

Graphical User Interfaces (GUI)

W. M. Song 桑慧敏

Tsing Hua Univ. 清華大學

2015.12.02

“Reset, Run” and “Experimenter”

Reset, Run

- **Advantage:** We see all values in GlobalTable update while running the FlexSim.
- **Disadvantage:** (1) There is no automatic “warmup time” setting. (2) Troublesome to run many replications.

Experimenter

- **Advantage:**
Easy to set up “Warmup Time”
- **Disadvantage:**
We do not see values in global Table update while running the “experimenter”

- **Q :** How can we integrate both advantages of “Reset Run” and “Experimenter”?

“Reset, Run” and “Experimenter”

- Q 1: How can we integrate both advantages of “Reset Run” and “Experimenter”?
- Q 2: How do we clear those Tables which **without** “Clear on Reset”?

A1: Self Code (手動程式), see P. 7

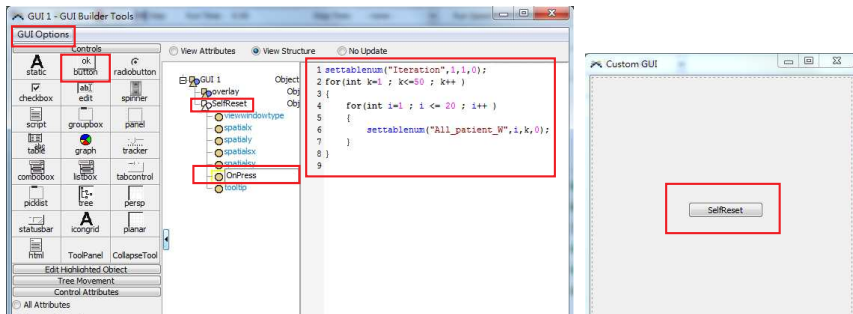
- `applicationcommand` (“reset”)
- `applicationcommand` (“run”)

A 2: Graphical User Interfaces (GUI), see P. 4

- Apply in PE Case, Page 5

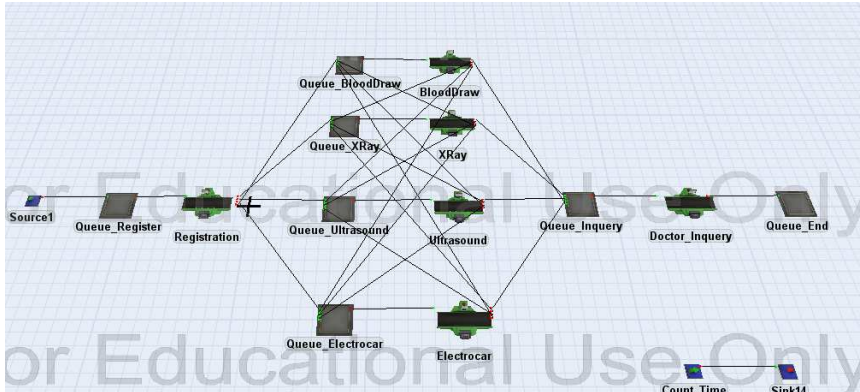
Graphical User Interfaces (GUI)

- Tools → Graphical User Interfaces → Add
- Controls → bottom → drag to right part **Customer GUI**
- Change name on **Treenode** (eg. **SelfReset**, see Figures below) → 點選左上角之 GUI Options → Assign this GUI to the "Control" button
- SelfReset → OnPress → Write code to reset the two new tables in PE case (see next page) → Finally close this window
- To implement "reset", click "SelfReset" (in Custom GUI) and "Reset" (in left top of FlexSim interface)



Ex: PE Model With 4 Inspections

- Q: Replicate 50 times for 20 customers. Draw Confidence Intervals for $WQ(i), i = 1, 2, \dots, 20$.
- Remind: You are asked to write this code in FlexSim in the Final Exam



Record to Tables in Every Replication

- Add 2 Global Tables: All_patient_W, Iteration
- Table Name: All_patient_W , Row: 20 , Columns: 50
 - Get and Set WQ for 20 Patients in every Replication
- Table Name: Iteration , Row: 1 , Columns: 1
 - Record No. of Replications
- Note: we do not click “Clear on Reset” for these two tables

Global Table - All_patient_W

Name: All_patient_W Rows: 20.00 Columns: 50.00 ☐ Clear on Reset

	Replication1	Replication2	Replication3	Col 4	Col 5	Col 6	Col 7
Patient1	48.08	40.91	41.89	42.17	44.95	34.38	
Patient2	46.88	46.66	46.09	42.85	47.64	53.48	
Patient3	41.00	57.57	48.87	43.19	53.66	63.59	
Row 4	62.75	63.08	58.78	48.84	70.35	62.66	
Row 5	38.79	72.47	63.14	56.76	63.78	76.46	
Row 6	46.69	67.96	70.85	66.02	71.66	88.78	
Row 7	45.66	72.41	74.74	80.85	64.62	78.10	
Row 8	58.31	72.41	79.91	88.51	85.31	92.97	
Row 9	61.14	72.56	106.51	102.53	88.52	90.30	
Row 10	70.99	76.18	114.53	101.06	96.62	87.01	
Row 11	83.07	83.67	119.51	100.25	99.94	98.34	
Row 12	79.76	95.04	126.63	102.41	107.31	99.61	
Row 13	72.96	102.57	123.43	93.82	109.70	104.78	
Row 14	83.53	104.98	125.20	113.31	120.10	108.07	
Row 15	68.79	105.13	127.57	94.96	123.40	104.53	
Row 16	86.02	113.90	116.32	100.08	116.28	110.67	
Row 17	74.25	114.80	136.04	108.79	120.37	125.27	
Row 18	80.90	95.01	130.26	108.92	124.39	121.57	
Row 19	75.43	122.54	121.01	106.74	138.38	126.66	
Row 20	83.99	107.84	142.61	110.53	143.39	135.81	

Global Table - Iteration

Name: Iteration Rows: 1.00 Columns: 1.00 ☐ Clear on Reset

0.00

Applicationcommand ("reset"), ("run")

- Queue_End → Triggers → OnEntry



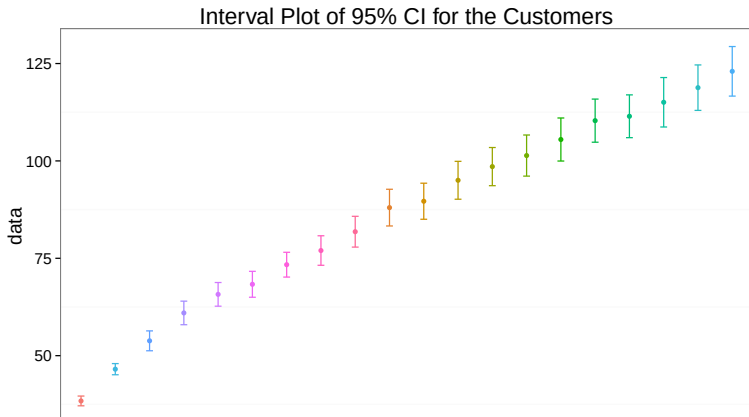
```

/Queue_End - OnEntry
Queue_End - OnEntry
28 // Performance Measures
29 if(getinput(current)==gettablenum("Total_Arrivals",1,1))
30 {
31   settablenum("Iteration",1,1,gettablenum("Iteration",1,1)+1);
32   //*****E(W) Start*****
33   double time_sum=0;
34   double average_time;
35   for(int i=1;i<=gettablenum("Total_Arrivals",1,1);i++)
36   {
37     time_sum = time_sum + gettablenum("Each_Patient_Wait_Time",i,3);
38   }
39   average_time = time_sum/gettablenum("Total_Arrivals",1,1);
40   settablenum("Performance_Measures",1,1,average_time);
41   //*****E(W) Stop*****
42   :
43   :
44   :
45   :
46   :
47   :
48   :
49   :
50   :
51   :
52   :
53   :
54   :
55   :
56   :
57   :
58   :
59   //***** Utilization Start*****
60   double query_start_time = gettablenum("Query_Start_Time",1,1);
61   double utilization = (query_time_sum)/(time()- query_start_time);
62   settablenum("Performance_Measures",4,1,utilization);
63   //***** Utilization Stop*****
64   //endreplication(1);
65   for(int k=1; k<=gettablenum("Total_Arrivals",1,1) ; k++)
66   {
67     settablenum("All_patient_W",k,gettablenum("Iteration",1,1),gettablenum("Each_Patient_Wait_Time",k,3));
68   }
69   if(gettablenum("Iteration",1,1)<50)
70   {
71     applicationcommand("reset");
72     applicationcommand("run");
73   }
74 }
75

```

Interval Plot for WaitTime in Queue

- Interval Plot Via R (20 Customers)
- It seems that there is no steady-state



Appendix: R Code for C.I.

```
1 library(dplyr)
2 library(ggplot2)
3 data = read.xlsx("c:/Users/Kevin/Dropbox/SongGrads-2015/課堂/PE_GUI_R/20customer50Rep.xlsx",1,header=T);
4
5 # [GetStats] :
6 # Takes one argument data, and returns
7 # the mean and 95% CI of each column in a data frame
8 GetStats <- function( data )
9 {
10   stats <- lapply( data, function(x)
11   {
12     # obtain the mean and 95 percent confidence interval
13     average <- mean(x)
14     upper_bound <- average + 1.96 * se(x)
15     lower_bound <- average - 1.96 * se(x)
16
17     return( list( average = average, upper_bound = upper_bound, lower_bound = lower_bound ) )
18   })
19
20 CustomerCI <- do.call( rbind, stats ) %>%
21   data.frame() %>%
22   mutate_each( funs(as.numeric) ) %>%
23   mutate( name = names(data) )
24
25   return(CustomerCI)
26 }
27
28 CustomerCI <- GetStats( data)
29 # plot
30 # order of the x axis
31 x_axis_label <- c( names( data ) )
32
33 ggplot( CustomerCI, aes( name, average, color = name ) ) +
34   geom_point() +
35   geom_errorbar( aes( ymax = upper_bound, ymin = lower_bound ), width = 0.2 ) +
36   guides( color = FALSE ) +
37   labs( x = NULL, y = "data", title = "Interval Plot of 95% CI for the Customers" ) +
38   theme_bw(16)+
39   theme(panel.grid.major=element_line(colour=NA))+
40   scale_x_discrete( limits = x_axis_label ) +
41   theme( axis.text.x = element_text( angle = 30, hjust = 1 ) )
42
```