Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

DEVELOPING INDUSTRIAL STRENGTH SIMULATION MODELS USING VISUAL BASIC FOR APPLICATIONS (VBA)

Marvin S. Seppanen

Productive Systems
2225 Garvin Heights Road
Winona, MN 55987-5465, U.S.A.

ABSTRACT

Since 1984 the author has developed simulation models that use input data from spreadsheets. These original applications used a standalone Basic program to convert Lotus 123® data into Siman Experiment Frames. While this process has evolved overtime, it did not reach a truly viable level until Arena® 3.0 introduced Visual Basic® for Applications (VBA) by Microsoft®. This advanced tutorial demonstrates the basic concepts developed by the author to transfer data between Excel® and Arena. The same techniques can be used to communicate simulation data with a wide range of VBA supported tools, such as Access®, AutoCAD®, and Visio®. Arena permits the model developer to use VBA as the model file is loaded, executed, or terminated or as entities flow through the Arena model modules. This tutorial focuses on the design of Excel workbooks for simulation applications and the transfer of data to/from Arena using VBA.

1 INTRODUCTION

Figure 1 illustrates some of the potential range of data exchange available to the model developer using VBA code. This tutorial concentrates on the exchange of data between Arena and Excel. However, the user interface is identical in all application that supports VBA. Pressing the Alt-F11 key on any application that supports VBA loads the editor illustrated in Figure 2. The structure of the VBA language is beyond the scope of this presentation but documented in numerous texts, including Wells (1995), Getz (1997), and Lomax (1998).

The new VBA user has much to learn but help is always as close as the F1 and F2 keys. Pressing the F2 key in the Microsoft Visual Basic editor provides lists of available constants, functions, and properties. Of these three constructs only the value of a property can be changed and then only if it has not been declared as read only. Actually VBA is quite easy to learn because the built in editor automatically checks the syntax of each code line as it is entered. The Debug Compile feature checks that all variables have been defined prior to attempting execution. Finally even during model execution, many VBA errors can be interactively debugged and corrected without restarting the execution of the Arena model. This interactive debugger also includes a breakpoint feature and the ability to view the current variable values as the code is executed.

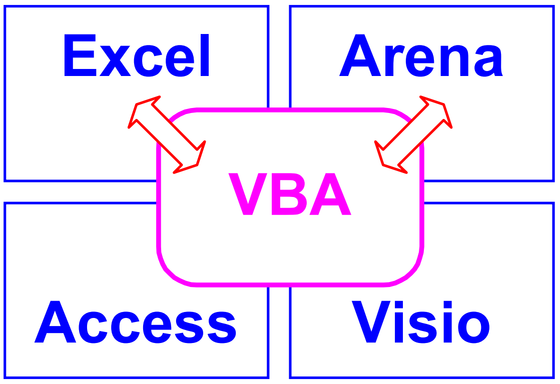
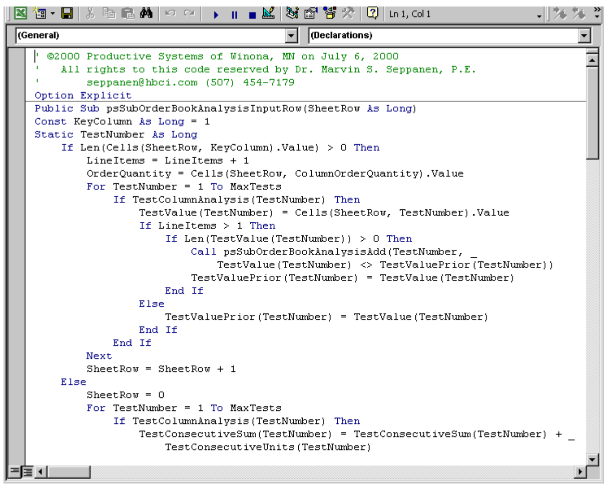


Figure 1: Some Potential VBA Data Exchanges



```
' ©2000 Productive Systems of Winona, MN on July 6, 2000
'   All rights to this code reserved by Dr. Marvin S. Seppanen, P.E.
'       seppanen@hbci.com (507) 454-7179
Option Explicit
Public Sub psSubOrderBookAnalysisInputRow(SheetRow As Long)
Const KeyColumn As Long = 1
Static TestNumber As Long
    If Len(Cells(SheetRow, KeyColumn).Value) > 0 Then
        LineItems = LineItems + 1
        OrderQuantity = Cells(SheetRow, ColumnOrderQuantity).Value
        For TestNumber = 1 To MaxTests
            If TestColumnAnalysis(TestNumber) Then
                TestValue(TestNumber) = Cells(SheetRow, TestNumber).Value
                If LineItems > 1 Then
                    If Len(TestValue(TestNumber)) > 0 Then
                        Call psSubOrderBookAnalysisAdd(TestNumber, _
                            TestValue(TestNumber) <> TestValuePrior(TestNumber))
                        TestValuePrior(TestNumber) = TestValue(TestNumber)
                    End If
                Else
                    TestValuePrior(TestNumber) = TestValue(TestNumber)
                End If
            End If
        Next
        SheetRow = SheetRow + 1
    Else
        SheetRow = 0
        For TestNumber = 1 To MaxTests
            If TestColumnAnalysis(TestNumber) Then
                TestConsecutiveSum(TestNumber) = TestConsecutiveSum(TestNumber) + _
                    TestConsecutiveUnits(TestNumber)
```

Figure 2: VBA Editor Window

2 ARENA MODEL DESIGN USING VBA

The author has used embedded VBA code to generate the entire Arena experiment frame and animation displays for a large number of very complex simulation models. This technique allows simulation users to maintain and modify the required model data without knowing the details of the Arena simulation software. In addition, the Simulation Workbook format has proven to be a benefit when designing the initial model structure. All model variables, attributes, expressions, resources, queues, and stations are specified in the workbook. Therefore it is relatively easy to change or update those definitions.

Figure 3 illustrates the typical flow of information into/from the industrial strength Arena model. All user changeable data (options, parameters, etc.) are contained in the Simulation Workbook, Simulation.XLS. When Arena loads the model file, Model.DOE, the imbedded VBA codes opens Excel and processes the information contained in the Simulation Workbook. This data is used to generate the model's experiment frame and animation display. The information read from the Simulation Workbook is reported in a text-based file, Workbook.TXT to provide a record of how the simulation run was structured. If input data errors are detected, a special report file, Message.TXT is also generated. If specified by the user in the Simulation Workbook, the simulation model is automatically executed. The VBA may generate a customized detailed report file, Trace.TXT to support model verification. The standard Arena report file, Model.OUT, and database, Model.MDB, are also generated.

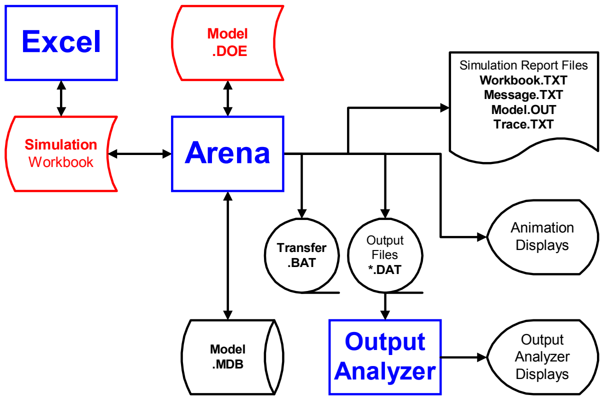


Figure 3: Industrial Strength Simulation Data Flow

When included in the model design, the simulation results are added to the Simulation Workbook as the model is executed. At the very minimum a few key statistics such as weekly yield rates are recorded at the termination of each simulation replication. Other simulation data saved in the Simulation Workbook might include the Counters, Tallies, and Discrete Statistics accumulated by Arena.

3 ARENA SUPPORT OF VBA

It should be noted that the VBA code and its associated data are independent of the internal data structures of either Arena or Excel. The model developer must take the necessary steps to pass the required data between VBA and Arena. This separation of data structures has some advantages which compensate for Arena shortcomings. First, Arena is strictly numerically based and does not contain string information except for symbol names. Therefore in general, it is not possible for Arena to read a list of machine names and then to later generate utilization statistics for those machines along with their names. The ability of VBA to incorporate a wide range of variable types makes it very easy to circumvent this Arena limitation. Second, Arena does not control the order of its symbols unless the model developer specifically numbers each symbol. Thus, the model developer might define Arena entity attributes: A, B, and C. Arena does neither assures the ordering of the attributes nor where a new attribute Z will be placed. VBA permits the model developer to determine the location of Arena symbols. For example the VBA statement:

```
AttributeZ= SIMAN.SymbolNumber("Z")
```

assigns the location of Arena Entity attribute Z to the VBA variable AttributeZ. The value of the Arena entity's attribute Z can be set to 10.5 using the following VBA statement:

```
SIMAN.EntityAttribute(CurrentEntity,
AttributeZ)=10.5
```

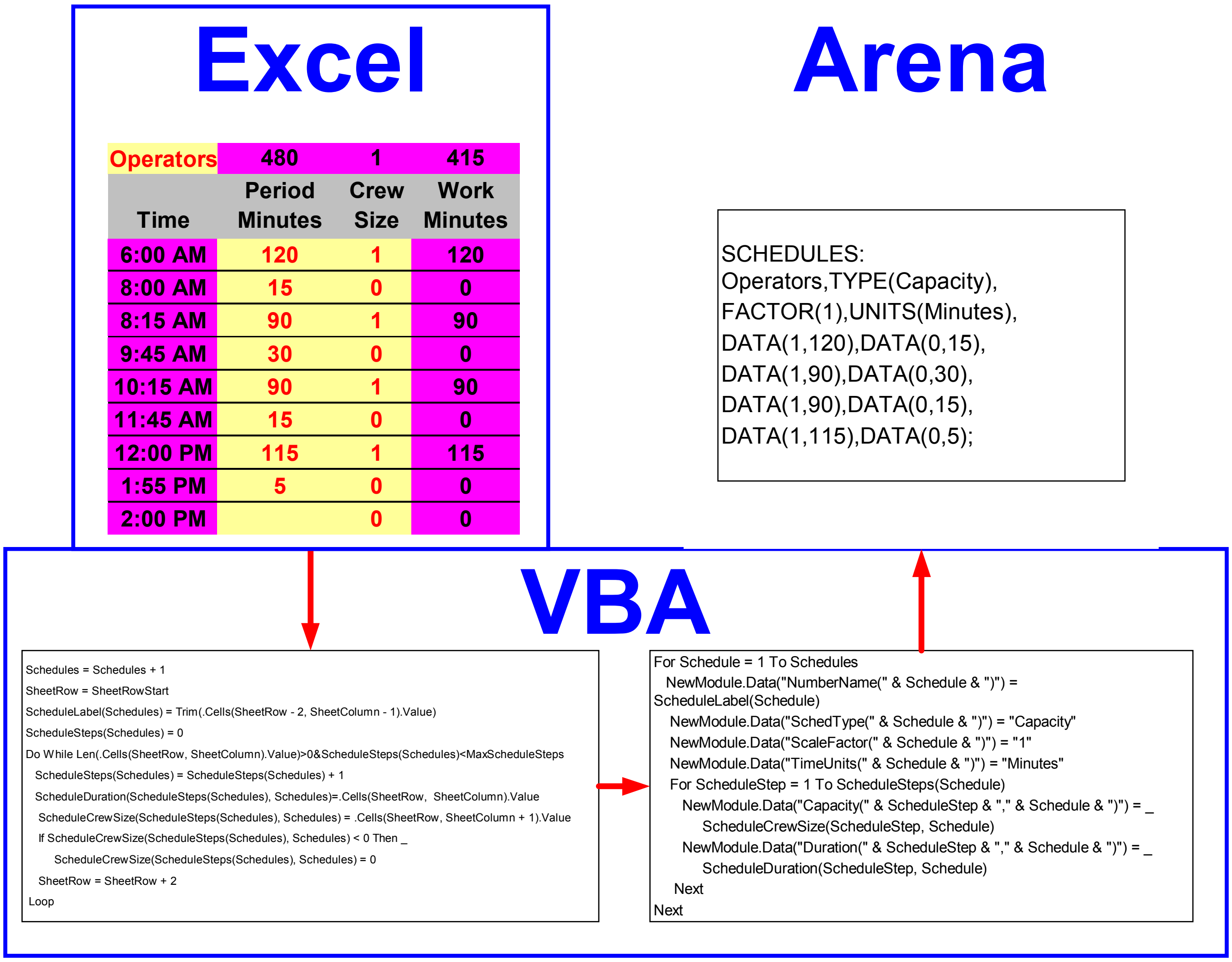
In the above example, SymbolNumber and EntityAttribute are Members of Arena VBA Object model provided by Rockwell Software (Systems Modeling Corporation) and available by pressing the F2 key.

4 MODEL DATA FLOW EXAMPLE

Figure 4 illustrates the type flow of data from Excel to Arena via VBA in an industrial strength simulation model. In this case a resource schedule over the working shift is provided in an Excel sheet. VBA reads that data and generates an Arena Schedule Module.

4.1 Simulation Workbook Sheet

The Excel data is contained in the Schedule Sheet of the Simulation Workbook. The data displayed in red can be changed by the user. In this case the data represents work and break periods for operators on an 8-hour shift spanning the period from 6:00 AM to 2:00 PM. This data input is relatively straight forward. The Excel workbook can be structured to include data entry helps, such as the total shift length (480 minutes) and the total working period (415 minutes).



Operators	480	1	415
Time	Period Minutes	Crew Size	Work Minutes
6:00 AM	120	1	120
8:00 AM	15	0	0
8:15 AM	90	1	90
9:45 AM	30	0	0
10:15 AM	90	1	90
11:45 AM	15	0	0
12:00 PM	115	1	115
1:55 PM	5	0	0
2:00 PM		0	0

```
SCHEDULES:
Operators,TYPE(Capacity),
FACTOR(1),UNITS(Minutes),
DATA(1,120),DATA(0,15),
DATA(1,90),DATA(0,30),
DATA(1,90),DATA(0,15),
DATA(1,115),DATA(0,5);
```

```
Schedules = Schedules + 1
SheetRow = SheetRowStart
ScheduleLabel(Schedules) = Trim(.Cells(SheetRow - 2, SheetColumn - 1).Value)
ScheduleSteps(Schedules) = 0
Do While Len(.Cells(SheetRow, SheetColumn).Value)>0&ScheduleSteps(Schedules)<MaxScheduleSteps
  ScheduleSteps(Schedules) = ScheduleSteps(Schedules) + 1
  ScheduleDuration(ScheduleSteps(Schedules), Schedules)=.Cells(SheetRow,  SheetColumn).Value
   ScheduleCrewSize(ScheduleSteps(Schedules), Schedules) = .Cells(SheetRow, SheetColumn + 1).Value
   If ScheduleCrewSize(ScheduleSteps(Schedules), Schedules) < 0 Then _
       ScheduleCrewSize(ScheduleSteps(Schedules), Schedules) = 0
   SheetRow = SheetRow + 2
 Loop
```

```
For Schedule = 1 To Schedules
  NewModule.Data("NumberName(" & Schedule & ")") =
ScheduleLabel(Schedule)
   NewModule.Data("SchedType(" & Schedule & ")") = "Capacity"
   NewModule.Data("ScaleFactor(" & Schedule & ")") = "1"
   NewModule.Data("TimeUnits(" & Schedule & ")") = "Minutes"
   For ScheduleStep = 1 To ScheduleSteps(Schedule)
     NewModule.Data("Capacity(" & ScheduleStep & "," & Schedule & ")") = _
        ScheduleCrewSize(ScheduleStep, Schedule)
     NewModule.Data("Duration(" & ScheduleStep & "," & Schedule & ")") = _
        ScheduleDuration(ScheduleStep, Schedule)
    Next
Next
```

Figure 4: VBA Data Exchange, Excel to Arena

4.2 VBA Code

The VBA code fragments illustrated above could either be part of the Excel workbook, the Arena model, or a stand-alone Visual Basic program. Because the VBA syntax is identical in both applications, the code location is a matter of the model designer's choice.

The VBA code fragment in the lower left of Figure 4 reads the schedule data from the Excel Simulation Workbook. The VBA code fragment in the lower right uses that data to generate the Arena Schedules Module from the Elements template.

4.3 Arena Experiment Frame

The Arena experiment frame segment illustrated in the upper right corner of Figure 4 indicates the resulting Schedules element.

4.4 Other Considerations

The process illustrated in this section can be used to generate the entire Arena experiment frame and all animation constructs. Arena data file, Elements.TXT contains a detailed description of each experiment frame element and its associated Operand Names. It is absolutely necessary that each Operand Name be correctly spelled. Arena's Operand Naming scheme is less than 100% consistent. It has been found that placing all Elements in a single submodel make them easy to locate and when necessary, to delete from the model. It is generally best to start each model generation without any prior experiment frame. This is particularly important when the new model is smaller in size than the previous model.

5 MODEL DATA CAPTURING USING VBA

Prior to the introduction of Arena 4.0 in 1999, Arena reported all of its internally generated statistics to text file, Model.OUT. This file was sequentially generated by replication but did contain total run summary statistics for Output statistics. Quite frankly the text based report format was difficult to use for analysis purposes other than manually reviewing a few numeric values. The Arena 4.0 incorporation of a Crystal Reports® database has been an initial step in eliminating this shortcoming; however it is still far from a completely functional tool. Again VBA can come to the rescue.

5.1 VBA Generated Text Report

In the simplest form a custom report can be generated using VBA code to extract Arena statistics and then to write the text based reports. For example, the following VBA code prints the average and maximum values of the fifth Arena Discrete Statistic:

```
Print #1, "Average", Siman.DStatAverage(5), _
          "Maximum",  Siman.DStatMaximum(5)
```

While the above example works well if the model developer is certain that the fifth Discrete Statistic contains the desired information, it is often best to assure the location of a particular piece of Arena information like the following:

```
DStatNumber = Siman.SymbolNumber("Main.Queue")
If DstatNumber > 0 Then
   Print #1, _
  "Average", Siman.DStatAverage(DstatNumber), _
  "Maximum", Siman.DStatMaximum(DstatNumber)
  EndIf
```

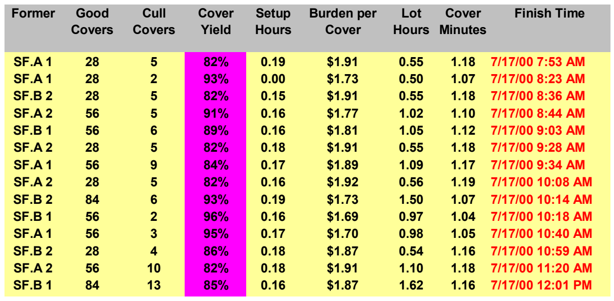
It is also important to incorporate error checking for assuring smooth model execution. The follow set of statements do a good basic job of error recovery:

```
On Error GoTo ErrorExit
   …
   Exit Sub
ErrorExit:
   MsgBox "Error #"&Err.Number&":"& _
      Err.Description & vbCrLf & _
      "psSubArenaSymbolNumberGet"
End Sub
```

5.2 VBA Saving Data to Excel

Saving both the simulation model's input data and results in the Simulation Workbook is the major advantage of using VBA. This combination of data makes the future analysis less error prone. It is harder to mistakenly make wrong conclusions when both the simulation input and output data are contained in the same workbook.

Detailed simulation data can be quickly analyzed when saved in the Simulation Workbook. Figure 5 illustrates a sequence of simulation events, in this case detailed statistics on the completion of each production lot. The simulation user can quickly locate specific data using Excel's filtering techniques. For example only the information for Former SF.B2 with 84 good covers can be selected with only four mouse clicks. This same Excel worksheet also contains a preformatted graph that is illustrated in Figure 6. In this case the Burden per Cover (variable to be minimized) is plotted verses the production lot size.



Former	Good Covers	Cull Covers	Cover Yield	Setup Hours	Burden per Cover	Lot Hours	Cover Minutes	Finish Time
SF.A 1	28	5	82%	0.19	$1.91	0.55	1.18	7/17/00 7:53 AM
SF.A 1	28	2	93%	0.00	$1.73	0.50	1.07	7/17/00 8:23 AM
SF.B 2	28	5	82%	0.15	$1.91	0.55	1.18	7/17/00 8:36 AM
SF.A 2	56	5	91%	0.16	$1.77	1.02	1.10	7/17/00 8:44 AM
SF.B 1	56	6	89%	0.16	$1.81	1.05	1.12	7/17/00 9:03 AM
SF.A 2	28	5	82%	0.18	$1.91	0.55	1.18	7/17/00 9:28 AM
SF.A 1	56	9	84%	0.17	$1.89	1.09	1.17	7/17/00 9:34 AM
SF.A 2	28	5	82%	0.16	$1.92	0.56	1.19	7/17/00 10:08 AM
SF.B 2	84	6	93%	0.19	$1.73	1.50	1.07	7/17/00 10:14 AM
SF.B 1	56	2	96%	0.16	$1.69	0.97	1.04	7/17/00 10:18 AM
SF.A 1	56	3	95%	0.17	$1.70	0.98	1.05	7/17/00 10:40 AM
SF.B 2	28	4	86%	0.18	$1.87	0.54	1.16	7/17/00 10:59 AM
SF.A 2	56	10	82%	0.18	$1.91	1.10	1.18	7/17/00 11:20 AM
SF.B 1	84	13	85%	0.16	$1.87	1.62	1.16	7/17/00 12:01 PM

Figure 5: Detailed Simulation Data Generated by VBA

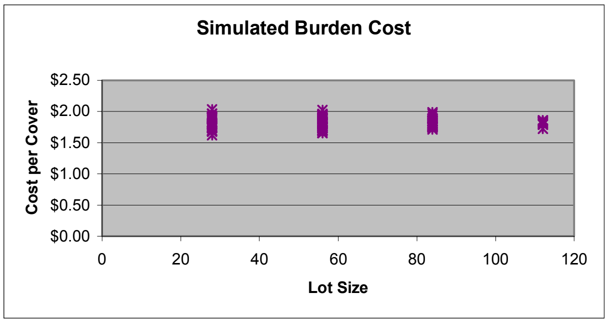


Figure 6: Automated Excel Graph

5.2.1 Using the VBA Block to Save Event Data

The method used to save the data illustrated in Figure 5 and plotted in Figure 6 is illustrated below. The Simulation Workbook contains a worksheet named, Default, which includes the basic structure and the graph. The following VBA segment copies the Default worksheet, renames it Week #, and writes appropriate header information:

```
With smXL.Workbooks(FileNumber)
  .Worksheets("Default").Copy Before:= _
  smXL.Workbooks(FileNumber).Sheets(Replication)
  .Worksheets("Default (2)").Name=SheetWeek
  .Worksheets(SheetWeek).Cells(2,8)Value = _
  SimulationStartDate+DaysPerWeek*(Replication-1)
End With
```

Data are recorded in the worksheet whenever an entity enters a VBA block using the following VBA code:

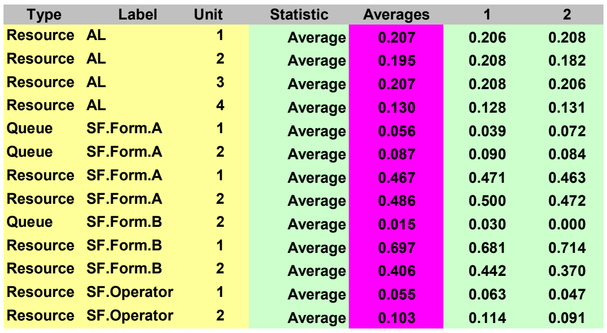
```
With smXL.Workbooks(FileNumber). _
      Worksheets(SheetWeek)
    .Cells(SheetRow,SheetColumn).Value = _
      Siman.RunCurrentTime / HoursPerDay
End With
```

In the example above the current simulation time is stored in the appropriate worksheet cell. At the end of the replication, excess worksheet rows are eliminated using the following VBA code:

```
With smXL.Workbooks(FileNumber). _
       Worksheets(SheetWeek)
    .Rows(SheetRow&”:”&SheetRowRows).Delete _
       Shift:=xlUp
End With
```

5.2.2 Using VBA to Save Arena Discrete Statistics

The various statistics accumulated and reported by Arena can be used to generate worksheet data as illustrated in Figure 7. This format of simulation result data allows for quick analysis because the information for all simulation replications is arranged by column along with the overall run average. Excel filtering allows for the isolation of statistics by catalog such as Resource or Queue.



Type	Label	Unit	Statistic	Averages	1	2
Resource	AL	1	Average	0.207	0.206	0.208
Resource	AL	2	Average	0.195	0.208	0.182
Resource	AL	3	Average	0.207	0.208	0.206
Resource	AL	4	Average	0.130	0.128	0.131
Queue	SF.Form.A	1	Average	0.056	0.039	0.072
Queue	SF.Form.A	2	Average	0.087	0.090	0.084
Resource	SF.Form.A	1	Average	0.467	0.471	0.463
Resource	SF.Form.A	2	Average	0.486	0.500	0.472
Queue	SF.Form.B	2	Average	0.015	0.030	0.000
Resource	SF.Form.B	1	Average	0.697	0.681	0.714
Resource	SF.Form.B	2	Average	0.406	0.442	0.370
Resource	SF.Operator	1	Average	0.055	0.063	0.047
Resource	SF.Operator	2	Average	0.103	0.114	0.091

Figure 7: Discrete Statistics Worksheet

The worksheet illustrated in Figure 7 is generated in a manner similar to that outlined above, a prototype worksheet that contains the default formatting is included in the Simulation Workbook. At the beginning of the simulation run, the prototype worksheet is copied and expanded to include a row for each Discrete Statistic element. The model resources and queues are then identified. At the conclusion of each simulation replication the individual statistics are added to the worksheet using the following VBA code:

```
With smXL.Workbooks(FileNumber)
    .Worksheets(“Dstat”).
    For DStats = 1 To CountDStats
       SheetRow = SheetRow + 1
       .Cells(SheetRow,SheetColumn).Value = _
            Siman.DStatAverage(DStats)
    Next
End With
```

6 CONCLUSIONS

This paper and the associated tutorial has outlined the process by which VBA can be utilized in the development of Industrial Strength Arena simulation models. None of these techniques are currently documented in detail. The best method to learn how to incorporate VBA is by review working examples. Rockwell Software (Systems Modeling Corporation) has developed an extensive series of mini-models and includes them with their distribution software as Smart Files. Productive Systems has also developed a wide range of working models using VBA/Excel techniques and provides that code to its clients.

REFERENCES

Getz, K., and M. Gilbert. 1997. *VBA developer’s handbook*™. Sybex, Inc., San Francisco.

Lomax, P., 1998. *VB & VBA in a nutshell: The language*. O’Reilly & Associates, Inc., Sebastopol, CA.

Seppanen, M.S., 1998. Designing simulation models to use visual basic for applications. In *Proceedings of the 1998 European Simulation MultiConference*, ed. R. Zobel and D. Moeller, 100-104. Society for Computer Simulation International, Manchester, UK.

Seppanen, M.S., 1996. A database approach to simulation modeling. In *Proceedings of the 1996 European Simulation Symposium*, 332-335 (Budapest, Hungary, June 2-6). SCS.

Seppanen, M.S., 1995. Developing industrial strength simulation models. In *Proceedings of the 1995 Winter Simulation Conference*, ed., Christos Alexopoulos, Keebom Kang, William Lilegdon, and David Goldsman, 936-939. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Systems Modeling Corporation, 1999. *Arena online help and smart models*, Sewickley, PA.

Systems Modeling Corporation, 1999. *Arena user’s guide*, Sewickley, PA.

Wells, E., 1995. Developing Microsoft® Excel 95 solution with Visual Basic® for Applications. Microsoft Press, Redmond, WA.

AUTHOR BIOGRAPHY

MARVIN SEPPANEN has been a user of SIMAN/Arena since 1983. His company, Productive Systems of Winona, Minnesota, is an independent Industrial Engineering consulting firm specializing in simulation modeling and analysis of manufacturing systems. He holds his BME, MSIE, and Ph.D. (Operations Research) degrees from the University of Minnesota. Before starting Productive Systems he was an Associate Professor of Industrial Engineering at General Motors Institute and The University of Alabama. He is a Registered Professional Engineer; Senior Member and chapter officer, Institute of Industrial Engineers; Member, The Society for Computer Simulation; Member, Society of Manufacturing Engineers; Member, American Society for Control; and is certified at

the Fellow Level by the American Production and Inventory Control Society. He teaches Arena/Visio based simulation to Management and Manufacturing Systems Engineering graduate students at the University of St. Thomas in St. Paul, Minnesota. Dr. Seppanen can be reached at `<seppanen@hbci.com>`.