "story" about what happens to a "typical" entity**
- **May have many types of entities, "fake" entities for things like machine breakdowns**
- **Usually requires special simulation software**
  - Underneath, still executed as event-scheduling
- **The view normally taken by Arena**
  - Arena translates your model description into a program in the SIMAN simulation language for execution

Randomness in Simulation

- **The above was just one "replication" — a sample of size one (not worth much)**
- **Made a total of five replications:**

Performance Measure	Replication 1	2	3	4	5	Sample Avg.	Std. Dev.	95% Half Width
Total production	5	3	6	2	3	3.80	1.64	2.04
Average waiting time in queue	2.53	1.19	1.03	1.62	0.00	1.27	0.92	1.14
Maximum waiting time in queue	8.16	3.56	2.97	3.24	0.00	3.59*	2.93*	3.63*
Average total time in system	6.44	5.10	4.16	6.71	4.26	5.33	1.19	1.48
Maximum total time in system	12.62	6.63	6.27	7.71	4.96	7.64*	2.95*	3.67*
Time-average number of parts in queue	0.79	0.18	0.36	0.16	0.05	0.31	0.29	0.36
Maximum number of parts in queue	3	1	2	1	1	1.60*	0.89*	1.11*
Drill-press utilization	0.92	0.59	0.90	0.51	0.70	0.72	0.18	0.23

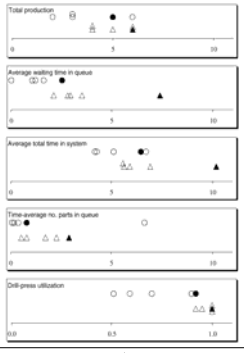
Note substantial variability across replications

- **Confidence intervals for expected values:**
  - In general, $\bar{X} \pm t_{n-1,1-\alpha/2} s/\sqrt{n}$
  - For expected total production, $3.80 \pm (2.776)(1.64/\sqrt{5})$
    3.80 ± 2.04

Comparing Alternatives

- **Usually, simulation is used for more than just a single model "configuration"**
- **Often want to compare alternatives, select or search for the best (via some criterion)**
- **Simple processing system: What would happen if the arrival rate were to double?**
  - Cut interarrival times in half
  - Rerun the model for double-time arrivals
  - Make five replications

Results: Original vs. Double-Time Arrivals



- **Original – circles**
- **Double-time – triangles**
- **Replication 1 – filled in**
- **Replications 2-5 – hollow**
- **Note variability**
- **Danger of making decisions based on one (first) replication**
- **Hard to see if there are really differences**
- **Need: Statistical analysis of simulation output data**

Overview of a Simulation Study

- Understand the system
- Be clear about the goals
- Formulate the model representation
- Translate into modeling software
- Verify “program”
- Validate model
- Design experiments
- Make runs
- Analyze, get insight, document results