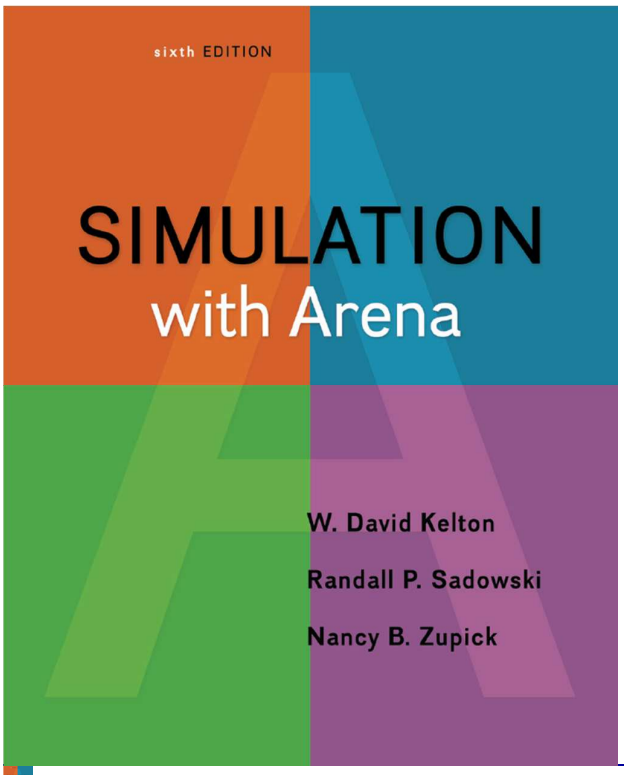


What is Simulation?

Chapter 1

Last revision December 21, 2013

Simulation Is …

- ***Simulation*** **– very broad term – methods and applications to imitate or mimic real systems, usually via computer**
- **Applies in many fields, industries**
- **Very popular, powerful**
- **Book covers simulation in general, *Arena* simulation software in particular**
- **This chapter – general ideas, terminology, examples of applications, good/bad things, kinds of simulation, software options, how/when simulation is used**

Systems

- ***System*** **– facility or process, actual or planned**
  - Examples abound …
    - Manufacturing facility
    - Bank operation
    - Airport operations (passengers, security, planes, crews, baggage)
    - Transportation/logistics/distribution operation
    - Hospital facilities (emergency room, operating room, admissions)
    - Computer network
    - Freeway system
    - Business process (insurance office)
    - Criminal justice system
    - Chemical plant
    - Fast-food restaurant
    - Supermarket
    - Theme park
    - Emergency-response system
    - Shipping ports, berths
    - Military combat, logistics

Work With the System?

- **Study system – measure, improve, design, control**
  - Maybe just play with actual system
    - Advantage — unquestionably looking at the right thing
  - But often impossible in reality with actual system
    - System doesn't exist
    - Would be disruptive, expensive, dangerous

Models

- ***Model*** **– set of assumptions/approximations about how system works**
  - Study model instead of real system … usually much easier, faster, cheaper, safer
  - Can try wide-ranging ideas with model
    - Make your mistakes on the computer where they *don't* count, rather than for real where they *do* count
  - Often, just *building* model is instructive – regardless of results
  - Model *validity* (any kind of model … not just simulation)
    - Care in building to mimic reality faithfully
    - Level of detail
    - Get same conclusions from model as you would from system
    - More in Chapter 13

Types of Models

- ***Physical*** **(*iconic*) models**
  - Tabletop material-handling models
  - Mock-ups of fast-food restaurants
  - Flight simulators
- ***Logical*** **(*mathematical*) models**
  - Approximations, assumptions about system's operation
  - Often represented via computer program in appropriate software
  - Exercise program to try things, get results, learn about model behavior

Studying Logical Models

- **If model is simple enough, use traditional mathematical analysis … get exact results, lots of insight into model**
  - Queueing theory
  - Differential equations
  - Linear programming
- **But complex systems can seldom be *validly* represented by simple analytic model**
  - Danger of over-simplifying assumptions … model validity?
  - Type III error – working on the wrong problem
- **Often, complex system requires complex model, analytical methods don't apply … what to do?**

Computer Simulation

- **Methods for studying wide variety of models of systems**
  - Numerically evaluate on computer
  - Use software to imitate system's operations, characteristics, often over time
- **Can use to study simple models, but should not use if an analytical solution is available**
- **Real power of simulation – studying complex models**
- **Simulation can tolerate complex models since we don't even aspire to an analytical solution**

Popularity of Simulation

- **Has been consistently ranked as the most useful, popular tool in broader area of operations research / management science**
  - 1978: M.S. graduates of CWRU O.R. Department … after graduation
    1. Statistical analysis
    2. Forecasting
    3. Systems Analysis
    4. Information systems
    5. Simulation
  - 1979: Survey 137 large firms, which methods used?
    1. Statistical analysis (93% used it)
    2. Simulation (84%)
    3. Followed by LP, PERT/CPM, inventory theory, NLP, …

Popularity of Simulation (cont'd.)

  - 1980: (A)IIE O.R. division members
    - First in utility and interest — simulation
    - First in familiarity — LP (simulation was second)
  - 1983, 1989, 1993: Longitudinal study of corporate practice
    1. Statistical analysis
    2. Simulation
  - 1989: Survey of surveys
    - Heavy use of simulation consistently reported
  - 2012 (Powers thesis): Literally exponential growth in number of simulation papers
- **Since most of these surveys, hardware/software have improved, making simulation even more attractive**
  - Historical impediment to simulation – computer speed

Advantages of Simulation

- **Flexibility to model things as they are (even if messy and complicated)**
  - Avoid *looking where the light is* (a morality play):

    You're walking along in the dark and see someone on hands and knees searching the ground under a street light.

You:	"What's wrong? Can I help you?"
Other person:	"I dropped my car keys and can't find them."
You:	"Oh, so you dropped them around here, huh?"
Other person:	"No, I dropped them over there." (Points into the darkness.)
You:	"Then why are you looking here?"
Other person:	"Because this is where the light is."

- **Allows uncertainty, nonstationarity in modeling**
  - The only thing that's for sure: nothing is for sure
  - Danger of ignoring system variability
  - Model validity

Advantages of Simulation (cont'd.)

- **Advances in computing/cost ratios**
  - Estimated that 75% of computing power is used for various kinds of simulations
  - Dedicated machines (e.g., real-time shop-floor control)
- **Advances in simulation software**
  - Far easier to use (GUIs)
  - No longer as restrictive in modeling constructs (hierarchical, down to C)
  - Statistical design & analysis capabilities

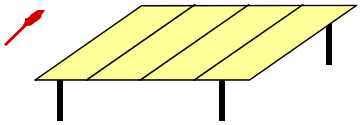
The Bad News

- **Don't get exact answers, only approximations, estimates**
  - Also true of many other modern methods
  - Can bound errors by machine roundoff
- **Get random output (*RIRO*) from stochastic simulations**
  - Statistical design, analysis of simulation experiments
  - Exploit: noise control, replicability, sequential sampling, variance-reduction techniques
  - Catch: "standard" statistical methods seldom work

Different Kinds of Simulation

- **Static vs. *Dynamic***
  - Does time have a role in model?
- **Continuous-change vs. *Discrete-change***
  - Can "state" change continuously, or only at discrete points in time?
- **Deterministic vs. *Stochastic***
  - Is everything for sure or is there uncertainty?
- **Most operational models:**
  - *Dynamic*, *Discrete-change*, *Stochastic*
    - But Chapter 2 discusses one static model
    - And Chapter 11 discusses continuous and combined discrete-continuous models

Simulation by Hand: The Buffon Needle Problem



- Estimate π (George Louis Leclerc, c. 1733)
- Toss needle of length *l* onto table with stripes *d* (>*l*) apart
- *P* (needle crosses a line) = $\frac{2l}{\pi d}$
- Repeat; tally $\hat{p}$ = proportion of times a line is crossed
- Estimate π by $\frac{2l}{\hat{p}d}$

Just for fun:
http://www.mste.uiuc.edu/reese/buffon/bufjava.html
http://www.angelfire.com/wa/hurben/buff.html

Why Toss Needles?

- **Buffon needle problem seems silly now, but has important simulation features:**
  - Experiment to *estimate* something hard to compute exactly (in 1733)
  - *Randomness*, so estimate will not be exact; estimate the error in the estimate
  - *Replication* (the more the better) to reduce error
  - *Sequential sampling* to control error — keep tossing until probable error in estimate is "small enough"
  - *Variance reduction* (*Buffon Cross*)

Using Computers to Simulate

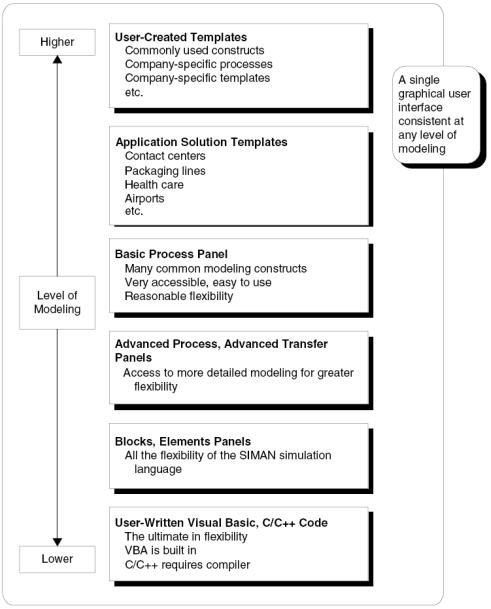
- **General-purpose languages (C, C++, C#, Java, Matlab, FORTRAN, others)**
  - Tedious, low-level, error-prone
  - But, almost complete flexibility
- **Support packages for general-purpose languages**
  - Subroutines for list processing, bookkeeping, time advance
  - Widely distributed, widely modified
- **Spreadsheets**
  - Usually static models (only *very* simple dynamic models)
  - Financial scenarios, distribution sampling, SQC
  - Examples in Chapter 2 (one static, one dynamic)
  - Add-ins are available (@RISK, Crystal Ball)

Using Computers to Simulate (cont'd.)

- **Simulation languages**
  - GPSS, SLX, SIMAN (on which Arena is based, included in Arena)
  - Popular, some still in use
  - Learning curve for features, effective use, syntax
- **High-level simulators**
  - Very easy, graphical interface
  - Domain-restricted (manufacturing, communications)
  - Limited flexibility — model validity?

Where Arena Fits In

- **Hierarchical structure**
  - Multiple levels of modeling
  - Mix different modeling levels together in same model
  - Often, start high then go lower as needed
- **Get ease-of-use advantage of simulators without sacrificing modeling flexibility**



When Simulations are Used

- **Use of simulation has evolved with hardware, software**
- **Early years (1950s – 1960s)**
  - Very expensive, specialized tool
  - Required big computers, special training
  - Mostly in FORTRAN (or even Assembler)
  - Processing cost as high as $1000/hour for a sub-PC level machine

When Simulations are Used (cont'd.)

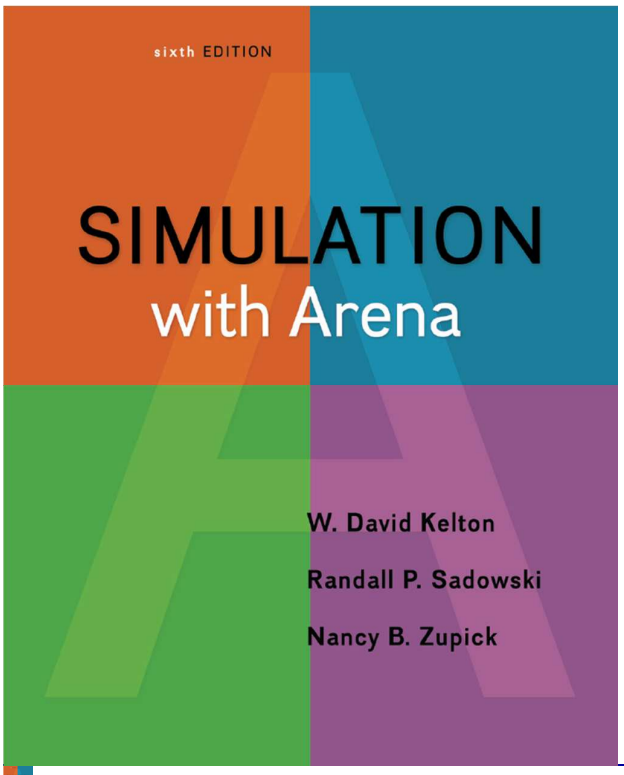
- **Formative years (1970s – early 1980s)**
  - Computers got faster, cheaper
  - Value of simulation more widely recognized
  - Simulation software improved, but still languages to be learned, typed, batch processed
  - Often used to clean up "disasters" in auto, aerospace industries
    - Car plant; heavy demand for certain model
    - Line underperforming
    - Simulated, problem identified
    - But demand had dried up — simulation was too late

When Simulations are Used (cont'd.)

- **Recent past (late 1980s – mid 2000s)**
  - Microcomputer power
  - Software expanded into GUIs, animation
  - Wider acceptance across more areas
    - Traditional manufacturing applications
    - Services
    - Health care
    - "Business processes"
  - Still mostly in large firms
  - Simulation is often part of "specs"

When Simulations are Used (cont'd.)

- **Present**
  - Proliferating into smaller firms
  - Becoming a standard tool
  - Being used earlier in design phase
  - Real-time control
  - 3D graphics, business dashboards
- **Future**
  - Integration with other applications for visualization, analysis
  - Networked sharing of data in real time
  - Internet-enabled distributed model building, execution
  - Specialized vertical "templates" for specific industries, firms
  - Better model re-usability, operational decision making
  - Automated statistical design, analysis



Fundamental Simulation Concepts

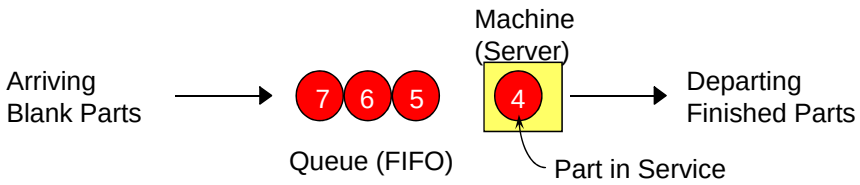
Chapter 2

Last revision March 9, 2014

What We'll Do ...

- **Underlying ideas, methods, and issues in simulation**
- **Software-independent (setting up for Arena)**
- **Example of a simple processing system**
  - Decompose problem
  - Terminology
  - Simulation by hand
  - Some basic statistical issues
- **Spreadsheet simulation**
  - Simple static, dynamic models
- **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 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 Study: Output Performance Measures

- ***Total production*** **of parts over 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 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 Study: Output Performance Measures (cont'd.)

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

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

- **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 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 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 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

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

  - 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 right "times," in 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 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 types of entities is first thing to do in building 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 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 whole model
  - Not tied to entities
  - Entities can access, change variables
  - Writing on 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 simulation

Pieces of a Simulation Model (cont'd.)

- ***Queues***
  - Place for entities to wait when they can't move on (maybe since 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 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 simulation
  - At end of simulation, used to compute final output performance measures

Pieces of a Simulation Model (cont'd.)

- **Statistical accumulators for simple processing system**
  - Number of parts produced so far
  - Total of waiting times spent in queue so far
  - No. of parts that have gone through 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: 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++, Java, Matlab, FORTRAN, etc.)**

Events for the Simple Processing System

- ***Arrival*** **of a new part to 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 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 machine idle (Note: there will be no departure event scheduled on future events calendar, which is as desired)

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

- ***The End***
  - Update time-persistent statistics (to end of simulation)
  - Compute final output performance measures using current (= final) values of statistical accumulators
- **After each event, 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

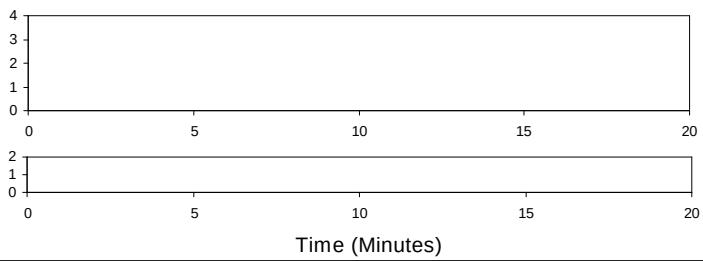
- **Simulation clock variable (internal in Arena)**
- **Event calendar: list of event *records*:**
  - [Entity No., Event Time, Event Type]
  - Keep *ranked* in increasing order on Event Time
  - Next event always in top record
  - Initially, schedule first Arrival, The End (Dep.?)
- **State variables: 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 next
- Will omit times in system, “max” computations here (see text for complete details)

Simulation by Hand: Setup

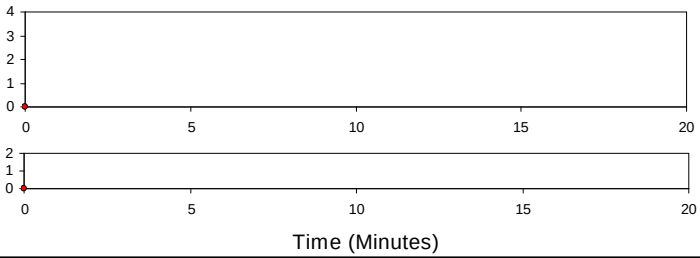
<table>
<tr><td>System  </td><td>Clock</td><td>B(t)</td><td>Q(t)</td><td>Arrival times of custs. in queue</td><td>Event calendar</td></tr>
<tr><td>Number of completed waiting times in queue</td><td colspan="2">Total of waiting times in queue</td><td colspan="2">Area under Q(t)</td><td>Area under B(t)</td></tr>
<tr><td>Q(t) graph

B(t) graph</td><td colspan="5"> 
</td></tr>
<tr><td>Interarrival times</td><td colspan="5">1.73, 1.35, 0.71, 0.62, 14.28, 0.70, 15.52, 3.15, 1.76, 1.00, ...</td></tr>
<tr><td>Service times</td><td colspan="5">2.90, 1.76, 3.39, 4.52, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...</td></tr>
</table>

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]

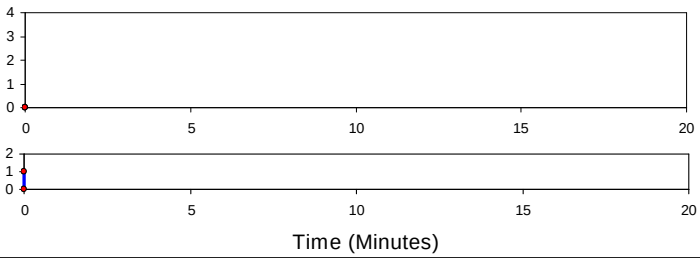
Number of completed waiting times in queue	Total of waiting times in queue	Area under *Q*(*t*)	Area under *B*(*t*)
0	0.00	0.00	0.00

Q(*t*) graph *B*(*t*) graph	
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
	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	
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, ...

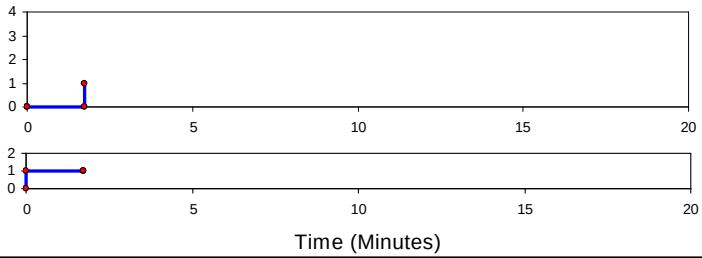
Simulation by Hand: *t* = 1.73, Arrival of Part 2

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



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, ...

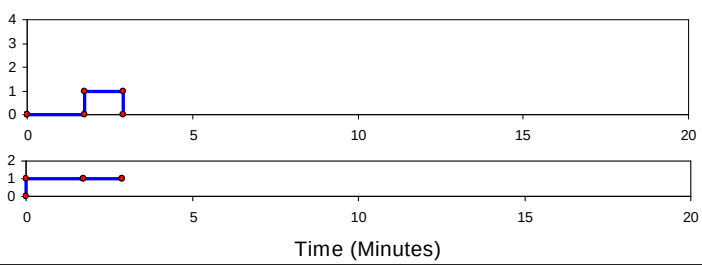
Simulation by Hand: *t* = 2.90, Departure of Part 1

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, ...

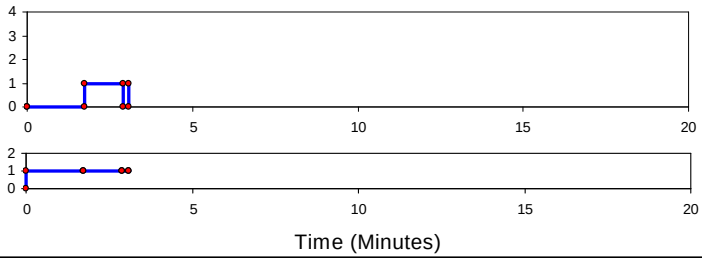
Simulation by Hand: *t* = 3.08, Arrival of Part 3

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, ...

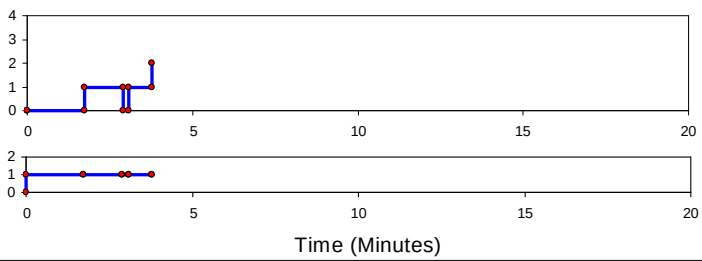
Simulation by Hand: *t* = 3.79, Arrival of Part 4

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, ...

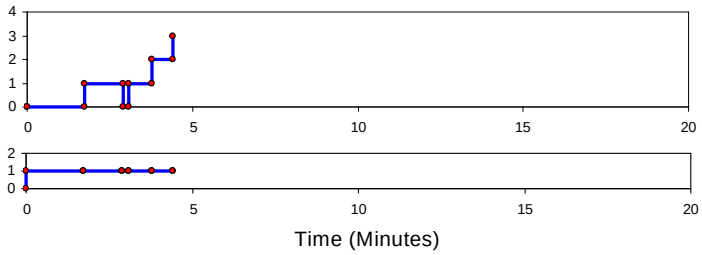
Simulation by Hand: *t* = 4.41, Arrival of Part 5

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~~, ~~0.62~~, ~~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, ...

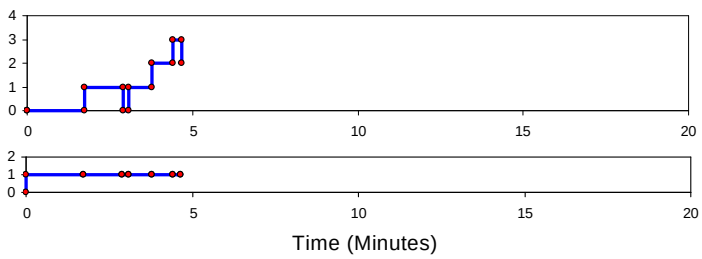
Simulation by Hand: *t* = 4.66, Departure of Part 2

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~~, ~~0.62~~, ~~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, ...

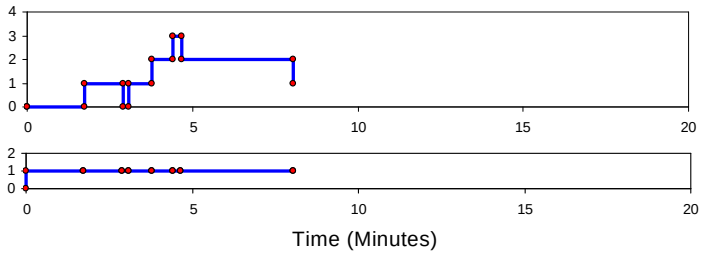
Simulation by Hand: *t* = 8.05, Departure of Part 3

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~~, ~~0.62~~, ~~14.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.00~~, ~~1.76~~, ~~3.36~~, ~~4.52~~, 4.46, 4.36, 2.07, 3.36, 2.37, 5.38, ...

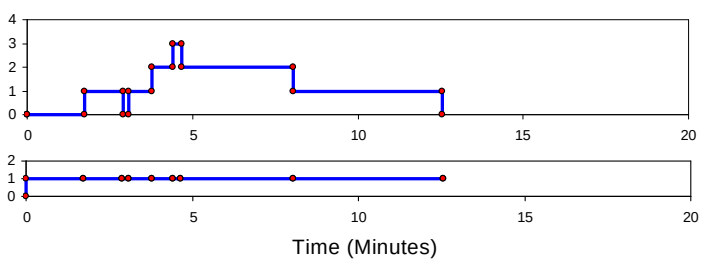
Simulation by Hand: *t* = 12.57, Departure of Part 4

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	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.62~~, ~~14.28~~, 0.70, 15.52, 3.15, 1.76, 1.00, ...
Service times	~~2.00~~, ~~1.76~~, ~~3.36~~, ~~4.52~~, ~~4.46~~, 4.36, 2.07, 3.36, 2.37, 5.38, ...

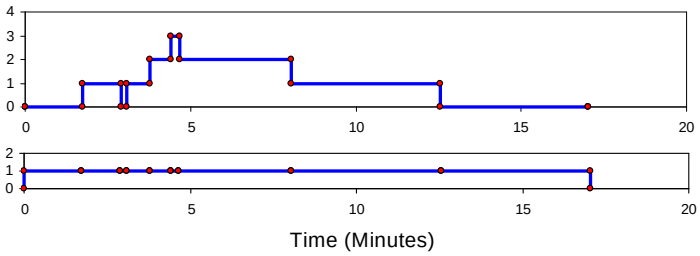
Simulation by Hand: *t* = 17.03, Departure of Part 5

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	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.02~~, ~~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, ...

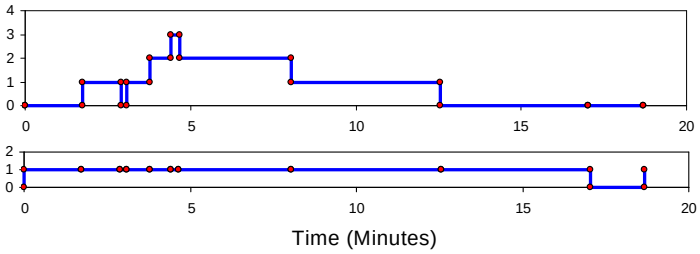
Simulation by Hand: *t* = 18.69, Arrival of Part 6

System	Clock	*B*(*t*)	*Q*(*t*)	Arrival times of custs. in queue	Event calendar
	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	~~1.73~~, ~~1.35~~, ~~0.71~~, ~~0.02~~, ~~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, ...

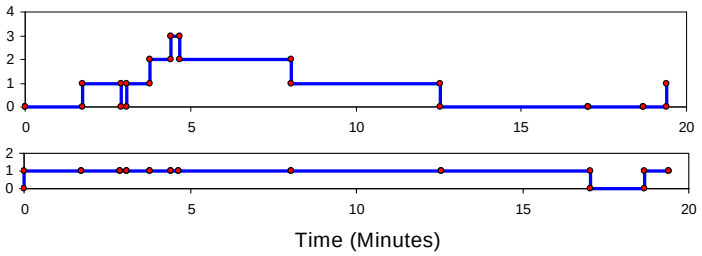
Simulation by Hand: *t* = 19.39, Arrival of Part 7

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	~~2.90~~, ~~1.76~~, ~~3.39~~, ~~4.52~~, ~~4.46~~, ~~4.36~~, 2.07, 3.36, 2.37, 5.38, ...

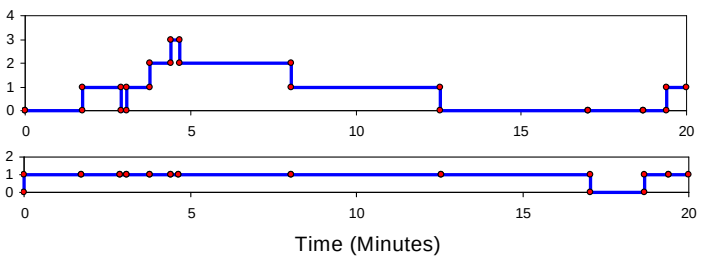
Simulation by Hand: *t* = 20.00, The End

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	~~2.90~~, ~~1.76~~, ~~3.39~~, ~~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

Just-Finished Event			Variables		Attributes		Statistical Accumulators								Event Calendar
Entity No.	Time t	Event Type	$Q(t)$	$B(t)$	Arrival Times: (In Queue)	In Service	P	N	ΣWQ	WQ^*	ΣTS	TS^*	$\int Q$	Q^*	$\int B$	[Entity No., Time, Type]
–	0.00	Init	0	0	( )	–	0	0	0.00	0.00	0.00	0.00	0.00	0	0.00	[1, 0.00, Arr] [–, 20.00, End]
1	0.00	Arr	0	1	( )	0.00	0	1	0.00	0.00	0.00	0.00	0.00	0	0.00	[2, 1.73, Arr] [1, 2.90, Dep] [–, 20.00, End]
2	1.73	Arr	1	1	(1.73)	0.00	0	1	0.00	0.00	0.00	0.00	0.00	1	1.73	[1, 2.90, Dep] [3, 3.08, Arr] [–, 20.00 End]
1	2.90	Dep	0	1	( )	1.73	1	2	1.17	1.17	2.90	2.90	1.17	1	2.90	[3, 3.08, Arr] [2, 4.66, Dep] [–, 20.00, End]
3	3.08	Arr	1	1	(3.08)	1.73	1	2	1.17	1.17	2.90	2.90	1.17	1	3.08	[4, 3.79, Arr] [2, 4.66, Dep] [–, 20.00, End]
4	3.79	Arr	2	1	(3.79, 3.08)	1.73	1	2	1.17	1.17	2.90	2.90	1.88	2	3.79	[5, 4.41, Arr] [2, 4.66, Dep] [–, 20.00, End]
5	4.41	Arr	3	1	(4.41, 3.79, 3.08)	1.73	1	2	1.17	1.17	2.90	2.90	3.12	3	4.41	[2, 4.66, Dep] [6, 18.69, Arr] [–, 20.00, End]
2	4.66	Dep	2	1	(4.41, 3.79)	3.08	2	3	2.75	1.58	5.83	2.93	3.87	3	4.66	[3, 8.05, Dep] [6, 18.69, Arr] [–, 20.00, End]
3	8.05	Dep	1	1	(4.41)	3.79	3	4	7.01	4.26	10.80	4.97	10.65	3	8.05	[4, 12.57, Dep] [6, 18.69, Arr] [–, 20.00, End]
4	12.57	Dep	0	1	( )	4.41	4	5	15.17	8.16	19.58	8.78	15.17	3	12.57	[5, 17.03, Dep] [6, 18.69, Arr] [–, 20.00, End]
5	17.03	Dep	0	0	( )	–	5	5	15.17	8.16	32.20	12.62	15.17	3	17.03	[6, 18.69, Arr] [–, 20.00 End]
6	18.69	Arr	0	1	( )	18.69	5	6	15.17	8.16	32.20	12.62	15.17	3	17.03	[7, 19.39, Arr] [–, 20.00, End] [6, 23.05, Dep]
7	19.39	Arr	1	1	(19.39)	18.69	5	6	15.17	8.16	32.20	12.62	15.17	3	17.73	[–, 20.00, End] [6, 23.05, Dep] [8, 34.91, Arr]
–	20.00	End	1	1	(19.39)	18.69	5	6	15.17	8.16	32.20	12.62	15.78	3	18.34	[6, 23.05, Dep] [8, 34.91, Arr]

Event-Scheduling Logic via Programming

- **Clearly well suited to standard programming language (C, C++, Java, etc.)**
- **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: Process-Interaction World View

- **Identify characteristic *entities* in 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
- **View normally taken by Arena**
  - Arena translates your model description into a program in SIMAN simulation language for execution

Randomness in Simulation

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

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

Substantial variability across replications

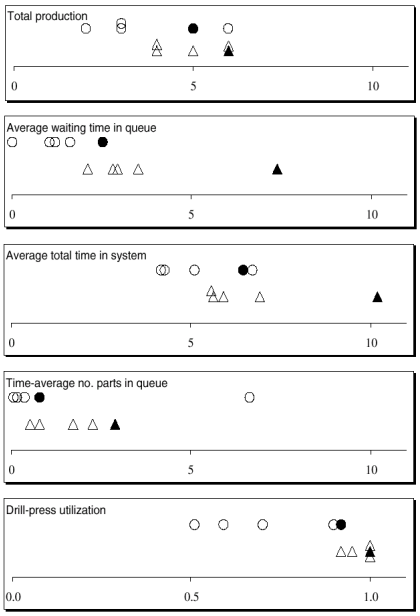
- **Confidence intervals for expected values:**
  - In general, $\bar{X} \pm t_{n-1,1-\alpha/2} s/\sqrt{n}$ (normality assumption?)
  - For expected total production, $3.80 \pm (2.776)(1.64/\sqrt{5})$
    3.80 ± 2.04

Precision?

Comparing Alternatives

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

Simulating with Spreadsheets: Introduction

- Popular, ubiquitous tool
- Can use for simple simulation models
  - Typically, only static models
    - Risk analysis, financial/investment scenarios
  - Only (very) simplest of dynamic models
- Two examples
  - Newsvendor problem (static)
  - Waiting times in single-server queue (dynamic)
    - Special recursion valid only in this case

Simulating with Spreadsheets: Newsvendor Problem – Setup

- **Rupert sells daily newspapers on street**
  - Rupert buys for c = \$0.55 each, sells for r = \$1.00 each
- **Each morning, Rupert buys q copies**
  - q is a fixed number, same every day
- **Demand during a day: $D = \max(\lfloor X \rceil, 0)$**
  - $X \sim$ normal (μ = 135.7, σ = 27.1), from historical data
  - $\lfloor X \rceil$ rounds X to nearest integer
- **If $D \leq q$, satisfy all demand, and $q - D \geq 0$ left over, sell for scrap at s = \$0.03 each**
- **If $D > q$, sells out (sells all q copies), no scrap**
  - But missed out on $D - q > 0$ sales
- **What should q be?**

Simulating with Spreadsheets: Newsvendor Problem – Formulation

- **Choose q to maximize expected profit per day**
  - q too small – sell out, miss \$0.45 profit per paper
  - q too big – have left over, scrap at a loss of \$0.52 per paper
- **Classic operations-research problem**
  - Many versions, variants, extensions, applications
  - Much research on exact solution in certain cases
  - But easy to simulate, even in a spreadsheet
- **Profit in a day, as a function of q:**

$$W(q) = \underbrace{r \min(D, q)}_{\text{Sales revenue}} + \underbrace{s \max(q - D, 0)}_{\text{Scrap revenue}} - \underbrace{cq}_{\text{Cost}}$$

  - $W(q)$ is a random variable – profit varies from day to day
- **Maximize E($W(q)$) over nonnegative integers q**

Simulating with Spreadsheets: Newsvendor Problem – Simulation

- **Set trial value of *q*, generate demand *D*, compute profit for that day**
  - Then repeat this for many days independently, average to estimate E($W(q)$)
    - Also get confidence interval, estimate of P(loss), histogram of $W(q)$
  - Try for a range of values of q
- **Need to generate demand $D = \max(\lfloor X \rceil, 0)$**
  - So need to generate $X \sim$ normal ($\mu = 135.7$, $\sigma = 27.1$)
  - (Much) ahead – Sec. 12.2, generating random *variates*
  - In this case, generate $X = \Phi^{-1}_{\mu,\sigma}(U)$
    U is a random number distributed uniformly on [0, 1] (Sec. 12.1)
    $\Phi_{\mu,\sigma}$ is cumulative distribution function of normal (μ, σ) distribution

Simulating with Spreadsheets: Newsvendor Problem – Excel

- **File `Newsvendor.xls`**
- **Input parameters in cells B4 – B8 (blue)**
- **Trial values for *q* in row 2 (pink)**
- **Day number (1, 2, ..., 30) in column D**
- **Demands in column E for each day:**

All files in book: www.mhhe.com/kelton, Student Edition, **`BookExamples.zip`**

```
= MAX(ROUND(NORMINV(RAND(), $B$7, $B$8), 0), 0)
```

Rounding function — Φ^{-1} — *U(0, 1) random number* — μ — σ

X ~ normal (μ, σ)

`RAND()` *is "volatile" so regenerates on any edit, or F9 key*

`$` *pins down following column or row when copying formula*

Round to nearest integer

MAX 2nd argument

Simulating with Spreadsheets: Newsvendor Problem – Excel (cont'd.)

- **For each *q*:**
  - "Sold" column: number of papers sold that day
  - "Scrap" column: number of papers scrapped that day
  - "Profit" column: profit (+, –, 0) that day
  - Placement of "**$**" in formulas to facilitate copying
- **At bottom of "Profit" columns (green):**
  - Average profit over 30 days
  - Half-width of 95% confidence interval on E($W(q)$)
    - Value 2.045 is upper 0.975 critical point of *t* distribution with 29 d.f.
    - Plot confidence intervals as "I-beams" on left edge
  - Estimate of P($W(q) < 0$)
    - Uses **COUNTIF** function
- **Histograms of $W(q)$ at bottom**
  - Vertical red line at 0, separates profits, losses

Simulating with Spreadsheets: Newsvendor Problem – Results

- **Fine point – used same daily demands (column E) for each day, across all trial values of *q***
  - Would have been valid to generate them independently
  - Why is it better to use same demands for all *q*?
- **Results**
  - Best *q* is about 140, maybe a little less
  - Randomness in all results (tap F9 key)
    - All demands, profits, graphics change
    - Confidence-interval, histogram plots change
    - Reminder that these are random outputs, random plots
  - Higher $q \Rightarrow$ more variability in profit
    - Histograms at bottom are wider for larger *q*
    - Higher chance of both large profits, but higher chance of loss, too
    - Risk/return tradeoff can be quantified – risk taker vs. risk-averse

Simulating with Spreadsheets: Single-Server Queue – Setup

- **Like hand simulation, but:**
  - Interarrival times ~ exponential with mean $1/\lambda = 1.6$ min.
  - Service times ~ uniform on [a, b] = [0.27, 2.29] min.
  - Stop when 50th waiting time in queue is observed
    - i.e., when 50th customer *begins* service, not exits system
- **Watch waiting times in queue WQ_1, WQ_2, ..., WQ_{50}**
  - Important – not watching anything else, unlike before
- **S_i = service time of customer i,**
  A_i = interarrival time between custs. $i - 1$ and i
- ***Lindley's recursion* (1952): Initialize $WQ_1 = 0$,**

  $$WQ_i = \max(WQ_{i-1} + S_{i-1} - A_i, 0),\ i = 2, 3, \ldots$$

Simulating with Spreadsheets: Single-Server Queue – Simulation

- **Need to generate random variates: let $U \sim U[0, 1]$**
  - Exponential (mean $1/\lambda$): $A_i = -(1/\lambda) \ln(1 - U)$
  - Uniform on [a, b]: $S_i = a + (b - a)\,U$
- **File `MU1.xls`**

All files in book: www.mhhe.com/kelton, *Student Edition,* **`BookExamples.zip`**

- **Input parameters in cells B4 – B6 (blue)**
  - Some theoretical outputs in cells B8 – B10
- **Customer number (i = 1, 2, ..., 50) in column D**
- **Five IID replications (three columns for each)**
  - IA = interarrival times, S = service times
  - WQ = waiting times in queue (plot, thin curves)
    - First one initialized to 0, remainder use Lindley's recursion
      Curves rise from 0, variation increases toward right
    - Creates positive autocorrelation down WQ columns
      Curves have less abrupt jumps than if WQ_i's were independent

Simulating with Spreadsheets: Single-Server Queue – Results

- **Column averages (green)**
  - Average interarrival, service times close to expectations
  - Average WQ_i within each replication
    - Not too far from steady-state expectation
    - Considerable variation
    - Many are below it (why?)
- **Cross-replication (by customer) averages (green)**
  - Column T, thick line in plot to dampen noise
- **Why no sample variance, histograms of WQ_i's?**
  - Could have computed both, as in newsvendor; two issues:
    - Nonstationarity – what is a "typical" WQ_i here?
    - Autocorrelation – biases variance estimate, may bias histogram if run is not "long enough"

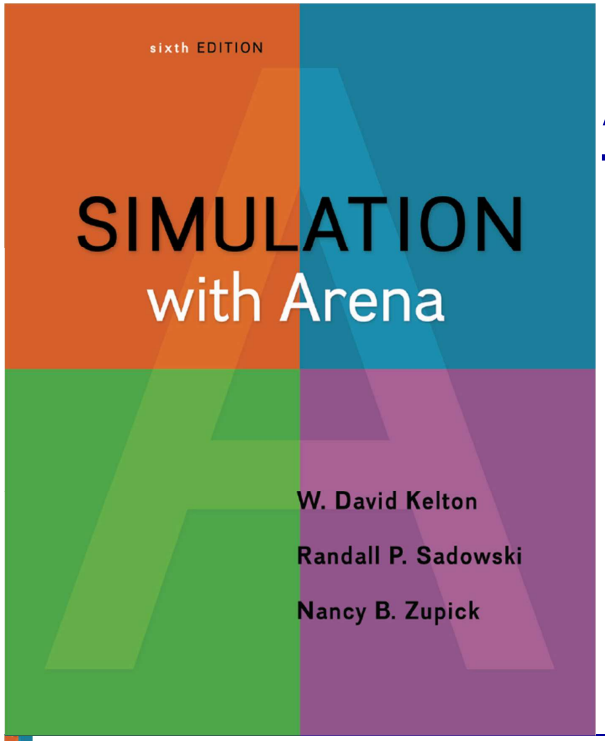
Simulating with Spreadsheets: Recap

- **Popular for static models**
  - Add-ins – @RISK, Crystal Ball
- **Inadequate tool for dynamic simulations if there's any complexity**
  - Extremely easy to simulate single-server queue in Arena – Chapter 3 main example
  - Can build very complex dynamic models with Arena – most of rest of book

Overview of a Simulation Study

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

More: Chapter 13



A Guided Tour Through Arena

Chapter 3

Last revision March 9, 2014

What We'll Do ...

- **Start Arena**
- **Load, explore, run an existing model**
  - Basically same as hand simulation in Chapter 2
  - Browse dialogs and menus
  - Run model
  - Look at results
- **Construct same model from scratch**
- **Use just these basic building blocks in case study to address real operational question**
- **Tour menus, toolbars, drawing, printing**
- **Help system**
- **Options for running and control**

Behavior of Arena

- **Arena is a true Windows application**
  - Appearance, operation, functions, are standard
  - Interoperability with other software (MS Office, CAD)
  - Interact, communicate with other software (Chapter 10)
- **Assume you already know basics of Windows:**
  - Disks, files, folders, paths
  - Mousing, keyboarding
  - Resizing, moving, maximizing, minimizing windows
  - Menu operations
  - Ctrl, Alt, Shift keys
  - Cut, copy, paste
  - Filling out dialog fields

Starting Up

- **Installing Arena – Appendix D**
- **Locate icon or shortcut; double-click**
  - Or, *Start > All Programs > Rockwell Software > Arena > Arena*
  - Licensed Mode vs. Training/Evaluation Mode (STUDENT)
- **See File, View, Tools, Help menus**
  - Other menus present if a model file is open
- **Toolbars with buttons**
  - Unless a model file is open, only New model file, Open model file, Template Attach/Detach, Context Help (click it, then click on buttons or menu items)
- **Tooltips – roll over toolbar buttons for names**
- **Quitting Arena: *File > Exit*, Alt+F4, or top right ✕**

Opening an Existing Model

- ***File > Open* … or button**
  - Navigate to desired disk/directory
  - *Click > Open* or double-click **`Model 03-01.doe`**
  - Book example models: www.mhhe.com/kelton, Student Edition, **`BookExamples.zip`**, put where you want
  - More examples (typical location on Windows 7): **`C:\Users\Public\Public Documents\Rockwell Software\Arena\Examples`**
- **Model window (right side of Arena window)**
  - Where model is built
  - Resize, maximize, minimize, scroll/pan, zoom
  - Can have multiple model windows open at once
- **Cut, Copy, Paste within Arena, and between Arena and other applications (when sensible)**

Why the **`.doe`** *default filename extension for Arena models?*

Flowchart and Spreadsheet Views

- **Model window split into two views**
  - *Flowchart* view
    - Graphics
    - Process flowchart
    - Animation, drawing
    - Edit things by double-clicking on them, get into a dialog
  - *Spreadsheet* view
    - Displays model data directly
    - Can edit, add, delete data in spreadsheet view
    - Displays all similar kinds of modeling elements at once
  - Many model parameters can be edited in either view
  - Horizontal splitter bar to apportion two views
  - *View > Split Screen* (or push) to see both flowchart and spreadsheet views (otherwise, only get view for active module type)

Project Bar

- **Usually down left edge of Arena window**
- **Hosts panels with modeling building blocks: *modules***
  - Both flowchart and spreadsheet modules
- **Displays one panel at a time**
  - Switch to different panels via horizontal buttons
  - Panels for Basic Process, Reports (after running), Navigate (to different views within a model or to different hierarchical submodels, thumbnail), … others can be attached (Template Attach button) for different modeling levels, specialties
- **Usually docked to left edge but can move, float**
- **Hide it via *View > Project Bar* or its own small**

Status Bar

- **At very bottom of Arena window**
- **Displays various information sensitive to status**
  - Coordinates of cursor in “worldspace”
  - When simulation is running:
    - Simulation clock value
    - Replication number being executed
    - Number of replications to be done
- **Hide by clearing (unchecking) *View > Status Bar***

Moving Around, Up, Down in Flowchart View of Model Window

- **Underlying *world space* for model**
  - (*x*, *y*) coordinates, arbitrary units (±32K in all directions)
- ***Pan* with scroll bars, arrow keys, thumbnail**
- ***Zoom* in (down): or + key or thumbnail**
- **Zoom out (up): or – key or thumbnail**
- **See all at min altitude: or * key**
- ***Named views***
  - Save a pan/zoom view for different parts of model
  - Assign a *Hot key* (case-sensitive)
  - Access via *View* > *Named Views …* or ? key or

To navigate via keyboard, flowchart view of model window must be active ... click in it.

- **Display *grid* ( ), *snap* to grid ( ) toggles**
- **Rulers, alignment, guides, glue – see text**

Modules

- **Basic building blocks of simulation model**
- **Two basic types: *flowchart* and *data***
- **Different types of modules for different actions, specifications**
- **"Blank" modules: on Project Bar**
  - Add a flowchart module to model: drag it from Project Bar into flowchart view of model window
    - Can have many instances of same kind of flowchart module in model
  - Use a data module: select it (single-click) in Project Bar, edit in spreadsheet view of model window
    - Only one instance of each kind of data module in model, but it can have many entries (rows) in spreadsheet view
    - Can edit via dialog – double-click on number in leftmost column

Flowchart Modules

- **Describe dynamic processes**
  - Nodes/places through which entities flow
  - Typically connected to each other in some way
- **Basic Process panel flowchart module types:**
  - Create, Dispose, Process, Decide, Batch, Separate, Assign, Record
- **Other panels – many other kinds**
- **Shape like flowcharting (later, colors for hints)**
- **Two ways to edit**
  - Double-click to open up, then fill out dialogs
  - Select (single-click) a module type in model or Project Bar, get all modules of that type in spreadsheet view

Data Modules

- **Set values, conditions, etc. for whole model**
  - No entity flow, no connections
- **Basic Process panel data module types:**
  - Attribute, Entity, Queue, Resource, Variable, Schedule, Set
- **Other panels – many other kinds**
- **Icons in Project Bar look like little spreadsheets**
- **To use a data module, select it (single-click) in Project Bar, edit in spreadsheet view**
  - Can edit via dialog – double-click in leftmost column, or right-click and select Edit via Dialog
  - Double-click where indicated to add new row
  - Right-click on row, column to do different things
- **At most one instance of each kind of data module in a model**
  - But each one can have many entries (rows)

Relations Among Modules

- **Flowchart, data modules related via names for objects**
  - Queues, Resources, Entity types, Variables, Expressions, Sets, … many others
- **Arena keeps internal lists of different kinds of names**
  - Presents existing lists to you where appropriate
  - Helps you remember names, protects you from typos
- **All names you make up in a model must be unique across model, even across different types of modules**

Internal Model Documentation

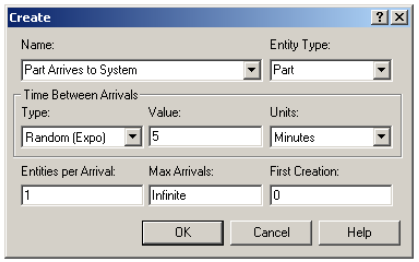
- ***Data Tips* on modules, graphics – hover mouse over object to see**
  - Default part – generic info on object (name, type)
  - User-specified part – right-click on object, select Properties, enter text under Description
  - Toggle display of Data tips via *View > Data Tips*
- ***Project Description* – *Run > Setup > Project Parameters,* enter text under Project Description**
- ***Model Documentation Report* – *Tools > Model Documentation Report***
  - Generates HTML file with model details (can choose which kinds of details to include)

Browsing Through Model 3-1

- **Open `Model 03-01.doe`, Book Examples folder**
  - www.mhhe.com/kelton, Student Edition, **`BookExamples.zip`**, unzip and put folder where you want on your system
- **Three flowchart modules**
  - Create, Process, Dispose
- **Entries in three data modules**
  - Entity, Queue, Resource
- **Animation objects**
  - Resource animation
  - Two plots
  - Some (passive) labels, “art” work

Create Flowchart Module

- **“Birth” node for entities**
- **Gave this instance of Create-type module the Name `Part Arrives to System`**
  - If we had other Create modules (we don’t) they’d all have different Names
- **Double-click on module to open property dialog:**



Create Flowchart Module (cont'd.)

- **Name – for module (type it in, overriding default)**
- **Entity Type – enter descriptive name**
  - Can have multiple Entity Types with distinct names
- **Time Between Arrivals area**
  - Specify nature of time separating consecutive arrivals
  - Type – pull-down list, several options
  - Value – depends on Type … for Random (Expo) is mean
  - Units – time units for Value
- **Entities per Arrival – constant, random variable, very general "Expression" (more later …)**
- **Max Arrivals – choke off arrivals (from here) after this many arrivals (batches, not entities)**
- **First Creation – time of first arrival (need not be 0)**

Editing Flowchart Modules in Spreadsheet View

- **Alternative to dialog for each instance of a module type**
- **See all instances of a module type at once**
  - Convenient for seeing, editing many things at once
- **Selecting a module in either flowchart or spreadsheet view also selects it in the other view**
- **Click, double-click fields to view, edit**
- **Right-click in row to Edit via Dialog, define user Data Tip (via Properties)**
- **Right-click in expression fields to get Expression Builder for help in constructing complex expressions with Arena variables (more later …)**

Entity Data Module

- **A data module, so edit in spreadsheet view only**
- **View, edit aspects of different entity Types in your model (we have just one entity Type, `Part`)**
- **Pull-down lists activated as you select fields**
- **Our only edit – Initial Picture for animation**
  - Picked **`Picture.Blue Ball`** from default list
  - Menu option *Edit* > *Entity Pictures ...* to see, modify

Process Flowchart Module

- **Represents machine, including:**
  - Resource
  - Queue
  - Entity delay time (processing)
- **Enter Name – `Drilling Center`**
- **Type – picked `Standard` to define logic here rather than in a submodel (more later …)**
- **Report Statistics check box at bottom**
  - To get utilizations, queue lengths, queue waiting times, etc.

Process Flowchart Module (cont'd.)

- **Logic area – what happens to entities here**
  - Action
    - *Seize Delay Release* – entity Seizes some number of units of a Resource (maybe after a wait in queue), Delay itself there for processing time, then Release units of Resource it had Seized – chose this option
      - *Delay* entity (red traffic light) – no Resources or queueing, just sit here for a time duration
      - *Seize Delay* (no Release … presumably Release downstream)
      - *Delay Release* (if Resource had been Seized upstream)
    - Priority for seizing – lower numbers ⇒ higher priority
    - Different Action choices could allow stringing together several Process modules for modeling flexibility
  - Resources – define Resource(s) to be seized, released
    - Double-click on row to open subdialog
    - Define Resource Name, Quantity of units *to be Seized/Released here*
      - *Not* where you *say* there are multiple Resource units ... do that in Resource *data* module
    - Several Resources present (Add) – entities must first Seize all

Process Flowchart Module (cont'd.)

- **Delay Type – choice of probability distributions, constant or general Expression (more later …)**
- **Units – time units for delay (*don't ignore*)**
- **Allocation – how to "charge" delay in costing (more later …)**
- **Prompts on next line – change depending on choice of Delay Type – specify numerical parameters involved**
- **Can also edit in spreadsheet view**
  - Subdialogs (e.g., Resource here) become secondary spreadsheets that pop up, must be closed

Resource Data Module

- **Defining `Drill Press` Resource in Process module automatically creates entry (row) for it in Resource data module**
- **Can edit it here for more options**
  - Type – could vary capacity Based on Schedule instead of having a Fixed Capacity
    - Would define Schedule in Schedule data module … later
  - Capacity (if Type = Capacity) is number of units of this resource that exist
  - Failures – cause resource to fail according to some pattern
    - Define this pattern via Failure data module (Advanced Process panel) … later

Queue Data Module

- **Specify aspects of queues in model**
  - We only have one, named **`Drilling Center.Queue`** (default name, given Process module name)
- **Type – specifies *queue discipline* or ranking rule**
  - If Lowest or Highest Attribute Value, then another field appears where you specify which attribute to use
- **Shared – if this queue will be shared among several resources (later …)**
- **Report Statistics – check for automatic collection and reporting of queue length, time in queue**

Animating Resources and Queues

- **Got queue animation automatically by specifying a Seize in Process module**
  - Entity pictures (blue balls) line up here in animation
- **Don't get Resource animation automatically**
  - To add it, use Resource button in Animate toolbar … get Resource Picture Placement dialog
    - Identifier – link to Resource name in pull-down list
    - Specify different pictures for Idle, Busy states

      For pre-defined "art" work, Open a picture library (.plb filename extension)

      Scroll up/down on right, select (single-click) a picture on right, select Idle or Busy state on left, then to copy picture
  - To edit later, double-click on picture in flowchart view

Dispose Flowchart Module

- **Represents entities leaving model boundaries**
- **Name the module**
- **Decide on Record Entity Statistics (average, maximum time in system of entities exiting here, costing information)**

Check boxes for statistics collection and reporting:
- *Most are checked (turned on) by default*
- *Little or no modeling effort to say yes to these*
- *But in some models can slow execution markedly*
- *Moral – if you have speed problems, clear these if you don't care*

Connecting Flowchart Modules

- **Establish (fixed) sequence of flowchart modules through which entities flow**
- **To make a connection**
  - Connect (*Object* > *Connect*), cursor becomes cross hairs
  - Click on exit point ▶ from source module, then entry point ■ on destination module
    - Green, red boxes light up to aid in hitting exit, entry points
  - Intermediate clicks for non-straight line in segments
- **To make many connections**
  - After each connection, right-click in blank space, select Repeat Last Action from pop-up menu
  - Or, double-click on , place multiple connections (no right-click needed), right-click or Esc to end

Connecting Flowchart Modules (cont'd.)

- **Object menu toggles**
  - Auto-Connect – automatically connect entry point of newly placed module from exit point of selected module
  - Smart Connect – force segments to horizontal/vertical
    - Makes for a tidy-looking flowchart, but has the disadvantage that it can cause connection lines to be directly on top of each other, making it impossible to tell them apart
  - Animate Connectors – show entity moves along connectors (zero time for statistics collection), for verification
- **Move entry/exit points relative to their module**
  - Right-click on entry/exit point
  - Select Allow Move from pop-up
  - Drag entry/exit point around

Dynamic Plots

- **Trace variables (e.g., queue lengths) as simulation runs – "data animation"**
- **Disappear after run ends**
  - To keep, save data, postprocess in Output Analyzer ... later
- **Plot button  from Animate toolbar**
  - Six tabs across top; many options (best just to explore)
  - Data Series tab – click Add button for each curve to be plotted on same set of axes
    - In right "Properties" area, enter Name, define Expression

      Pull down Build Expression, "+" Basic Process Variables, "+" Queue, Current Number in Queue, select **`Drilling Center.Queue`** in Queue Name field pull-down, note Current Expression **`NQ(Drilling Center.Queue)`** automatically filled in at bottom, OK button to copy this expression back out
    - DrawMode – Stairs or PointToPoint
    - Line/fill color, vertical-axis on left/right

Note automatic context-sensitive mini Help window on right

Dynamic Plots (cont'd.)

- Axes tab – choose Time (X) Axis on left
  - X axis is always simulated time
  - Scale area on right ("+" to open it) – specify Min/Max, MajorIncrement, AutoScroll ("windows" axis during simulation)
  - Title on right – type in Text (mention units!), set Visible to True
- Axes tab – choose Left Value (Y) Axis on left
  - Note possibility for a different right Y axis scale for multiple curves
  - Scale area on right – specify Min/Max, MajorIncrement, usually leave AutoScaleMaximum at True so Y axis scale will automatically adjust to contain whole plot during run
  - Title on right
- Legend tab – clear Show Legend box since we have only one curve, and Y axis is labeled
- Other tabs – Titles, Areas, 3-D View ... just explore

- **Drop plot in via crosshairs (resize, move later)**

Dressing Things Up

- **Add drawing objects from Draw toolbar**
  - Similar to other drawing, CAD packages
  - Object-oriented drawing tools (layers, etc.), not just a paint tool
- **Add Text to annotate**
  - Control font, size, color, orientation

Setting Run Conditions

- ***Run* > *Setup* menu dialog – seven tabs**
  - Project Parameters – Title, Name, Project Description, stats
  - Replication Parameters
    - Number of Replications
    - Initialization options Between Replications
    - Start Date/Time to associate with start of simulation
    - Warm-up Period (when statistics are cleared)
    - Replication Length (and Time Units)
    - Hours per "Day" (convenience for 16-hour days, etc.)
    - Base Time Units (output measures, internal computations, units where not specified in dialog, e.g. Plot X Axis time units)
    - Terminating Condition (complex stopping rules)
  - Tabs for run speed, run control, reports, array sizes, visuals

Terminating your simulation:
- *You must specify – part of modeling*
- *Arena has no default termination*
- *If you don't specify termination, Arena will usually keep running forever*

Running It

- **Plain-vanilla run: Click ▶ from Standard toolbar (like audio/video players)**
  - First time or after changes: *Check*
  - Enters *run mode* — can move around but not edit
  - Speed up or slow down animation display via slider bar
    - Or tap > on keyboard to speed up, < to slow down
  - When done, asked if you want to see summary reports
  - Click ■ to get out of run mode (*can't edit until you do*)
  - Can *pause* run with ▮▮ or Esc key
- **Other run control, viewing, checking options**

Viewing Reports

- **Click Yes in Arena box at end of run**
  - Opens new reports window (separate from model window) inside Arena window
  - Project Bar shows Reports panel, different reports (each one would be a new window)
  - Remember to close all reports windows before future runs
- **Default installation shows Category Overview report – summarizes many things about run**
  - Reports have "pages" to browse (and)
  - Also, "table contents" tree at left for quick jumps via ⊞, ⊟
- **Times are in Base Time Units for model**

Viewing Reports – Examples

- **Entity → Time → Total Time → Part:**
  - Avg. time in system was 6.4397 min., max was 12.6185
- **Resource → Usage → Instantaneous Utilization → Drill Press:**
  - Utilization was 0.9171 (busy 91.71% of the time)
- **Process → Other → Number In → Drilling Center:**
  - During run, 7 parts entered **`Drilling Center`**
- **Process → Other → Number Out → Drilling Center:**
  - 5 entities left **`Drilling Center`** (so were produced)
- **Entity → Time → Wait Time → Part:**
  - Avg. wait time in all queues was 3.0340 min. (counts only entities that left the *system*, but Queue → Time → Waiting Time → Drilling Center.Queue counts all entities that left *this queue*, so these results can differ)
- **Entity → Other → Wip → Part:**
  - Average Work in Process was 1.7060, max WIP was 4

Types of Statistics Reported

- **Many output statistics are one of three types:**
  - *Tally* – avg., max, min of a discrete list of numbers
    - Used for discrete-time output processes like waiting times in queue, total times in system
  - *Time-persistent* – time-average, max, min of a plot of something where *x*-axis is continuous time
    - Used for continuous-time output processes like queue lengths, WIP, server-busy functions (for utilizations)
  - *Counter* – accumulated sums of something, usually just nose counts of how many times something happened
    - Often used to count entities passing through a point in model

More on Reports and their Files

- **Reports we just saw – based on MS Access `.mdb` database that Arena actually writes as it runs**
  - **`.mdb`** file is saved and can be viewed later
  - Viewing within Arena via SAP Crystal Reports to query Access database, produce reports like Category Overview
- **Arena also produces a plain-text summary report (`.out` filename extension)**
  - Previous versions of Arena, underlying SIMAN language
  - Fairly cryptic, but gives quick view of lots of output data
    - Also contains a few things not in Access/Crystal Reports
  - Multiple reports for multiple replications
- **"Half Width" columns – 95% confidence intervals on outputs in long-run simulations … later**

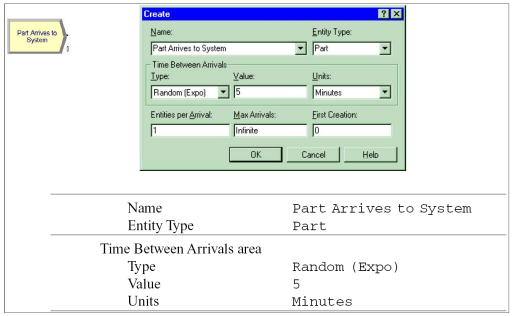
Build It Yourself

- **Build same model from scratch – details in text**
- **Handy user-interface tricks:**
  - Right-click in an empty spot in flowchart view – small box of options, including Repeat Last Action … useful in repetitive editing like placing lots of same module type
  - Ctrl+D or Ins key – duplicates whatever's selected in flowchart view, offsetting it a bit … drag elsewhere, edit
- **Open new (blank) model window – name it, save it, maybe maximize it**
- **Attach modeling panels you'll need to Project Bar if not there**

Build It Yourself (cont'd.)

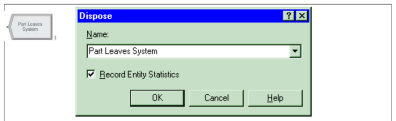
- **Place, connect flowchart modules**
- **Edit flowchart, data modules as needed**
  - Experiment with Expression Builder – right-click in expression field
- **Add plots, animation, artwork**
- **Add named views (? key or *View > Named Views* or), with hot key (case-sensitive)**
- **Edit *Run > Setup* dialog**
- **"Displays" in text**
  - Compact way of saying what needs to be done in a dialog
  - Omits Arena defaults
  - Shows completed dialogs, table of actions needed

"Displays" for Create, Process, Dispose Modules



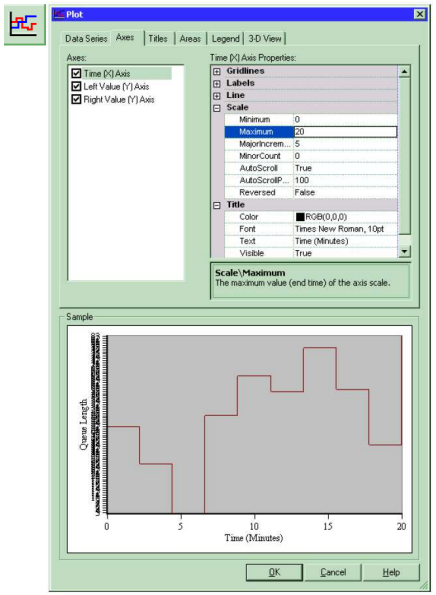
Name	Part Arrives to System
Entity Type	Part
Time Between Arrivals area	
Type	Random (Expo)
Value	5
Units	Minutes

Name	Drilling Center
Action	Seize Delay Release
Resources (secondary dialog via *Add* button)	
Type	Resource
Resource Name	Drill Press
Quantity	1
Delay Type	Triangular
Units	Minutes
Minimum	1
Value	3
Maximum	6



Name	Part Leaves System

"Display" for Queue-Length Plot



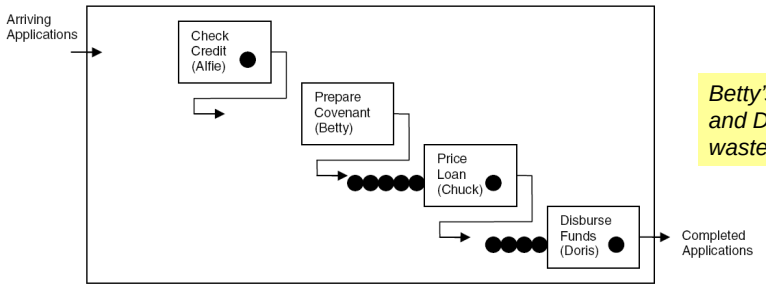
Data Series tab	
Add button	*click*
Series1 Properties > Source Data > Name	`Queue Length` (changes Series1 Properties above to `Queue Length Properties`)
Queue Length Properties:	
Source Data > Expression	`NQ(Drilling Center.Queue)` (or use Expression Builder)
Line > DrawMode	`Stairs` (select from pull-down)
Axes tab	
Axes > Time (X) Axis	*click* (on text to highlight, leave the box checked)
Time (X) Axis Properties:	
Scale > Maximum	`20`
Scale > MajorIncrement	`5`
Title > Text	`Time (Minutes)`
Title > Visible	`True` (select from pull-down)
Axes > Left Value (Y) Axis	*click* (on text to highlight, leave the box checked)
Left Value (Y) Axis Properties:	
Scale > MajorIncrement	`1`
Title > Text	`Queue Length`
Title > Visible	`True` (select from pull-down)
Legend tab	
Show Legend check box	*clear* (uncheck)

Axes tab showing here ...
Figure 3-14 in book shows Data Series tab

Case Study: Specialized Serial vs. Generalized Parallel Processing

- **Loan applications go through four steps**
  - Check credit, prepare covenant, price loan, disburse funds
  - Each step takes expo (1 hour)
  - Applications arrive with expo (1.25 hour) interarrival times
    - First application arrives at time 0
  - Run for 160 hours
  - Watch avg, max no. applications in process (WIP); avg, max total time in system of applications
  - Four employees, each can do any process step
- **Serial specialized processing or generalized parallel processing?**
  - What's the effect of service-time variability on decision?

Case Study: Model 3-2, Specialized Serial Processing



Betty's now idle but Chuck and Doris are overloaded – wasted capacity?

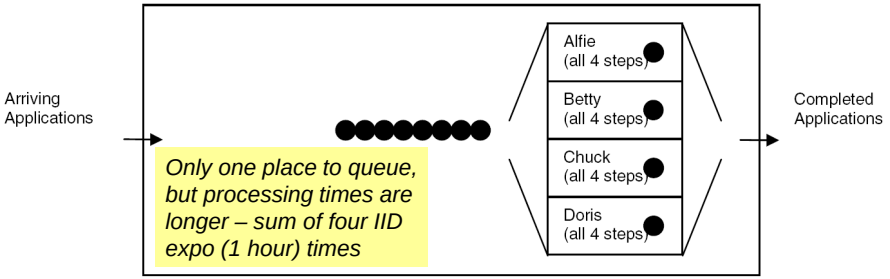
- **File `Model 03-02.doe`**
- **Create module – similar to Model 3-1 except expo mean, time units**
  - Set Entity Type to **`Application`**

All files in book: www.mhhe.com/kelton, Student Edition, **`BookExamples.zip`**

Case Study: Model 3-2, Specialized Serial Processing (cont'd.)

- **Four Process modules – similar to Model 3-1**
  - Four separate Resources
  - Expo process time: Expression (via Expression Builder)
- **Dispose module similar to Model 3-1**
- **Default entity picture (report) is OK**
- **Default Resource animations almost OK**
  - Make Idle picture same as Busy
  - Select correct Resource name in Identifier field
- **Queue, Resource data modules OK**
- **Plot WIP – use Expression builder to find `EntitiesWIP(Application)`**
  - Fixed Y axis max = 25 to compare with next three models
- **Fill in *Run* > *Setup*, lengthen queue animations**

Case Study: Model 3-3, Generalized Parallel Processing



- **File `Model 03-03.doe`**
- **Create, Dispose, plot, *Run* > *Setup* almost same**
  - Just change some labels, etc.

Case Study: Model 3-3, Generalized Parallel Processing (cont'd.)

- **Replace four earlier Process modules with just a single Process module**
  - One Resource (**`Loan Officer`**), but four units of it
  - Still set Quantity to 1 since application just needs 1 officer
  - Delay type – Expression

    `EXPO(1) + EXPO(1) + EXPO(1) + EXPO(1)`
    - Why not **`4*EXPO(1)`**?
- **Modify Resource Animation for four units**
  - Open Model 3-2 Resource Animation to get Resource Picture Placement window, open Idle picture
  - Duplicate white square three times, realign; copy to Busy
  - In model window, double-click Seize Area, then Add three
  - Still not completely accurate animation (order) – need Sets

Case Study: Compare Model 3-2 vs. 3-3

Model	Total WIP		Total Time in System		Total Waiting Time		Number Processed	Avg. Utilization
	Avg.	Max.	Avg.	Max.	Avg.	Max.		
3-2 (serial)	12.39	21	16.08	27.21	11.98	22.27	117	0.78
3-3 (parallel)	4.61	10	5.38	13.73	1.33	6.82	135	0.87

- ***Caution*: This is from only one replication of each configuration, so there's output variability**
  - Are differences statistically significant? (Exercise 6-19)

Case Study: Effect of Task-Time Variability

- **Is parallel always better than serial under any conditions?**
  - Many aspects could matter
  - Focus on task-time variability
- **Now, each task time ~ expo (1 hour)**
  - Highly variable distribution

    P(task time < 10 minutes) = 0.15
    P(task time > 2 hours) = 0.14
    } *See text*
  - In serial config., just one large task time congests greatly
    - In parallel config. it would also congest, but probably not by as much since other three tasks are probably not all large too
- **Other extreme – each task time is *exactly* 1 hour**
  - Leave interarrival times as expo (1.25 hours)
  - Models 3-4 (serial), 3-5 (parallel) – alter Process modules

Case Study: Effect of Task-Time Variability (cont'd.)

	Model	Total WIP		Total Time in System		Total Waiting Time		Number Processed	Avg. Utilization
		Avg.	Max.	Avg.	Max.	Avg.	Max.		
Expo service	3-2 (serial)	12.39	21	16.08	27.21	11.98	22.27	117	0.78
	3-3 (parallel)	4.61	10	5.38	13.73	1.33	6.82	135	0.87
Constant service	3-4 (serial)	3.49	12	5.32	11.38	1.32	7.38	102	0.65
	3-5 (parallel)	3.17	11	4.81	10.05	0.81	6.05	102	0.66

- **For constant service, parallel improvement appears minor**
  - Maybe not even statistically significant (Exercise 6-19)
- **Some further questions**
  - In parallel, work is integrated/generalized, so would it be slower per task? (Exercises 3-13, 6-20)
  - Effect of worker breaks? (Chapters 4, 5)
  - Differences statistically significant? (Exercises 6-19, 6-20)

More on Menus – File Menu

- **Model-file management**
- **Template attach/detach**
- **DXF import (from CAD packages), Visio import**
- **Color palettes**
- **Printing**
- **Send (e-mail) open model file**
- **Recent models**
- **Exit from Arena**

Edit Menu

- **Undo/Redo**
- **Cut/Copy/Paste**
- **Paste Link (create OLE link)**
- **Duplicate, Delete selection**
- **Select/Deselect All**
- **Entity Pictures – change content, definition of pictures presented in Entity data module**
- **Find – searches all modules, animation objects for a text string … useful for finding wrong names, typos after an error message from Arena**

Edit Menu (cont'd.)

- **Replace – replaces all instances of a text string with another text string**
- **Properties – display internal Arena object properties**
- **Links – to link to other files (spreadsheets, sounds, etc.)**
- **Insert New Object/Control – from other applications (e.g., graphics, VBA, ActiveX)**
- **Object – edit object imported from another application**

View Menu

- **Zooming – discussed before**
  - Zoom Factor – step size when zooming
- **Views – canned Arena views of flowchart view**
- **Named Views – define, change, use views**
- **Rulers, Grid, Guides, Snap, Glue – align objects**
  - Page breaks – shows page breaks if printed
- **Data Tips – toggles display of Data Tips**
- **Connector Arrows – show entity-flow direction**
- **Layers – which objects show up in which mode**

View Menu (cont'd.)

- **Split Screen – if checked, shows both flowchart, spreadsheet views**
- **Runtime Elements Bar – if checked, displays window allowing choice of what is displayed during execution**
- **Toolbars – decide which toolbars show up**
- **Project/Status Bar – toggle to show up or not**
- **Debug Bar – if checked, displays window of debugging tools during run**

Tools Menu

- **Arena NewsFlash – internet feed for updates, etc.**
- **Arena Symbol Factory – make animation symbols**
- **Separate applications for modeling, analysis**
  - Input Analyzer – fit probability distributions for input, using field-collected data … more in Chapt. 4
  - Process Analyzer – run, compare many "scenarios" at once … more in Chapt. 6
    - Also Output Analyzer … not on menus ... start from Start menu
  - Visual Designer for 3D animation, etc.
  - Expression Builder – very useful tool (described earlier)
- **ReportDatabase – export results to CSV file**
- **Model Documentation Report – generate HTML file with many details of this model**

Tools Menu (cont'd.)

- **Import/Export model to/from Database – bring in, save model details to Excel or Access**
- **OptQuest for Arena – separate application that "takes over" running of model to search for an optimal scenario … more in Chapt. 6**
- **AVI Capture – record actions (editing, animation) to .avi file for playback**
- **Macro – create Visual Basic macros (mini programs), VB editor ... more in Chapter 10**
- **Module count – reports module instances**
- **Options – control many aspects of how Arena works, looks**

Arrange Menu

- For modeling, graphics objects – first select object(s)
- Bring object to Front, Send to Back – “stacking”
- Group, Ungroup objects (move together, etc.)
- Flip around Vertical, Horizontal line
- Rotate object (90° clockwise)
- Align objects on top, bottom, left, or right edges
- Distribute objects evenly (horizontally, vertically)
- Flowchart Alignment – arrange flowchart modules (horizontally, vertically)
- Snap Object to Grid – for selected object(s)
- Change Object Snap Point on snapped object

Object Menu

- Connect tool – changes cursor to cross hairs
  - Hit twice for repeated connections, right-click or Esc to exit
- Auto-Connect new module to selected module
- Smart Connect – new connections in horizontal/vertical segments only
- Animate Connectors – show entities moving (at infinite speed for statistics collection)
- Animate At Desktop Color Depth – use desktop color depth (could slow run)
  - If not checked, color is 8-bit (256 colors), runs faster
- Submodel – define, manage hierarchical submodels, useful for large, complex models

Run Menu

- Setup – control model run conditions
- Entries to run, check, pause, step through
- Alternatives to watch execution, view results (or errors)
- Control how run goes and is displayed
- Most capabilities on Run Interaction Toolbar – details later
- Access “code” in underlying SIMAN simulation language

Window Menu

- Cascade, Tile multiple open model windows
- Arrange Icons for any minimized model windows
- Use system Background Color – use Windows colors rather than Arena settings
- List of open model windows

Help Menu

- One of several ways to get into Help system
- Arena Help – TOC, Index, Search
- What's This? – adds ? to cursor, then click on things for brief description
- Release notes – recent changes, requirements
- Arena Smart Files – subject-based index to many small but complete models that illustrate specific modeling techniques (*very* useful)
- List of attached modeling panels – select to get Help on that one

Help Menu (cont'd.)

- Arena Product Manuals – detailed PDF reference documents on Arena components
- Activation – for licensing
- Copy protection information for commercial, research, and lab versions
- About Arena... – version number, licensing information, etc.

More on Toolbars

- Collections of buttons for "frequent" operations
  - Most are duplication of menu entries
  - Standard, Draw, Animate, Integration, View, Arrange, Run Interaction, Record Macro, AVI Capture, Animate Transfer, Dialog Design, Project/Status/Debug Bars
- *View* > *Toolbars* (or right-click in a toolbar area) to decide which ones show up, which to hide
- Toolbars can be torn off ("floating" palettes), or "docked" to an edge of screen
- Arena remembers Toolbars for next time
- *View* > *Toolbars* > *Customize* to alter how toolbars and buttons are displayed
- See text for run-through description of toolbars and buttons (or, just experiment)

More on Drawing

- Draw via toolbar buttons only (no menus):

- Line, Polyline (Shift for 45°), Arc, Bézier Curve
- Box, Polygon, Ellipse (fill, line, shade)
- Text (font, size, style)
- Colors for Lines, Fill, Text, Window Background
- Line Width, Style, Arrow Style, Pattern
- Show Dimensions – shows sizes, lengths for precise drawing
- Best way to learn: play around on scratch model

Printing

- **Print all or parts of flowchart view of active model window – supports color**
- **Usual Print, Print Preview, Print Setup (File menu)**
- **Could consume many pages … also prints named views separately**
  - Print Preview, select only what you want for printing
- ***View > Page Breaks* to show how pages will break**
- **Alternative to printing from Arena: Windows Snipping Tool or PrintScreen key – sends screen to clipboard, paste into another application**
  - Alt+PrintScreen – sends only active window to clipboard
  - Could first pass through a paint application to crop, etc.

Help!

- **Extensive, comprehensive online system – including complete (electronic) manuals**
- **Interlinked via hypertext for cross referencing**
- **Multiple entry points, including Help menu (described above), links to websites**
- **button for context-sensitive help**
  - Click it, then click what you're curious about
- Help **button in most dialogs**
- **button (What's This?) in dialogs for info on things in that dialog**

Help! (cont'd.)

- **Tooltips – roll over something, get sticky note**
- **SMART library – small models illustrating points – subject index via *Help > Arena Smart Files***
  - See the Help entry for location of files on your system
    - Typical location on Windows 7:
      `C:\Users\Public\Public Documents\Rockwell Software\Arena\Smarts`
- **Online Help – http://www.rockwellautomation.com/support**
- **Examples folder – several detailed, complete examples, some fairly complex**
  - Typical location on Windows 7:
    `C:\Users\Public\Public Documents\Rockwell Software\Arena\Examples`

More on Running Models

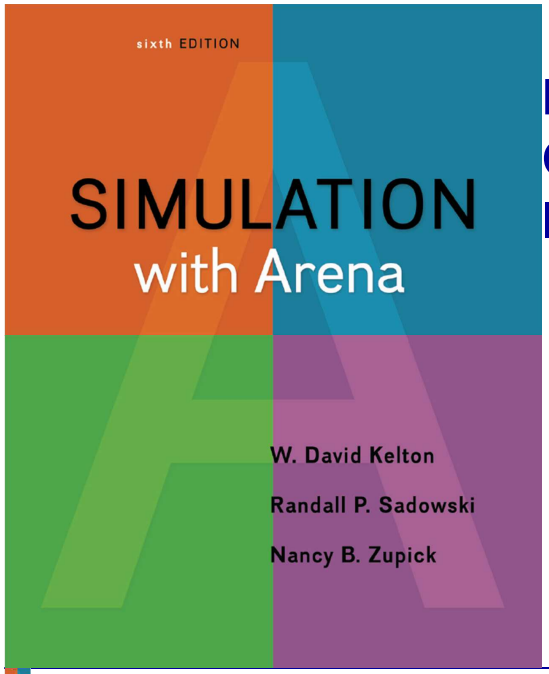
- **Run Menu; Standard & Run Interaction toolbars**
- ***Run > Setup* – many options to control run**
  - These are attached to model, and are not global
- ***Run > Go*** **– run simulation "normally" (depends on selections from *Run > Run Control* and *Tools > Options > Run Control*)**
- ***Run > Step*** **– one "step" at a time (verify, debug)**
- ***Run > Fast-Forward*** **– disable animation (faster)**
- ***Run > Pause*** **(or Esc key) – freeze run, resume with Go**
- ***Run > Start Over*** **– go back to beginning of simulation**

More on Running Models (cont'd.)

- *Run > End* – get out of run mode
- *Run > Check Model* – like compiling
- *Run > Review Errors* – for most recent Check
- *Run > Run Control > Command* – bring up interactive command-line window to control run
- *Run > Run Control > Breakpoints* – set times, conditions to interrupt for checks, illustration
- *Run > Run Control > Watch* – bring up a window to watch a variable or expression during run

More on Running Models (cont'd.)

- *Run > Run Control > Break on Module* – set/clear break when an entity enters or resumes activity on a module
- *Run > Run Control > Highlight Active Module* – highlight flowchart module being executed
- *Run > Run Control > Batch Run (No Animation)* – run model with no animation … this is even faster than Fast-Forward … usually used for "production runs" for statistical analysis
- *Run > SIMAN* – view or write model (.mod) and experiment (.exp) files for underlying SIMAN model



Modeling Basic Operations and Inputs

Chapter 4

Last revision December 22, 2013

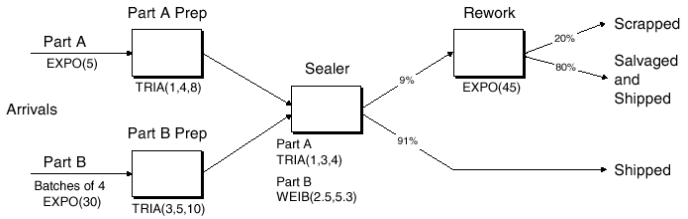
What We'll Do ...

- **Model 4-1: Electronic assembly/test system**
  - Modeling approaches
  - New Arena modules (Decide, Record)
- **Model 4-2: Enhanced electronic assembly/test**
  - Resource Schedules, States, and Failures
  - Frequency outputs
  - More on utilizations
- **Model 4-3: Enhancing the animation**
  - Queues, Entity Pictures, Resource Pictures
  - Adding Plots and Variables

What We'll Do ... (cont'd.)

- **Model 4-4: Adding entity travel times**
  - Stations, Transfers, Routes, animation of entity movement
- **Finding and fixing model errors**
- **Input analysis**
  - Specifying input distributions, parameters
  - Deterministic vs. random input
  - Collecting and using data
  - Fitting input distributions via Arena Input Analyzer
  - No data?
  - Nonstationary arrival processes
  - Multivariate and correlated input data

Electronic Assembly/Test System (Model 4-1)



- **Produce two different sealed elect. units (A, B)**
- **Arriving parts: cast metal cases machined to accept electronic parts**
- **Part A, Part B – separate prep areas**
- **Both go to Sealer for assembly, testing – then to Shipping (out) if OK, or else to Rework**
- **Rework – Salvaged (and Shipped), or Scrapped**

Part A

- **Interarrivals: expo (5) min.**
- **From arrival point, go immediately to Part A Prep**
  - Process = (machine + deburr + clean) ~ tria (1, 4, 8) min.
- **Go immediately to Sealer**
  - Process = (assemble + test) ~ tria (1, 3, 4) min.
  - 91% pass (i.e, 0.91 probability independently for each part), go to Shipped; Else go to Rework
- **Rework: (re-process + testing) ~ expo (45) min.**
  - 80% pass (i.e, 0.80 probability independently for each part), go to Salvaged; Else go to Scrapped

Part B

- **Interarrivals: *batches* of 4, expo (30) min.**
- **Upon arrival, batch breaks into 4 individual parts**
- **Proceed immediately to Part B Prep area**
  - Process = (machine + deburr +clean) ~ tria (3, 5, 10)
- **Go to Sealer**
  - Process = (assemble + test) ~ weib (2.5, 5.3) min. , *different* from Part A, though at same station
  - 91% pass, go to Shipped; Else go to Rework
- **Rework: (re-process + test) = expo (45) min.**
  - 80% pass (i.e, 0.80 probability independently for each part), go to Salvaged; Else go to Scrapped

Run Conditions, Output

- **Start empty & idle, run for 32 hours**
- **Collect statistics for each work area on**
  - Resource utilization
  - Number in queue
  - Time in queue
- **For each exit point (Shipped, Salvaged, Scrapped), collect total time in system (a.k.a. cycle time)**

Developing a Modeling Approach

- **Define pieces of model, modules, data structures, control logic**
- **Appropriate level of detail – judgment call**
- **Often multiple ways to model, represent logic**
- **This model:**
  - Entities are individual parts (two types)
  - Separate Create modules for two part types
  - Separate Process modules for each Prep area
  - Process modules for Sealer and Rework, each followed by a Decide module (2-way by Chance)
  - Depart modules for Shipped, Salvaged, Scrapped
  - Attribute **`Sealer Time`** assigned after Creates in Assign modules (parts have *different* times at *the* Sealer)
  - Record modules just before Departs for time in system

Building Model

- **New model window**
- **Attach Basic Process panel (if needed)**
- **Place modules**
  - Create (× 2)
  - Assign (× 2)
  - Process (× 4)
  - Decide (× 2)
  - Record (× 3)
  - Dispose (× 3)
- **Right click — repeat last action (place module)**
- **Auto-Connect, or manually connect via** 

Alternate strategy – place one module at a time, fill it out completely

Part A Create Module

- **Name: `Part A Arrive`**
- **Entity Type: `Part A`**
- **Time Between Arrivals**
  - Type: **`Random (Expo)`**
    - Pull-down list with options
  - Value: **`5`**
  - Units: **`Minutes`**
    - Pull-down list with options
- **Default what's not mentioned above**

Once these entries are made, they are placed on list for names of that type (Module Name, Entity Type, etc.) and will appear on future pull-down lists for that type of name.

Part B Create Module

- **Name: `Part B Arrive`**
- **Entity Type: `Part B`**
- **Time Between Arrivals**
  - Type: **`Random (Expo)`**
    - Pull-down list with options
  - Value: **`30`**
  - Units: **`Minutes`**
    - Pull-down list with options
- **Entities per Arrival: `4`**

Part A Attributes Assign Module

- **Name: `Assign Part A Sealer and Arrive Time`**
- **Add button:**
  - Type: **`Attribute`**
  - Attribute Name: **`Sealer Time`**
  - New Value: **`TRIA(1, 3, 4)`**
- **Add button:**
  - Type: **`Attribute`**
  - Attribute Name: **`Arrive Time`**
  - New Value: **`TNOW`** (to compute time in system on exit)

TNOW is internal Arena variable name for simulation clock; see Help > Arena Help > Contents > Variables, Functions, and Distributions > Variables > Date and Time Variables

Part B Attributes Assign Module

- **Name: `Assign Part B Sealer and Arrive Time`**
- **Add button:**
  - Type: **`Attribute`**
  - Attribute Name: **`Sealer Time`**
  - New Value: **`WEIB(2.5, 5.3)`**
- **Add button:**
  - Type: **`Attribute`**
  - Attribute Name: **`Arrive Time`**
  - New Value: **`TNOW`**

Names for things in Arena
– Default names usually suggested
– Names placed on appropriate pull-down lists for future reference
– All names in a model must be unique (even across different kinds of objects)

Prep A Process Module

- **Name: `Prep A Process`**
- **Action: `Seize Delay Release`**
- **Resources subdialog (Add button):**
  - Type: **`Resource`** (a pull-down option)
  - Resource Name: **`Prep A`**
  - Quantity: **`1`** (default)
- **Delay Type: `Triangular`**
- **Units: `Minutes`**
- **Minimum: `1`**
- **Value (Most Likely): `4`**
- **Maximum: `8`**

If several Resources were named (Add button), entity would have to Seize them all before Delay could start.

Prep B Process Module

- **Name: `Prep B Process`**
- **Action: `Seize Delay Release`**
- **Resources subdialog (Add button):**
  - Type: **`Resource`** (a pull-down option)
  - Resource Name: **`Prep B`**
  - Quantity: **`1`** (default)
- **Delay Type: `Triangular`**
- **Units: `Minutes`**
- **Minimum: `3`**
- **Value (Most Likely): `5`**
- **Maximum: `10`**

Sealer Process Module

- **Name: `Sealer Process`**
- **Action: `Seize Delay Release`**
- **Resources subdialog (Add button):**
  - Type: **`Resource`** (a pull-down option)
  - Resource Name: **`Sealer`**
  - Quantity: **`1`** (default)
- **Delay Type: `Expression`**
- **Units: `Minutes`**
- **Expression: `Sealer Time`**

*Recall – **`Sealer Time`** attribute was defined upstream for both Parts A and B … now its value is being used … allows for different distributions for A and B.*

Sealer Inspection-Result *Decide* Module

- **Decide module provides branch points**
  - *By Condition* (entity Attributes, global Variables)
  - *By Chance* (multi-sided, possibly-biased hypercoin flip)
- **Name: `Failed Sealer Inspection`**
- **Type: `2-way by Chance` (default)**
- **Percent True: `9`**
- **Different exit points for True, False results – connect appropriately downstream**

- *Note it's **percent** true, not **probability** of true … so "9" means probability of 0.09*
- *We arbitrarily decided "true" meant part failed inspection … could have reversed (but would change numerical results ... why? ... does this upset you? ... why?)*
- *This is a rich, deep, versatile module … explore its Help button*

Rework Process Module

- **Name: `Rework Process`**
- **Action: `Seize Delay Release`**
- **Resources subdialog (Add button):**
  - Type: **`Resource`** (a pull-down option)
  - Resource Name: **`Rework`**
  - Quantity: **`1`** (default)
- **Delay Type: `Expression`**
- **Units: `Minutes`**
- **Expression: `EXPO(45)`**

*Had to use general **`Expression`** choice for Delay Type since what we want (**`EXPO`**) is not directly on Delay Type pull-down list.*

Rework Inspection-Result Decide Module

- **Name: `Failed Rework Inspection`**
- **Type: `2-way by Chance` (default)**
- **Percent True: `20`**

We arbitrarily decided "true" meant part failed inspection

Record Modules

- **Arena collects and reports many output statistics by default, but sometimes not all you want**
- **Want time in system (avg, max) of parts sorted by their exit point (Shipped, Salvaged, Scrapped)**
  - It's this sorting that Arena doesn't do by default ... it would automatically sort *by Entity Type* if we had Entities checked in *Run > Setup > Project Parameters* (which we don't)
- **Record module can be placed in flowchart to collect and report various kinds of statistics from within model run as entities pass through it**
- **For Tally-type output performance measures**

Shipped Parts *Record* Module

- **Name: `Record Shipped Parts`**
- **Type: `Time Interval`**
  - Records time elapsed up to now (**`TNOW`**) from when an entity attribute was marked with a time "stamp" upstream ... Attribute Name is below ...
  - There are several other options for Type … explore via Record module's Help button!
- **Attribute Name: `Arrive Time`**
  - Was defined upstream as clock value in Assign modules instantly after each entity was Created
- **Tally Name: `Record Shipped Parts`**
  - Determines label in reports

Other two Record modules – just like this except for Name and Tally Name.

Dispose Modules

- **Three separate exit points for three separate part disposition (Shipped, Salvaged, Scrapped)**
- **Could have directed all three exit types to a single Dispose module**
  - Separate ones gets animation counts of three dispositions
  - Separate Dispose modules allows for differentially checking boxes to Record Entity Statistics
    - Produces flow statistics separated by entity type (*if* Entities Statistics Collection is checked in *Run* > *Setup* > *Project Parameters*), *not* by final disposition of part … so we *did* need our Record modules and Arrive Time attribute

Run > *Setup* for Run Control

- **Without this, model would run forever – no defaults for termination rule**
  - That's part of modeling, and generally affects results!
- **Project Parameters tab:**
  - Fill in Project Title, Analyst Name
  - Defaults for Statistics Collection, but we cleared check box for Entities
    - Not needed for what we want (we installed our own Record modules), and would slow execution
- **Replication Parameters tab:**
  - Replication length: **`32`**, accept **`Hours`** default for Time Units
  - Base Time Units: **`Minutes`** for inputs without Time Units option, internal arithmetic, and units on output reports

Different Part A, B Entity Pictures

- **Entity data module (just single-click on it in Project Bar, edit via spreadsheet only)**
- **Row for each Entity Type (`Part A`, `Part B`)**
- **Pull down Initial Picture pull-down menu, select different pictures for each Entity Type**
  - *Edit* > *Entity Pictures* to see, change list of pictures that's presented here … more later

Running Model

- **Check (if desired)**
  - Find button to help find errors
- **Go (will automatically pre-Check if needed)**
  - Some graphics don't show during run … will return when you End your run … control via *View > Layers*
  - Status Bar shows run progress – replication number, simulation time, simulation status
- **Animation speed**
  - Slider bar at top, or increase (> key), decrease (< key)
- **Pause (❚❚) or Esc key; ▶ to resume**
- ***Run > Step* (⏭) to debug**
- ***Run > Fast-Forward* (⏩) to turn off animation**
  - *Run > Run Control > Batch Run (No Animation)* is fastest

Viewing Results

- **Counters during animation for modules**
  - Create, Dispose, Decide – incremented when entity leaves
  - Process – number of entities currently in module
- **Asked at end if you want to see reports**
  - What you get depends on *Run > Setup > Project Parameters*
    - Looks like Rework area is bottleneck ... more later
  - Navigate through report with browsing arrows, tree at left
  - Tally, Time-Persistent, and Counter statistics
  - Avg, Min, Max, and 95% Confidence Interval half-widths
    - Confidence intervals are for steady-state expectations … Chapter 7
    - May not be produced if run is not long enough for reliable stats
- **Generally difficult/unreliable to draw conclusions from just one run … more later**

Model 4-2: Enhanced Electronic Assembly and Test System

- **Original model shown to production manager**
  - Pointed out that this is only first shift of a two-shift day — on second shift there are two operators at Rework (bottleneck station) … 16-hour days
  - Pointed out that Sealer fails sometimes
    - Uptimes ~ expo (2) hours
    - Repair times ~ expo (4) min.
  - Wants to buy racks to hold rework queue
    - A rack holds 10 parts
    - How many racks should be bought?
  - Run for 10 days (16-hour days)
- **Need: *Resource Schedules*, *Resource States*, *Resource Failures***

Run Conditions

- **Redefine a "day" to be 16 hours – *Run > Setup > Replication Parameters***
- **Change Replication Length to 10 (of these) days**

Schedules

- **Vary Capacity (no. units) of a resource over time**
- **In Resource Data module (spreadsheet view)**
  - For Rework Resource, change Type from **`Fixed Capacity`** to **`Based on Schedule`**
  - Two new columns – Schedule Name and Schedule Rule
  - Type in a Schedule Name (**`Rework Schedule`**)
  - Select a Schedule Rule – details of capacity decrease if Resource is allocated to an entity
    - *Wait* – Capacity decrease waits until entity releases Resource, and "break" will be full but maybe start/end late
    - *Ignore* – Capacity goes down immediately for stat collection, but work goes on until finished … "break" could be shorter or gone
    - *Preempt* – Processing is interrupted, resumed at end of "break"

Schedules (cont'd.)

- **Define actual Schedule that Resource will follow – Schedule data module**
  - Row already there since we defined **`Rework Schedule`**
  - Format Type is Duration for entries based on elapsed time past simulation start time
  - Type is Capacity, for Resource schedule (more later on Arrival Type)
  - Click in Durations column, get Graphical Schedule Editor
    - *X*-axis is time, *Y*-axis is Resource Capacity
    - Click and drag to define graph
    - Options button to control axis scaling, time slots in editor, whether schedule loops or stays at a final level forever
    - Can use Graphical Schedule Editor only if time durations are integers, with no Variables or Expressions involved

Schedules (cont'd.)

- Alternatively, right-click in row, select Edit via Dialog
  - Enter schedule Name
  - Enter pairs for Capacity, Duration … as many pairs as needed
    - If all durations are specified, schedule repeats forever
    - If any duration is empty, it defaults to infinity
  - *Can* involve Variables, Expressions
- Another alternative – right-click in row, select Edit via Spreadsheet
  - Enter capacity Value, Duration pairs

Resource Failures

- **Usually for unplanned, random downtimes**
- **Can start definition in Resource or Failure module (Advanced Process panel) … we'll start in Failure**
- **Attach Advanced Process panel if needed, single-click on Failure, get spreadsheet view**
- **To create new Failure, double-click – add new row**
- **Name the Failure**
- **Type – Time-based, Count-based (we'll do Time)**
- **Specify Up Time, Down Time, with Units for both**

Resource Failures (cont'd.)

- **Attach this Failure to correct Resource**
  - Resource module, Failures column, Sealer row – click
  - Get pop-up Failures window, pick Failure Name **`Sealer Failure`** from pull-down list
  - Choose Failure Rule from **`Wait`**, **`Ignore`**, **`Preempt`** (as in Schedules)
- **Can have multiple Failures (separate names) acting on a resource**
- **Can re-use defined Failures for multiple Resources (operate independently if they involve random variables)**

Frequencies

- **Record time-persistent occurrence frequency of variable, expression, or resource state**
  - Use here to record % of time rework queue is of length 0, (0, 10], (10, 20], … for info on number of racks needed
- **Statistic data module (Advanced Process panel)**
  - Five Types of statistics, of which Frequencies is one
  - Specify Name (**`Rework Queue Stats`**), Frequency Type (**`Value`**)
  - Specify Expression to track and categorize
    - Right-click in field to get to Expression Builder
  - Report Label (**`Rework Queue Stats`**)
  - Pop-up secondary spreadsheet for Categories (browse file)

Frequencies (cont'd.)

- **Add another Frequency (in Statistic module) to give a finer description of Sealer states**
  - Produces statistics on proportion of time Sealer is in each of its *three* possible states – Busy, Idle, and Failed
- **Frequencies are not part of default Category Overview report**
  - Open Frequencies report from Project Bar (get separate window)

Results of Model 4-2

- **Differ from those of Model 4-1 since this is a longer run, modeling assumptions are different**
  - All of which causes underlying random-number stream to be used differently (Chapter 12)
- **Prep A/B didn't change (other than run length and random variation) … need statistical analysis of simulation output (Chapters 6, 7, 12)**
- **Sealer is more congested (it now fails)**
- **Rework is less congested (50% higher staffing)**
- **Frequencies report suggests one rack suffices about 95% of the time, two racks all the time**
  - Standard vs. Restricted Percents – see text

Utilizations – Fine Points

- **Two utilizations reported for each Resource**
  - *Instantaneous Utilization* is time-average of ratio of number of units that are busy to number of units that are scheduled
    - By definition, counts periods when zero units are scheduled as zero-utilization periods
  - *Scheduled Utilization* is average number busy divided by average number available
    - No division-by-zero problem, assuming there were ever any units of Resource scheduled at all (if not, it shouldn't be in model)
- **Identical for fixed-capacity Resource**
- **Can differ for Resources on a variable Schedule**
  - If Resource capacity varies among several different positive values, it's better to use Scheduled Utilization
  - More issues, even finer points – see text

Model 4-3: Enhancing the Animation

- **Get "Spartan" generic default animation for some things (queues, connector-animation movement)**
  - Usually sufficient for verification, validation
- **Often want to customize, enhance it a bit**
  - More realism, impact
- **Pull animation away from logic in model window**
  - Useful for big models, complex animation
  - Named Views for model logic, animation, or close-ups
- **Default animation objects are connected to model logic and move with the module**
  - Identifiers, physical location (Shift-drag to decouple)

Changing Animation Queues

- **Lengthen (click, drag, maybe hold shift) to "hold" more entities**
  - Simulation logic, results OK if animated queue overflows
- **Rotate to re-orient for realism**
- **Change "form" of queue from *Line* (default) to *Point* — fixed places for entities**
  - Double-click on queue
  - Select Type to be Point
  - Click Points… button
  - Successively click Add for points, then OK
  - Drag them around on screen
  - *Check* Rotate box to show entities turning

Changing Entity Pictures

- **Earlier – used Entity data module to assign different Initial Pictures to different Entity Types**
- **Can customize list, alter pictures in it**
  - *Edit > Entity Pictures*
  - Left column – names, pictures currently on list
  - Right column –picture libraries (.plb filename extension)
  - Add a hand-drawn picture – Add button on left, name it in Value field at top, double-click on blank depressed button, then artwork (or paste in a copied graphics image)
    - New name won't appear in Entity data module until you type it there
  - Edit an existing picture – double-click, artwork
  - Copy a picture over from picture library

Adding Resource Pictures

- **Animate a Resource – Resource button  in animate toolbar – get Resource Picture Placement window**
- **Left column – default pictures for different Resource states**
  - Attach logically to a Resource by Identifier pull-down list
  - Double-click to edit artwork by hand, or paste in previously copied graphics images
  - Seize area – where seizing entity will "reside"
  - Multiple seize areas for multi-capacity Resources
- **Right column – picture libraries (.plb files) – can copy over to selected (depressed) state pictures**
- **Accept window, cross hairs, click to place**
  - Resize, reposition later

Adding Variables and Plots

- **Variable animation – just show a value of something as a number, watch it change**
  - Variable object  from Animate toolbar
  - Double-click, specify Expression to be shown (Expression Builder), and cosmetics
  - Resize, reposition later
- **Dynamic animated plots – Chapter 3**
- **Other animation objects from Animate toolbar**
  - Clock (TNOW), variety of formats
  - Level (thermometer) animation
  - Others discussed later

Model 4-4: Electronic Assembly and Test System with Part Transfers

- **Generalize Model 4-3**
- **All part transfers now take 2 minutes (not instant)**
  - Model, animate this
  - Materially changes model logic, results
  - Two-minute transfer times for:
    - Arriving parts to prep areas
    - Departing parts to appropriate exit
    - All internal part transfers
  - Transfers take two minutes regardless of distance
    - Fix this (unrealistic) assumption in Chapter 8

New Arena Constructs

- ***Station* – location where some process occurs**
  - Arrivals, manufacturing cells, departures
  - Each Station given a unique name
  - Can serve as an entry point for a section of model logic
  - *Station marker*  represents a logical station in flowchart/animation
- ***Station Transfer* – entities move between Stations without direct connection**
  - Several different types – we'll use *Routes* here, which allow for positive transfer time, but no other delays like "room" on transitway or transporters
  - *Route paths* represent Routes in flowchart/animation

Adding Route Logic – From Arrival

- **Stations and Station Transfers affect both model logic and animation**
- **Start with Model 4-3 … change to Model 4-4**
- **For incoming parts (A and B) delete connection from Assign modules to "Prep" Process modules**
  - Replace with Station/Route module pairs
    - Station module (Advanced Transfer panel) – define entity's location
      Module Name vs. Station Name
    - Route module (Advanced Transfer panel) – send entity out
      Route Time, Destination Station
  - No direct connections exiting from Route modules – Route module's Destination Station Name defines that

Adding Remaining Route Logic

- **Add Station modules for entry to each Prep area**
  - Station names are **`Prep A Station`**, **`Prep B Station`**, and are destination stations for Routes after arrivals
- **Process modules for Prep A, Prep B unchanged**
- **After prep, entities connected to Route module to send to next station (sealer)**
  - Don't need a separate Station module for outgoing side
- **Similar changes for rest of model**
  - Station modules for incoming parts into sealer, rework, each of three Record modules (entity exit points)
  - Route modules for outgoing parts out of sealer inspection, rework inspection (two for each Decide module – pass/fail)
- **Could run model now, get correct results … but no animation of transfers …**

Why Not Just Add Delays?

- **Simpler way to get two-minute transfer times:**
  - Insert a Process module with Action = Delay for 2 minutes on each relevant connection
  - Or, use Delay module from Advanced Process panel
- **This *would* work from modeling, numerical-output viewpoints**
- **But would not allow animation of part transfers, so we'll proceed with Stations and Routes**

Altering Animation – Stations

- **Add animation for Stations and Routes**
- **Station button , Animate Transfer toolbar**
  - Attach Identifier to it from pull-down list of station names
  - Get cross hairs, place (click) marker in animation
  - Can place several station markers for same logical station (e.g., to represent incoming, outgoing sides)
  - Can drag station markers around later

Altering Animation – Routes

- **Route button from Animate Transfer toolbar**
  - Options for appearance of entities as they travel route
  - Get cross hairs; click in origin, destination Station Markers
    - Intermediate clicks for corners along route
  - Can drag around endpoints, corners later
  - Alternatively, use Route animation to create both Station markers and Route animation
    - Click for beginning Station marker
    - Intermediate clicks for route corners
    - Double-click for ending Station marker
    - Then go back and double-click on the two Station markers to define their logical Identifiers

Altering Animation – Entity Pictures

- **Part B arrivals are in batches of four parts/batch**
  - But constant travel time to Prep B implies they travel "on top of each other" so it looks like just one part B
  - Try – change Route time from **2** to **`EXPO(2)`**, see separation along route
- **Create a dishonest illusion to animate batch**
  - Assign module just after **`Part B Arrive`**
  - Add assignment of Entity Picture to **`Picture.Batch B`**
  - *Edit* > *Entity Pictures* to draw new picture
    - Copy **`Picture.Part B`** and rename it **`Picture.Batch B`**
    - Double-click on picture, use Picture Editor to get four circles
  - When batch arrives to Prep B, change to single circle
    - Add Assign module after **`Prep B Arrival Station`**

Finding and Fixing Model Errors

- **If error prevents model from running, Arena will try to detect and lead you to it in Check or Run**
  - Undefined (or inconsistently spelled) Variables, Attributes, Resources
  - Unconnected modules
  - Duplicate names
  - Examples – see text
- **Highlight Active Module – selects active module during run animation**
- ***View* > *Layers* while running – change what shows during run animation**

Finding and Fixing Model Errors (cont'd.)

- **Module Break – stop when entity reaches module**
- **Debug Bar**
  - *View* > *Debug Bar*
  - Breakpoints, Calendar, Active Entity, Watch
  - Run Controller
  - Examples – see text

Input Analysis: Specifying Model Parameters, Distributions

- ***Structural*** **modeling: what we've done so far**
  - Logical aspects – entities, resources, paths, etc.
- ***Quantitative*** **modeling**
  - Numerical, distributional specifications
  - Like structural modeling, need to observe system's operation, take data if possible

Deterministic vs. Random Inputs

- ***Deterministic*****: nonrandom, fixed values**
  - Number of units of a resource
  - Entity transfer time (?)
  - Interarrival, processing times (?)
- ***Random*** **(a.k.a.** ***stochastic*****): model as a distribution, "draw" or "generate" values from to drive simulation**
  - Transfer, Interarrival, Processing times
  - What distribution? What distributional parameters?
  - Causes simulation output to be random, too
- **Don't just assume randomness away – validity**

Collecting Data

- **Generally hard, expensive, frustrating, boring**
  - System might not exist
  - Data available on wrong things – might have to change model according to what's available
  - Incomplete, "dirty" data
  - Too much data (!)
- **Sensitivity of outputs to uncertainty in inputs**
- **Match model detail to quality of data**
- **Cost – should be budgeted in project**
- **Capture variability in data – model validity**
- **Garbage In, Garbage Out (GIGO)**

Using Data: Alternatives and Issues

- **Use data "directly" in simulation**
  - Read actual observed values to drive model inputs (interarrivals, service times, part types, …)
    - Arena ReadWrite module ... see Model 10-2
  - All values will be "legal" and realistic
  - But can never go outside your observed data
  - May not have enough data for long or many runs
  - Computationally slow (reading disk files)
- **Or, fit probability distribution to data**
  - "Draw" or "generate" synthetic observations from this distribution to drive model inputs
  - We've done it this way so far
  - Can go beyond observed data (good and bad)
  - May not get a good "fit" to data – validity?

Fitting Distributions to Data with Arena Input Analyzer

- **Assume:**
  - Have sample data: Independent and Identically Distributed (IID) list of observed values from actual physical system
  - Want to select or fit a probability distribution for use in generating inputs for simulation model
- **Arena Input Analyzer**
  - Separate application, also via Tools menu in Arena
  - Fits distributions, gives valid Arena expression for generation to paste directly into simulation model

Fitting Distributions to Data with Arena Input Analyzer (cont'd.)

- **Fitting = deciding on distribution form (exponential, gamma, empirical, etc.) and estimating its parameters**
  - Several different methods (Maximum likelihood, moment matching, least squares, …)
  - Assess goodness of fit via hypothesis tests
    - H_0: fitted distribution adequately represents data
    - Get *p* value for test (small = poor fit)
- **Fitted "theoretical" vs. empirical distribution**
- **Continuous vs. discrete data, distribution**
- **"Best" fit from among several distributions**

Data Files for Arena Input Analyzer

- **Create data file**
  - Editor, word processor, spreadsheet, ...
  - Plain ASCII text – save as text or export)
  - Values separated by white space – blanks, tabs, linefeeds
  - Otherwise free format
- **Open data file from within Input Analyzer**
  - *File* > *New* or 
  - *File* > *Data File* > *Use Existing* or 
  - Get histogram, basic summary of data
  - To see data file: *Window* > *Input Data*
- **Generate "fake" data file to play around**
  - *File* > *Data File* > *Generate New*

Fit Menu

- **Fits distributions, does goodness-of-fit tests**
- **Fit a specific distribution form**
  - Plots density over histogram for visual "test"
  - Gives exact expression to Copy and Paste (Ctrl+C, Ctrl+V) over into simulation model
  - May include "offset" depending on distribution
  - Gives results of goodness-of-fit tests
    - Chi square, Kolmogorov-Smirnov tests
    - Most important part: *p-value*, always between 0 and 1:

      Probability of getting a data set that's more inconsistent with fitted distribution than data set you actually have, if fitted distribution is truly "the truth"

      "Small" *p* (< 0.05 or so): poor fit (try again or give up)

Fit Menu (cont'd.)

- **Fit all of Arena's (theoretical) distributions at once**
  - *Fit* > *Fit All* or 
  - Returns *minimum square-error* distribution
    - Square error = sum of squared discrepancies between histogram frequencies and fitted-distribution frequencies
    - Can depend on histogram intervals chosen: different intervals can lead to different "best" distribution
  - Could still be a poor fit, though (check *p* value)
  - To see all distributions, ranked: *Window* > *Fit All Summary* or 

Fit Menu (cont'd.)

- **"Fit" Empirical distribution (continuous or discrete): *Fit > Empirical***
  - Used when "theoretical" distributions fit poorly, or used intentionally
  - Can interpret results as Discrete or Continuous distribution
    - Discrete: get pairs (*Cumulative* Probability, Value)
    - Continuous: Arena will linearly interpolate *within* data range according to these pairs (so you can never generate values outside range, which might be good or bad)
    - *Edit* > *Copy Expression* to paste the (sometimes-lengthy) expression to Windows clipboard, then paste this copied expression into destination field in Arena model
      - Need to edit end of expression slightly by adding an extra pair at the end still inside the parentheses, "`, 1.000, xmax`" where `xmax` is the largest value you ever want to generate

Issues in Fitting Input Distributions

- **Not an exact science – no "right" answer**
- **Consider theoretical vs. empirical**
- **Consider range of distribution**
  - Infinite both ways (e.g., normal)
  - Positive (e.g., exponential, gamma)
  - Bounded (e.g., beta, uniform)
- **Consider ease of parameter manipulation to affect means, variances**
- **Simulation model sensitivity analysis**
- **Outliers, multimodal data**
  - Maybe split data set (details in text)

No Data?

- **Happens more often than you'd like**
- **No good solution; some (bad) options:**
  - Interview "experts"
    - Min, Max: Uniform
    - Avg., % error or absolute error: Uniform
    - Min, Mode, Max: Triangular
      Mode can be different from Mean – allows asymmetry
  - Interarrivals – independent, stationary
    - Exponential – still need some value for mean
  - Number of "random" events in an interval: Poisson
  - Sum of independent "pieces": normal (heed left tail ...)
  - Product of independent "pieces": lognormal

Cautions on Using Normal Distributions

- **Probably most familiar distribution – normal "bell curve" used widely in statistical inference**
- **But it has infinite tails in both directions … in particular, has an infinite left tail so can always (theoretically) generate negative values**
  - Many simulation input quantities (e.g., time durations) must be positive to make sense – Arena truncates negatives to 0
- **If mean μ is big relative to standard deviation σ, then P(negative) value is small … one in a million**
  - But in simulation, *one in a million can happen*
  - See text, Model 4-5
- **Moral – avoid normal as input distribution**

Nonstationary Arrival Processes

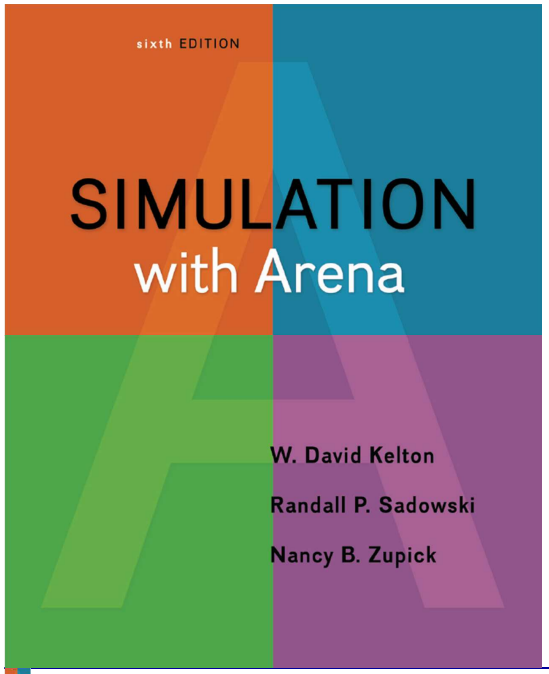
- **Events (often arrivals), rate varies over time**
  - Lunchtime at fast-food restaurants
  - Rush-hour traffic in cities
  - Telephone call centers
  - Seasonal demands for a manufactured product
- **It can be critical to model nonstationarity for model validity**
  - Ignoring peaks, valleys can mask important behavior
  - Can miss rush hours, etc.
- **Good model: *Nonstationary Poisson process***

Nonstationary Arrival Processes (cont'd.)

- **Two issues:**
  - How to specify/estimate *rate function*
  - How to generate from it properly during simulation
- **Several ways to estimate rate function – we'll just do *piecewise-constant***
  - Divide time frame of simulation into subintervals of time over which you think rate is fairly flat
  - Compute observed rate within each subinterval
  - In Arena, must convert to expected number of arrivals *per hour* on subintervals that need not be of one-hour length
    - Want expected 45 arrivals in a half hour; specify rate = 90 *per hour*
- **Example: Model 5-2 in Chapter 5**

Multivariate and Correlated Input Data

- **Usually assume all generated random observations in a simulation are independent (though from possibly different distributions)**
- **Sometimes not true:**
  - A "difficult" part requires long processing in both Prep and Sealer operations
  - This is positive correlation
- **Ignoring such relations can invalidate model**
- **See text for ideas, references**



Modeling Detailed Operations

Chapter 5

Last revision December 22, 2013

What We'll Do ...

- **Model 5-1: Simple call center**
  - Lower-level modeling, Advanced Process panel
  - Three-way decisions, Variables, Expressions, Storages
  - Blocks panel
  - Terminating vs. steady-state operation
  - Logical ("fake") entities
  - Terminating Condition in *Run* > *Setup*
- **Model 5-2: Enhanced call center**
  - Nonstationary Poisson arrival process
  - Sets – Resource, Counter
  - New Statistic data module Types
    - Counter, Time Persistent

What We'll Do ... (cont'd.)

- **Model 5-3: Enhanced call center with more output performance measures**
  - New Statistic data module Type
    - Output
  - Additional variable resources – look at staffing levels
- **Model 5-4: (*s*, *S*) inventory**
  - Not queueing
  - Choose to use low-level Blocks, Elements panels (SIMAN)
    - Can be done with higher-level panels

Model 5-1: Simple Call Center Setup

- **One phone number for customers to call in to**
  - 26 trunk lines, one needed for each call (incoming or outgoing, either talking or on hold)
  - Arriving call finding no free trunk lines gets busy signal, goes away
    - Count number of such rejected calls
- **Calls arrive with interarrivals ~ EXPO (0.857) min.**
  - First call arrives at time 0
- **Three incoming call types**
  - Initial recording to decide ~ UNIF (0.1, 0.6) min.
  - Tech support (76%, i.e., 0.76 probability independently for each call), sales (16%), order status (8%)

Model 5-1: Simple Call Center Setup (cont'd.)

- **Tech-support calls**
  - For product type 1 (25%), 2 (34%), or 3 (41%)
    - Recording/select time ~ UNIF (0.1, 0.5)
  - Needs qualified tech-support person
    - Two for type 1, three for type 2, three for type 3
      No crossover to another type ... will allow this in Model 5-2
    - Separate FIFO queues for each type
    - Conversation time ~ TRIA (3, 6, 18) min. for all types
  - Then leaves system
- **Sales calls**
  - All the same
  - Four sales staff, all the same
    - One FIFO queue feeding all sales staff
  - Conversation time ~ TRIA (4, 15, 45)
  - Then leaves system

Model 5-1: Simple Call Center Setup (cont'd.)

- **Order-status calls**
  - All the same
  - Handled automatically by phone system
    - No limit on number in process at a time, except for trunk-line limit
  - "Conversation" time ~ TRIA (2, 3, 4)
  - After "conversation," 15% of callers opt to talk to a person
    - Routed to sales staff, conversation lasts an additional TRIA (2, 3, 4)
    - Sales calls have higher priority (non-preemptive)
- **Center receives calls 8am – 6pm**
  - Must terminate arrival process at 6pm
  - Operate past 6pm if necessary to "flush out" all calls

Model 5-1: Simple Call Center Setup (cont'd.)

- **Output performance measures**
  - Number of calls attempted, rejected, completed
  - By call type – total time in system
  - By resource – time on hold, number of calls on hold
  - Resource utilization – of personnel, trunk lines
- **Terminating or steady-state**
  - Time frame of interest for each replication
    - Terminating – specific starting, stopping conditions (this model)
      Stopping conditions could be of several forms – fixed time, count, condition (here)
    - Steady-state – output performance measures are a limit as simulated time $\rightarrow \infty$
    - Choice usually depends on intent of study, not on model logic

Model 5-1: Simple Call Center Modeling Panels

- **Basic Process**
  - Highest, fastest modeling level, usually the place to start
- **Advanced Process**
  - Smaller building elements, other functions, more detail
- **Advanced Transfer**
  - Entity movement, material handling
- **Blocks, Elements**
  - Lowest modeling level, SIMAN simulation *language*
  - Repeats some capabilities of higher-level panels
  - Some functions available only here
- **Other special-purpose panels**
  - License-dependent

Model 5-1: Simple Call Center Data Structure

- **Re-use data in several places**
  - Define once, global to whole model
  - Redefine once – modeling generality, user efficiency
- **Arena (global) *Variables***
  - Store numbers (not formulas)
  - Define, initialize in Variable data module (Basic Process)
  - Can change during run (Assign module, other ways)
  - Scalar, 1-d array (vector), 2-d array (matrix)
- **Arena (global) *Expressions***
  - Store formulas (as well as numbers, but can't change)
    - Use math ops, numbers, random variates, Attributes, Variables, ...
  - Define in Expression data module (Advanced Process)
  - Scalar, 1-d array (vector), 2-d array (matrix)

Model 5-1: Simple Call Center Arrivals, Direct to Service

- **Create attempted calls**
  - Entity type **`Incoming Call`**, change later
  - Max Arrivals = **`MaxCalls`**, Variable initialized to 999999
    - At 6pm (time 600 minutes) change this to 1 to stop arrivals ... later
  - Entities per Arrival = **`CallsPerArrival`**, Variable initialized to 1
    - At 6pm (time 600 minutes) change this to 0 to kill arrivals ... later
- **Entity data module**
  - **`Incoming Call`** Entity Type already there
  - For Initial Picture, select **`Picture.Black Ball`**
- **Record module for an attempted call**
  - Add 1 to Counter Name **`Attempted Calls`**
  - Results – Category Overview report, User Specified

More detailed description – mouse over modules, read Data Tips that pop up

Model 5-1: Simple Call Center Arrivals, Direct to Service (cont'd.)

- **Decide module – Trunk Line Available?**
  - Type = 2-way by Condition
    - Select (logical) Expression for "If"
      - **`NR()`** is number of units of that resource that are busy now
      - **`MR()`** is number of units of that resource that exist now
  - False – Record rejected call counter, Dispose
  - True:
    - Seize a unit of **`Trunk Line`** Resource – Release later
      Resources data module for Trunk Line and other Resource levels
    - Increment Variable **`Total WIP`** for number of active calls
      Used in stopping rule at or after 6pm to sense if system is empty
    - Store module to animate entity during next Delay module
      Add Storage animation separately, identify with this logical storage by name
      Storage data module – entry made there by Store module
    - Delay module to listen to initial recording, make selection
      Could have used Process module, but this is simpler, faster
    - Unstore module to make entity animation disappear

Alternate strategy – Queue module from Blocks panel ... details in text

Model 5-1: Simple Call Center Arrivals, Direct to Service (cont'd.)

- **Decide module – Determine Call Type**
  - Three-sided coin flip – Type = N-way by Chance
    - Add button for more sides of coin
    - Get new exit point for each Add, plus one for Else
    - Note that probabilities are entered as *percentages* (0-100, not 0-1)
    - Last entry is "else"
- **Direct call to one of tech support, sales, or order-status areas**

Backed each area with colored box
Alternative way to organize – Submodels

Model 5-1: Simple Call Center Tech-Support Calls

- **Assign module**
  - Change Entity Type for separating out in results
  - Change Entity Picture for animation
- **Store – Delay – Unstore for recording, product type selection**
- **Decide module for product type**
  - Different three-sided coin flip
  - Direct to appropriate Process module for that product type
- **Process modules for tech-support service**
  - Seize-Delay-Release
  - Seize a unit from appropriate multi-unit Resource
  - Use **`Tech Time`** defined in Expression data module
- **Proceed to system exit logic ... later**

Model 5-1: Simple Call Center Sales Calls

- **Assign module – change Entity Type, Picture**
- **Process module**
  - Seize-Delay-Release
  - Seize a unit of **`Sales`** Resource
- **Sales calls priority over order-status calls that seek a person?**
  - Queue data module, **`Process Sales Call.Queue`**
    - Type = Lowest Attribute Value
      Attribute Name = **`Sales Call Priority`**
      Undefined for sales calls, so has value 0 ... will set to 1 for order-status calls that seek a person, putting sales calls ahead in the queue
    - Shared queue (with order-status calls seeking a person)

Not the only way to do this

- **Proceed to system-exit logic**

Model 5-1: Simple Call Center Order-Status Calls

- **Assign module – change Entity Type, Picture**
- **Delay block (Blocks panel) for robo-chat**
  - Includes Store/Unstore logic – alternative to earlier method
    - No automatic entry in Storage data module, so must enter manually
- **Decide module**
  - No sales person required – go directly to system-exit logic
  - Sales person required:
    - Assign module – set **`Sales Call Priority`** Attribute to 1 so these will have lower priority than real sales calls
    - Seize module for a unit of **`Sales`** resource
      Define Queue Name = **`Process Sales Call.Queue`** – shared with sales calls
      Process module does not allow for specifying a shared queue, so can't use here
    - Delay for conversation with sales person
    - Release the unit of **`Sales`** resource
- **Proceed to system-exit logic**

Model 5-1: Simple Call Center System Exit

- **All calls of all types come here when finished**
- **Release module – release the unit of `Trunk Line` resource seized upstream**
- **Assign module – decrement `Total WIP` variable**
- **Record module – increment `Completed Calls` counter**
- **Dispose of call**

Model 5-1: Simple Call Center Arrival-Cutoff Logic

- **Used to "choke off" arrival stream at 6pm**
- **Create a single *logical* (or "fake") entity at time 600 min. (6pm)**
  - Overkill on making sure just one is created
    - Time Between Arrivals = 999999 min., Max Arrivals = 1
- **Assign module to set Variable `MaxCalls` to 1**
  - Recall use of **`MaxCalls`** for Max Arrivals in Create module for attempted calls
- **Also in this Assign module, set `CallsPerArrival` to 0**
  - Since Create module will *always* schedule next arrival, and this makes the "size" of the next illegal arrival zero
- **Dispose of this single logical entity**

Creative use of such "logical" (a.k.a. "fake") entities enhances modeling flexibility, power, detail

Model 5-1: Simple Call Center Run > Setup

- **Replication Parameters tab (other tabs as usual)**
- **Base Time Units = Minutes**
- **Replication Length = Infinite (the default)**
- **Terminating Condition field:**

```
TNOW >= 600.0  &&  Total WIP == 0
```

`TNOW`	`>= 600.0`	`&&`	`Total WIP`	`== 0`
Arena clock Variable	*Greater than or equal to 600 minutes, (6pm)* (*Base Time Units*)	*Logical "and"*	*Variable we maintained in model*	*Equality test for zero*
It's 6pm or later		*and*	*there are no calls in the system.*	

*Could have used **`NR(Trunk Line)`** instead of **`Total WIP`***

Model 5-1: Simple Call Center Animation

- **Place three Storage animations**
  - Initial Recording Delay, Tech Call Recording Delay, Order Status Delay
  - Select proper Identifier in each from pull-down list
  - Graphic behaves like Queue animations
- **Four Queue animations**
  - Three tech-support call product types, sales
  - Came with four Process modules specifying Seize
- **Resource animations for three tech-support types, sales Resources**
  - Multi-unit Resource animations, as in Models 4-3, 4-4

Model 5-1: Simple Call Center Animation (cont'd.)

- **Variable animations for WIP at tech calls, sales**
  - For tech calls, Arena variable to animate is **`Process Product Type 1 Tech Call.WIP`**, etc. – pull-down list
  - For sales calls, must include order-status calls seeking a real person:

```
NR(Sales) + NQ(Process Sales Call.Queue)
```

- **Plot number of trunk lines busy, `NR(Trunk Line)`**
- **Labeling, background boxes as in model logic**

Model 5-1: Simple Call Center Results *(one replication ... sample of size only one!!)*

- **Trunk-lines-busy plot**
  - Starts, ends at 0 – startup, termination logic
  - Capped at 26 during run
- **734 attempted calls (User Specified section)**
  - 643 of them completed, the other 91 rejected
- **Sometimes see mixture of sales (green), order-status (blue) entities in sales queue**
- **Other "usual" outputs**
  - Times in system – separated out by call type
  - Queue lengths, times in queue – separated out by resource
  - Resource utilizations – normalized to [0, 1] by capacity

Model 5-2: Enhanced Call Center Changes

- **Incoming calls' arrival rate varies over day**
  - Probabilistic model – *Nonstationary Poisson process*
    - More in Section 12.3
  - Instead of a constant rate (= 1 / mean interarrival time), specify a rate *function*
    - Arena supports *piecewise-constant* rate function – "step" functions
      Easy to specify, strong theoretical support
    - Rate-function specification:

Caution – it's easy to generate this incorrectly ... see text for details

In Arena, rates MUST be entered in arrivals per HOUR, regardless of model's Base Time Units or time intervals

Table 5-2. Call Arrival Rates (Calls Per Hour)

Time	Rate	Time	Rate	Time	Rate	Time	Rate
8:00 - 8:30	20	10:30 - 11:00	75	1:00 - 1:30	110	3:30 - 4:00	90
8:30 - 9:00	35	11:00 - 11:30	75	1:30 - 2:00	95	4:00 - 4:30	70
9:00 - 9:30	45	11:30 - 12:00	90	2:00 - 2:30	105	4:30 - 5:00	65
9:30 - 10:00	50	12:00 - 12:30	95	2:30 - 3:00	90	5:00 - 5:30	45
10:00 - 10:30	70	12:30 - 1:00	105	3:00 - 3:30	85	5:30 - 6:00	30

Model 5-2: Enhanced Call Center Changes (cont'd.)

- **Sales-staff size varies over day**
  - Data in text, Schedule data module, **`Sales Schedule`**
- **Tech-support staff are partially cross-trained, work complicated schedule:**

Name	Product Lines	Time Period (30 minutes)																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Charity	1	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•					
Noah	1						•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•
Molly	1, 3			•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•			
Anna	1, 2, 3					•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	
Sammy	1, 2, 3				•	•	•	•	•	•	•		•	•	•	•	•	•	•	•			
Tierney	2	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•					
Aidan	2						•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•
Emma	2				•	•	•	•	•		•	•	•	•	•	•	•	•	•	•			
Mya	3	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•					
Ian	3						•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•
Christie	3				•	•	•	•	•		•	•	•	•	•	•	•	•	•	•			

Will use Arena Sets concept to implement this cross training

Model 5-2: Enhanced Call Center Changes (cont'd.)

- **4% of tech-support calls cannot be handled during the call, need offline back-office research**
  - Original call ends, same original talk-time distribution, gives up its trunk line, but not counted (yet) as completed
  - Case sent to back office (outside model boundaries), takes EXPO (60) minutes to resolve
    - Offline research may be carried over night, completed on a later day
  - Answer goes back to *same* tech-support person who took original call, with higher priority than incoming calls, but still might have to queue for this person
  - This tech-support person requests a trunk line for outgoing call, higher priority than incoming calls, but still might have to queue, talks for TRIA (2 ,4 ,9) min., call is *now* completed
  - Track number of each product type after research is done

Model 5-2: Enhanced Call Center Data Structure

- **Resources, Schedules**
  - Resource, Schedule data modules
  - Trunk Line – fixed capacity at 26
  - Sales – on Schedule **`Sales Schedule`**
  - 11 individual tech-support people on individual schedules
    - Caution – must fill out each schedule to all 22 half-hour periods, with leading/trailing 0's if necessary ... use Edit via Dialog or Spreadsheet, not graphical schedule editor
    - **`Ignore`** option to avoid shifting back schedule over multiple days
    - Include costing data for people in Resource data module
  - Define nonstationary arrival-rate function in Schedule module – **`Arrival Schedule`**
    - Enter trailing 0's in Edit via Dialog or Spreadsheet, not graphical schedule editor

Model 5-2: Enhanced Call Center Data Structure (cont'd.)

- **Sets – collect same-type items together**
  - Set, Advanced Set data modules (Basic, Advanced Process panels, resp.)
    - Refer to items in set by original name, or index (subscript) in set
  - Resource set for each tech-support product type
    - Members are those tech-support resources qualified
    - Individual resources already defined – Resource data module
    - Overlapping membership – some resources in multiple sets
    - Sets are ordered – here, put most versatile tech-support people at bottom, to "save" them for other calls ... Preferred Order in Seize
    - Will Seize from a set in model
  - Counter set – one for each hour
    - Count number of rejected calls in each hour
    - Individual counters already defined – Statistic data module
    - Use results later to decide when to increase staffing

Model 5-2: Enhanced Call Center Modifying the Model

- **Call-arrivals, termination, Run > Setup**
  - Create module
    - Type = Schedule, Schedule Name = **`Arrival Schedule`**
  - Delete the entire arrival-cutoff section from Model 5-1
    - **`Arrival Schedule`** cuts off arrivals at 6pm, via 0 rate
  - Delete **`Total WIP`** variable used to terminate Model 5-1
    - Use built-in NR(**`Trunk Line`**) instead in Terminating Condition
    - Delete Assign modules used to manage **`Total WIP`**
  - Record module for rejected calls
    - Index into Counter Set **`Rejected Calls`** with index

      `AINT((TNOW/60) + 1)`

      which is 1 for first hour, 2 for second hour, etc. (AINT truncates decimals toward zero)

Model 5-2: Enhanced Call Center Modifying the Model (cont'd.)

- **Tech-support calls**
  - Same through **`Determine Product Type`** Decide
  - Add Assign modules for each product type thereafter
    - Entity Type to distinguish product type in reports
    - Entity Picture to distinguish product type in animation
    - Attribute **`Tech Call Type`** (1, 2, or 3 by product type) for routing
  - Process modules, Resources subdialogs
    - Type = Set
    - Set Name = Product 1, etc.
    - Selection Rule = Preferred Order, to select earlier entries in set first

      Recall – we put more versatile tech-support people lower in the set list
    - Save Attribute = **`Tech Agent Index`**

      Entity attribute, carried along, in case of back-office research to send back to this same tech-support person for return call

Model 5-2: Enhanced Call Center Modifying the Model (cont'd.)

- **Back office, returned tech-support calls – all new**
  - Entry via True branch (4%) in Decide module **`Backoffice Research and Return Call?`**
  - Release this call's trunk line – going offline now
  - Delay (with storage) for EXPO (60) back-office research
  - Increment **`Tech Return WIP(Tech Call Type)`**
    - 1-dim. Variable array – defined in Variable data module
    - **`Tech Call Type`** is 1, 2, or 3, assigned in earlier Assign module
  - Decide module **`Product Type?`** based on Entity Type
  - Seize the same tech-support person – higher priority
  - *Then* seize a trunk line (higher priority), make return call
  - Then release this trunk line, tech-support person
  - Decrement **`Tech Return WIP(Tech Call Type)`**
  - Send entity to final Record, *after* trunk-line release there

Model 5-2: Enhanced Call Center Modifying the Model (cont'd.)

- **Statistic data module**
  - Ten Counter-type statistics, discussed earlier
  - Four Time-Persistent statistics to track expressions
    - **`Backoffice Research WIP`** to track total number of cases in research, via NSTO(**`Backoffice Research Storage`**)
    - **`Tech 1 Total Online WIP Stat`**, etc., to track number of that product type in back office via Expression **`Tech 1 Total Online WIP`**, etc., defined in Expression data module as

```
Process Product Type 1 Tech Call.WIP + Tech Return WIP(1), etc.
```

- **No changes needed in sales-calls or order-status-calls section of Model 5-1**

Model 5-2: Enhanced Call Center Modifying the Model (cont'd.)

- **Animation**
  - Delete **`Tech 1`**, **`Tech 2`**, and **`Tech 3`** resource animations
  - Change variables in three tech-support WIP displays to track total number of tech-support calls of that type present
  - New back-office storage animation, variable animation for number present
  - A new queue for each tech-support product type for return calls waiting for service
  - Added a resource animation (from a .plb library) for each individual tech-support person
    - Grouped by product type, colors for capabilities
- **Results**
  - Most rejected calls in hours 5-8 ... increase staff then ... ?

Model 5-3: Overall Call-Center Stats Setup

- **Develop overall operational-cost measure**
  - Two cost categories – staffing/resource, and poor service
- **Develop overall measure of service, % of calls rejected**
- **Add options for increased staffing, improvement**
- **Make 5 replications, focus on weekly costs**
  - IID replications, so will not carry over back-office research

Model 5-3: Overall Call-Center Stats Staffing/Resource Costs

- **Resource data module – hourly costs for people**
  - $20/hr. for each sales staffer
  - $18/hr. – $20/hr. for each tech-support, depending on skill
  - These salary costs paid when on duty, busy or idle
  - Summing, get $12,820/week (details in text)
  - View all this existing staff as fixed

Model 5-3: Overall Call-Center Stats Staffing/Resource Costs (cont'd.)

- **Increase sales, tech-support staff noon-4pm**
  - Variable **`New Sales`** = number of new sales staff
    - $17/hr., 4 hrs./day, 5 days/week, so $340/week for each add'l. staff
    - Schedule data module to add capacity – edit via dialog or spreadsheet, not graphical editor
    - Resource (**`Sales`**) already exists in Resource data module
  - Variables **`New Tech 1`**, etc., and **`New Tech All`** for number of new tech-support people qualified as named
    - $16/hr. for each one-product staff, $18/hr. for each all-product staff
      $320/week for each single-product staff, $360/week for each all-product staff
    - New entries in Resource data module
      Larry, Moe, Curly, Hermann for 1, 2, 3, All, resp.
    - Schedule data module to add capacity – dialog or spreadsheet edit

Model 5-3: Overall Call-Center Stats Staffing/Resource Costs (cont'd.)

- **Maybe increase number of trunk lines beyond 26**
  - $98/week flat fee for each trunk line
- **Define Expression `New Res Cost` for all resource costs:**

```
New Sales*340
+ (New Tech 1 + New Tech 2 + New Tech 3)*320
+ New Tech All*360
+ 98*MR(Trunk Line)
```

  - This does not depend on simulation results, only on setup

Model 5-3: Overall Call-Center Stats Customer-Dissatisfaction Costs

- **Incur cost for caller wait on hold, past a threshold**
  - 3 min. for tech, 1 min. for sales, 2 min. for order-status
  - Beyond threshold, incur per-min. costs of $0.368 for tech, $0.818 for sales, $0.346 for order-status
    - In practice, such costs are difficult to estimate
  - Three new Assign modules (orange backing) accumulate "excess" (beyond threshold) wait times on hold
    - Tech support (other two are similar): Variable **`Excess Tech Wait Time`** increased by **`MAX(ENTITY.WAITTIME - 3, 0)`**
      ENTITY.WAITTIME is built-in Arena attribute holding all wait times (including in queues) so far ... luckily, there were none before the preceding Process module
  - At end, multiply excess wait times by per-min. costs, multiplied by 5 (to put on a weekly basis)
    $5 \times \$0.368 = \$1.84$ for tech, $5 \times \$0.818 = \$4.09$ for sales, $5 \times \$0.346 = \$1.73$ for order-status

Model 5-3: Overall Call-Center Stats Overall Output Performance Measures

- **Statistic data module, `Total Cost` entry**
  - Type = Output, computed only at end of replication

```
New Res Cost
     + Excess Sales Wait Time  * 4.09
     + Excess Status Wait Time * 1.73
     + Excess Tech Wait Time   * 1.84
     + 12820
```

- **Statistic data module, `Percent Rejected` entry**
  - Counter **`Total Rejected Calls`** accumulated in new Record module in call-arrival area (orange backing)
    - Already accumulating hour by hour, but this is total over the day
  - Type = Output

```
100 * NC(Total Rejected Calls) / NC(Attempted Calls)
```

    - NC is Arena function that returns the value of that counter

Model 5-3: Overall Call-Center Stats Replication Conditions

- **Run > Setup > Replication Parameters, Initialize Between Replications**
  - Statistics? System? Details in text
  - Default is both – only way to get truly IID replications
    - Destroys overnight tech-support research jobs, but to do otherwise would complicate model – so accept
- **Run > Setup > Project Parameters**
  - Turned off all but Costing Statistics Collection, for speed
    - Costing required to get ENTITY.WAITTIME

Model 5-3: Overall Call-Center Stats Results

- **Results from five replications**
  - Base Case – no additional staff, still 26 trunk lines

    `Total Cost` = $22,242.55 ± $1,439.47

    `Percent Rejected` = 11.96% ± 1.39%

    Avg. over 5 replications 95% Conf. int. half-widths
  - Add 3 of each of five staff types, 3 more trunk lines

    `Total Cost` = $23,683.35 ± $616.00

    `Percent Rejected` = 1.61% ± 1.52%

    Is this "better?"
- **Use in Chapt. 6 for statistically valid experiments**
  - Statistical precision
  - Compare several alternatives, select best
  - Search for configuration that minimizes cost, subject to upper limit on percent rejected

Model 5-4: (*s*, *S*) Inventory Simulation Setup

- **Different kind of model – not queueing**
- **Use Blocks and Elements panels exclusively – SIMAN simulation language**
  - Mostly just to demonstrate this capability
  - Could be done with higher-level panels we've been using (Exercise 5-17)
- **Company carries a single discrete item (widgets) in inventory**
- **$I(t)$ = inventory level (an integer) at time t days past the beginning of the simulation; $I(0) = 60$**
- **Run simulation for 120 round-the-clock days**

Model 5-4: (*s*, *S*) Inventory Simulation Customer Demands Against Inventory

- **Customer interarrival times ~ EXPO (0.1) day (round the clock)**
  - First arrival not at time 0 but after an interarrival time past 0
- **Demand size is discrete random variable**
  - 1, 2, 3, 4 with respective probabilities 0.167, 0.333, 0.333, 0.167
- **If enough items are physically on hand in inventory to satisfy a demand, customer gets demand and leaves**
- **If demand > number of items on hand, customer gets whatever is there and the rest of the demand is backlogged (*I*(*t*) becomes negative)**
  - If *I*(*t*) was already negative, it just goes more negative

Model 5-4: (*s*, *S*) Inventory Simulation Inventory Review, Replenishment

- **"Take inventory" just past midnight each day**
  - So at exactly times 0, 1, 2, ..., 119 (not 120 ... see below)
  - Two managerially-chosen constant integers $s = 20$ and $S = 40$ (must have $s < S$ if we change these values)
  - If $I(t) \geq s$, do nothing until next inventory evaluation exactly 24 hours later
  - If $I(t) < s$, order $S - I(t)$ items from supplier (order "up to" *S*)
  - Order does not arrive instantly from supplier, but after a *delivery lag* (a.k.a. *lead time*) ~ UNIF(0.5, 1.0) day, so sometime during the last half of the day of ordering
    - In the meantime, inventory level could fall further from additional demands, so inventory level will not necessarily pop up to *S* when the order arrives, but to something less than *S*

Model 5-4: (*s*, *S*) Inventory Simulation Cost Structure

- **Average ordering cost per day**
  - When an order is placed, incur a fixed cost of \$32, plus an incremental cost of \$3 per item ordered
  - If no order is placed at the beginning of a day, there's no ordering cost, not even the fixed cost
  - At end of simulation, divide total of ordering costs by 120
- **Average holding cost per day**
  - Whenever $I(t) > 0$, incur \$1 per day per item on hand
  - Average holding cost = $\int_0^{120} 1 \times \max(I(t), 0)\, dt \Big/ 120$
- **Average shortage cost per day**
  - Whenever $I(t) < 0$, incur \$5 per day per item in backlog
  - Average shortage cost = $\int_0^{120} 5 \times \max(-I(t), 0)\, dt \Big/ 120$

Model 5-4: (*s*, *S*) Inventory Simulation Cost Structure (cont'd.)

- **During periods when *I*(*t*) = 0 there's neither holding nor shortage cost incurred**
- **Overall performance measure**
  = Average *total* cost per day
  = sum of average ordering, holding, and shortage costs per day
- **Don't evaluate inventory at time 120**
  - We might order and incur an ordering cost then, but order will never arrive
  - We'll fudge this, but an Exercise asks you to do it right

Model 5-4: (*s*, *S*) Inventory Simulation Data Structure

- **Use Blocks, Elements panels exclusively**
  - Even for Variables, Expressions, Attributes, Entities, statistics collection, and run control
- **Variables Element (initialized, or default to 0 initially)**
  - **`Inventory Level`** = ***I***(*t*), changes during run, initialized to 60
  - **`Little s`** = s = 20
  - **`Big S`** = S = 40
  - **`Total Ordering Cost`** accumulator
  - **`Setup Cost`** = 32
  - **`Incremental Cost`** = 3
  - **`Unit Holding Cost`** = 1
  - **`Unit Shortage Cost`** = 5
  - **`Days to Run`** = 119.9999 (The Fudge)

Model 5-4: (*s*, *S*) Inventory Simulation Data Structure (cont'd.)

- **Expressions element**
  - Define **`Interdemand Time`**, **`Demand Size`**, **`Evaluation Interval`**, **`Delivery Lag`**
    - ***Cumulative*** **probabilities in DISC function for `Demand Size`**
- **Attributes, Entities elements**
  - Just to define these objects
- **Project, Replicate elements**
  - Similar to *Run > Setup*
- **DStats element**
  - Request accumulation of integrals for total holding, shortage costs

Model 5-4: (*s*, *S*) Inventory Simulation Data Structure (cont'd.)

- **Outputs element**
  - Two entries, both of Data type "Output" so that they're executed only at end of run, and reported
  - Avg Ordering Cost computed
  - Avg Total Cost added up
    - OVALUE returns most recent value
    - DAVG returns time-persistent average

Model 5-4: (*s*, *S*) Inventory Simulation Logic for Customer Demands

- **Create block for arrival**
  - Entity Type is **`Customer`**
  - Uses **`Interdemand Time`** Expression
  - First Creation after an **`Interdemand Time`**
- **Assign block to decrement `Inventory Level` by a `Demand Size`**
  - **`Demand Size`** was defined as an Expression
  - Backlogging naturally happens
- **Dispose block for customer exit**
  - If backlogged, is accounted for automatically in the (simple) definition and tracking of **`Inventory Level`**

Model 5-4: (*s*, *S*) Inventory Simulation Inventory Evaluation

- **Create block for `Inventory Evaluator` entities**
  - First Creation at time 0 – evaluate inventory at start of run
  - Interval is **`Evaluation Interval`**, defined as Expression
- **Branch block – somewhat like Decide module**
  - To determine whether to place an order now
  - Add "branches," each evaluated as true or false
  - Clone of incoming entity sent out along each "true" branch, but at most Max Number of Branches will be sent out
    - So we set Max Number of Branches to **1** (default is ∞)
  - First branch of type "If" – if "true" we want to order
  - Second branch of type "Else" – if "true" it means that the first branch was "false" so we don't order – just Dispose

Model 5-4: (*s*, *S*) Inventory Simulation Placing an Order

- **If we exit the Branch block via the top "If" branch, it must be that $I(t) < s$ so we want to order up to *S***
- **Assign block**
  - Define **`Order Quantity`** Attribute
    - Could have made this a Variable in this model with these parameters, but it's more general for it to be an Attribute ... why?
  - Increment **`Total Ordering Cost`** Variable
- **Delay block for `Delivery Lag`**
- **Assign block to increment `Inventory Level` by the `Order Quantity`**
- **Dispose block**

Model 5-4: (*s*, *S*) Inventory Simulation Animation

- **Plot separate "in the black" and "in the red" curves**
  - If in backlog, red curve will be plotted in negative direction due to its Expression
- **Pair of Level ("thermometer") animations**
  - Fill Direction for "in the red" is Down