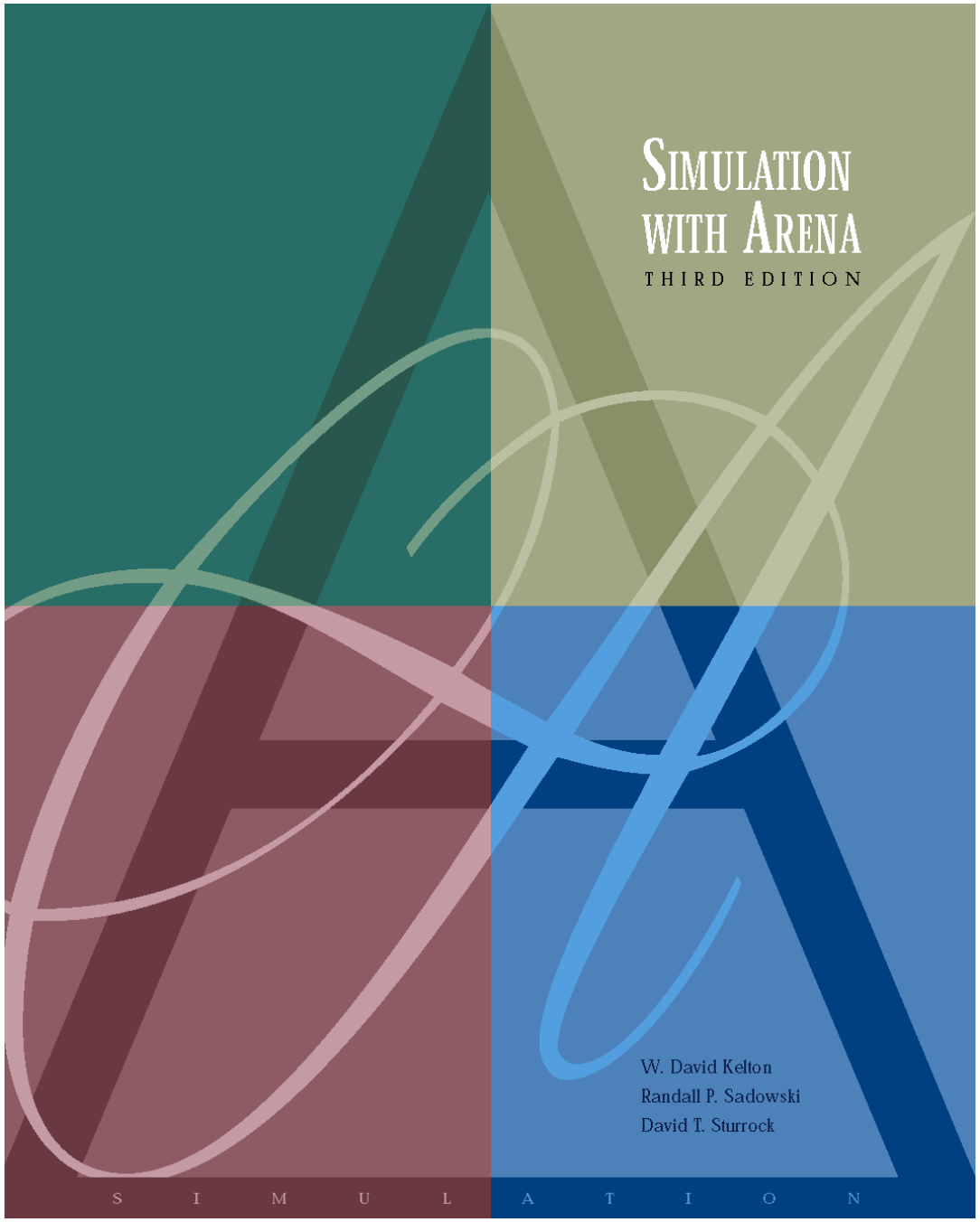


Arena Integration and Customization

Chapter 10

Last revision July 14, 2003

What We'll Do ...

- **Reading and Writing Data Files (ReadWrite)**
- **ActiveX™ and Visual Basic® for Applications (VBA)**
- **Creating Modules with Arena Professional Edition**

Reading and Writing Data Files

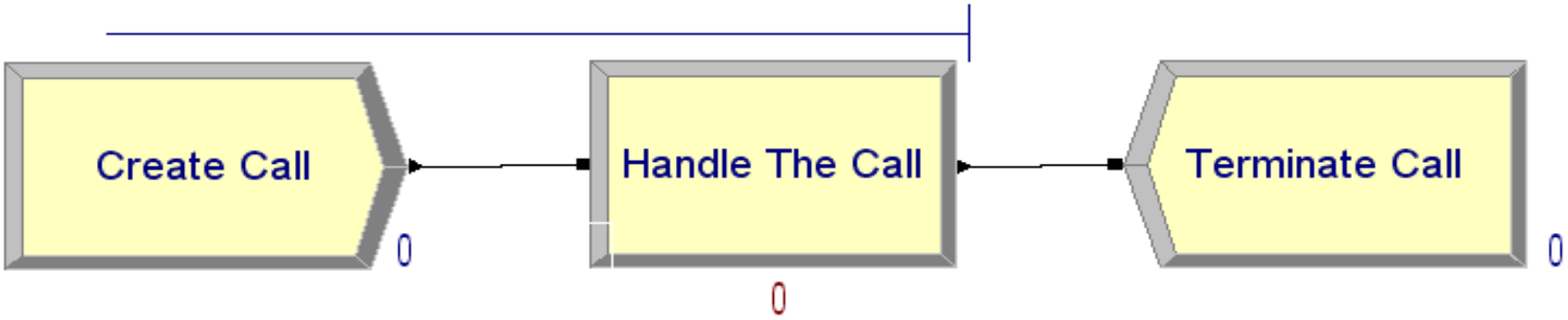
- **Reading entity arrivals from a text file**
- **Reading and writing Microsoft Access and Excel files**
- **Advanced reading and writing**

Reading Entity Arrivals From a Text File

- **Why data-driven simulations?**
  - Model validation
  - Evaluating how a particular scenario is handled
  - Modeling a specific arrival pattern
  - Assumes historical data exist and can be transformed for use in simulation

Simple Call Center Model

- **Single call stream**
- **Single agent resource**
- **Random call processing time**



External Call Center Data

- **Historical call arrival times**
- **Model 10-02 Input.txt**
  - ASCII file (e.g., Notepad, saved as text from Excel)
  - Absolute simulation arrival times

    1.038457

    2.374120

    4.749443

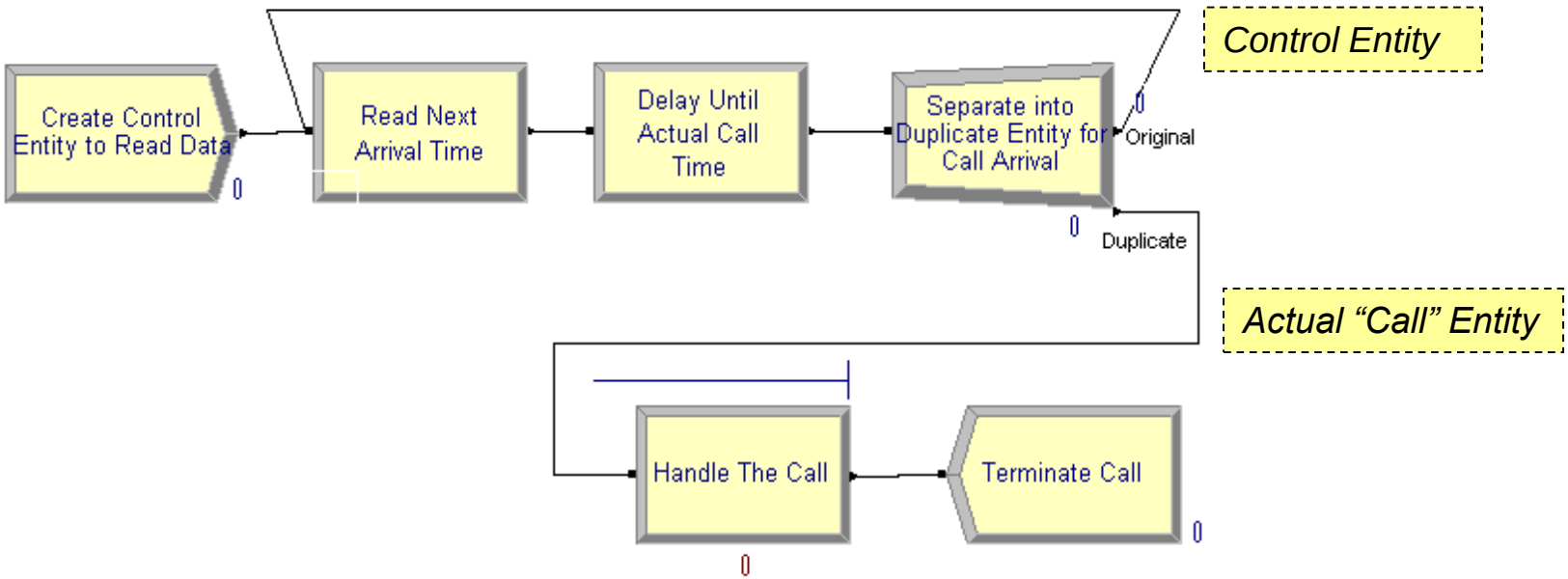
    9.899661

    10.525897

    17.098860

Model Logic to Read Data

- **Can't use simple time between arrivals**
- **Control entity**
  - Create only one
  - Duplicate to send actual "call" entity into model



Model Logic to Read Data (cont'd.)

- **ReadWrite module (Advanced Process)**
  - *Arena File Name*: description (actual disk filename is specified in File module)
  - *Assignments*: model variables/attributes to be assigned based on data read from file (*Call Start Time* attribute)
- **Delay/Duplicate Logic**
  - File contains "absolute" times; Delay module holds entity for a time *interval*
  - Delay control entity for interval until actual arrival time of call (*Call Start Time - TNOW*)
  - Create a duplicate (Separate module) to dispatch actual call into model. Original entity loops back to read next time.

Model Logic to Read Data (cont'd.)

- **File data module (Advanced Process)**
  - *Name*: Name referenced in other Arena modules.
  - *Access Type:* Sequential indicates to read in order.
  - *Operating System File Name*: The name used by file system. May be relative or fully qualified.
  - *End of File Action*: What to do when all records are read.

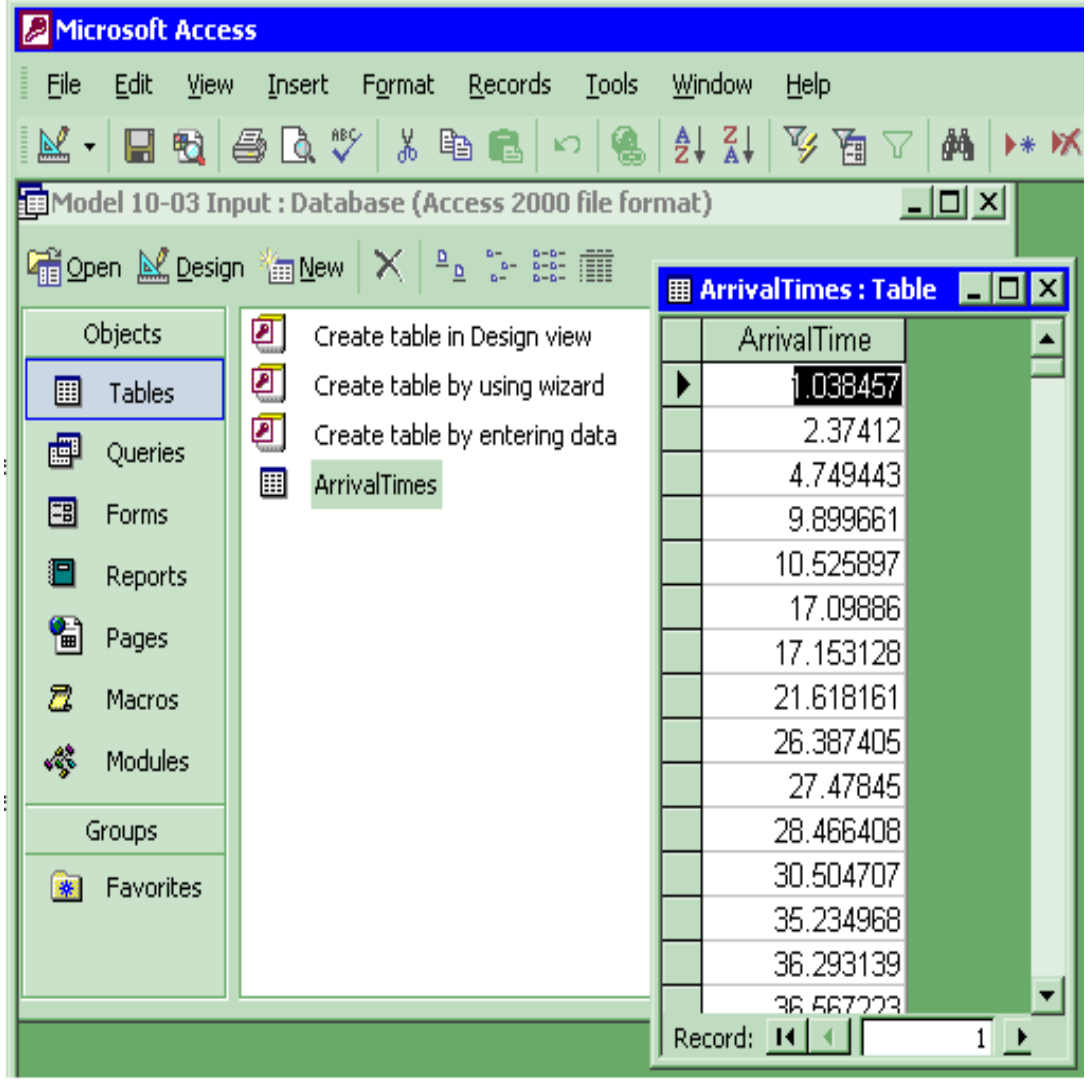
	Name	Access Type	Operating System File Name	Structure	End of File Action	Initialize Option	Comment Character
1	Arrivals File	Sequential File	Model 10-02 Input.txt	Free Format	Dispose	Hold	No

Run Termination

- **Run Setup options**
  - Maximum replications / simulation end time always terminates the simulation run.
- **System empties**
  - If no entities on calendar and no other time-based controls, run may terminate earlier than setup options dictate.
  - The control entity is disposed after it reads the last data value.

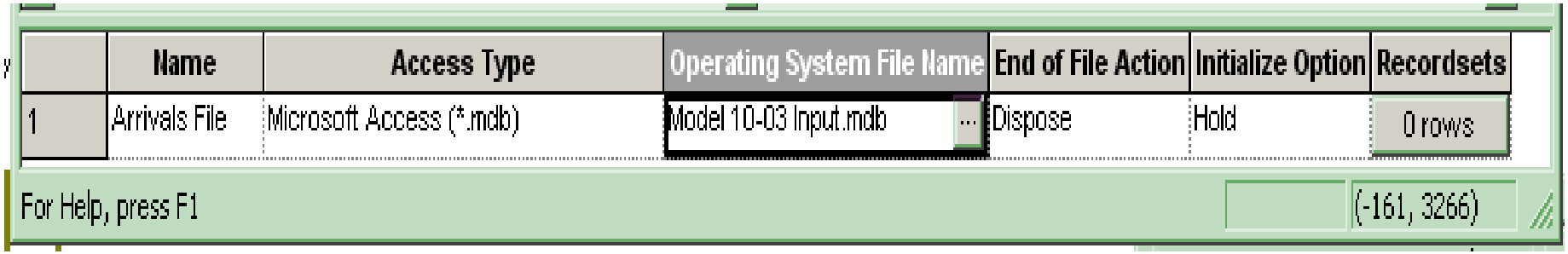
Reading Access Files

- **Sample Access data**
  - Model 10-03 Input.mdb
  - Table: ArrivalTimes



Reading Access Files

- **File data module (Advanced Process)**
  - *Access Type*: Microsoft Access (*.mdb)
  - *Operating System File Name*: Model 10-03 Input.mdb
  - *Recordsets*: Click to load the Recordsets Editor
- **Important:**
  - Never name an access file the same as the model name or it will conflict with the automatic output database file.

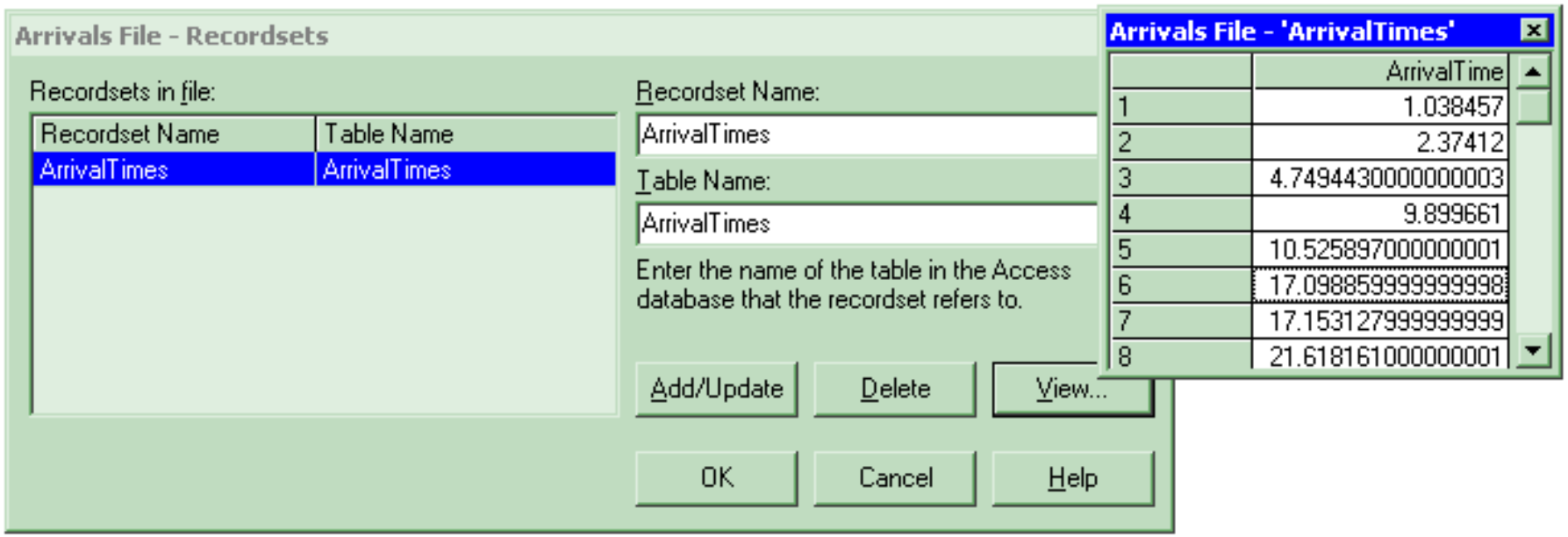


	Name	Access Type	Operating System File Name	End of File Action	Initialize Option	Recordsets
1	Arrivals File	Microsoft Access (*.mdb)	Model 10-03 Input.mdb	Dispose	Hold	0 rows

For Help, press F1 (-161, 3266)

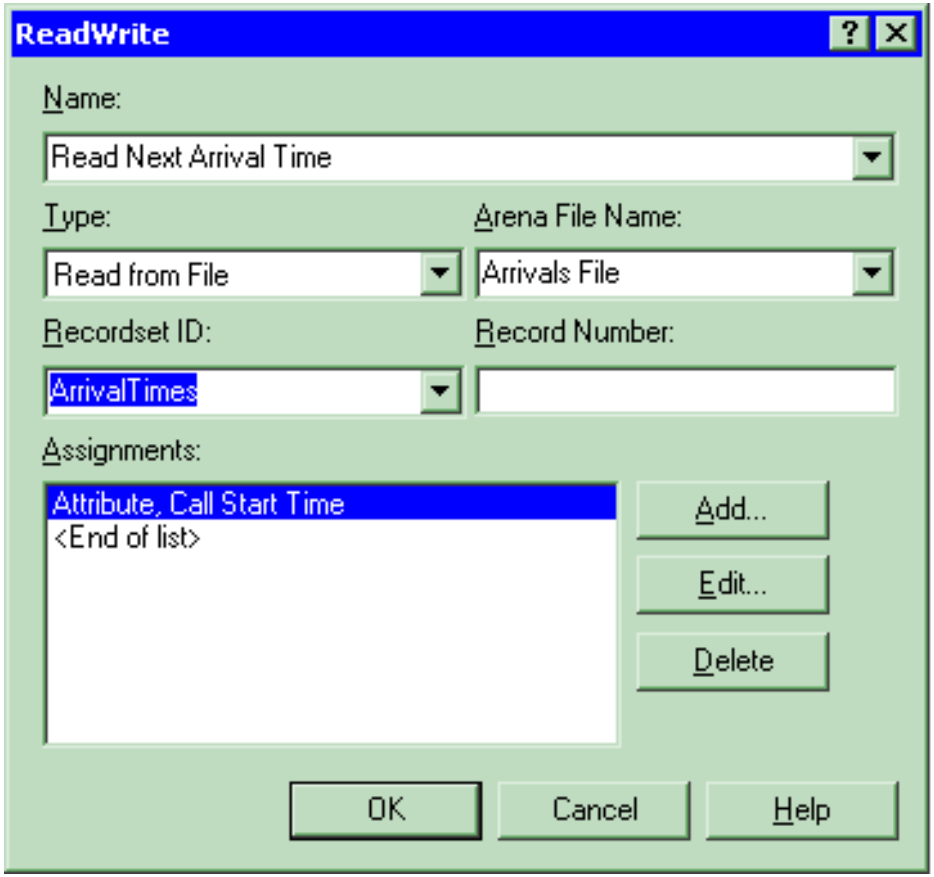
Reading Access Files

- **Recordsets Editor**
  - Associates a recordset name with a table
  - Table must already exist
  - *View* allows you to see a sample of the real data



Reading Access Files

- **ReadWrite module (Advanced Process)**
  - *Recordset ID*: Same as defined in Recordsets Editor

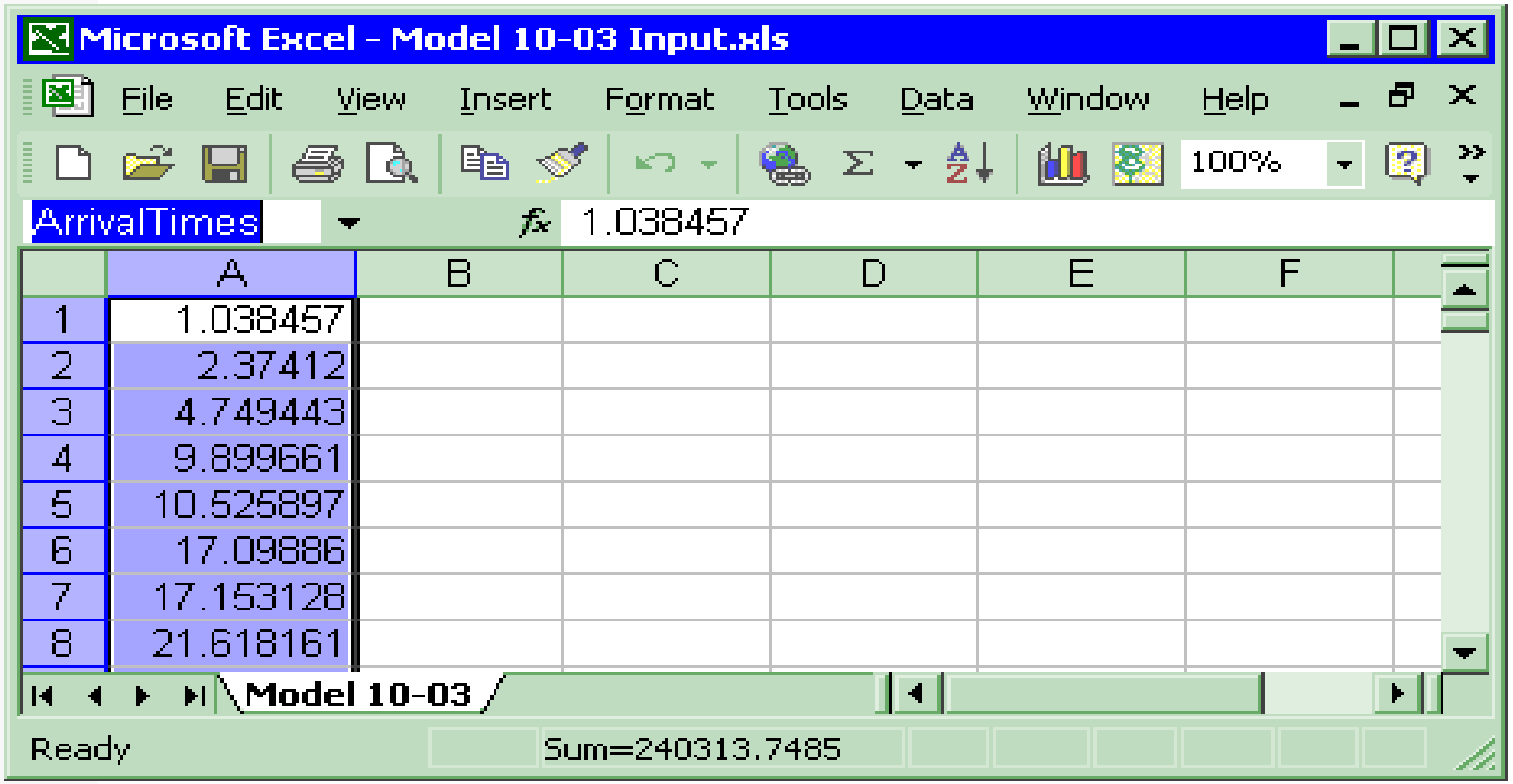


Reading Excel Files

- **Excel is not a relational database but has many similarities:**
  - *An Excel workbook is similar to an Access database file.*
  - *The rows and columns in a rectangular named range in an Excel worksheet are similar to the rows and columns of an Access table.*

Reading Excel Files

- **Sample Excel data**
  - Model 10-03 Input.xls
  - Named Range: ArrivalTimes



	A	B	C	D	E	F
1	1.038457					
2	2.37412					
3	4.749443					
4	9.899661					
5	10.525897					
6	17.09886					
7	17.153128					
8	21.618161					

Reading Excel Files

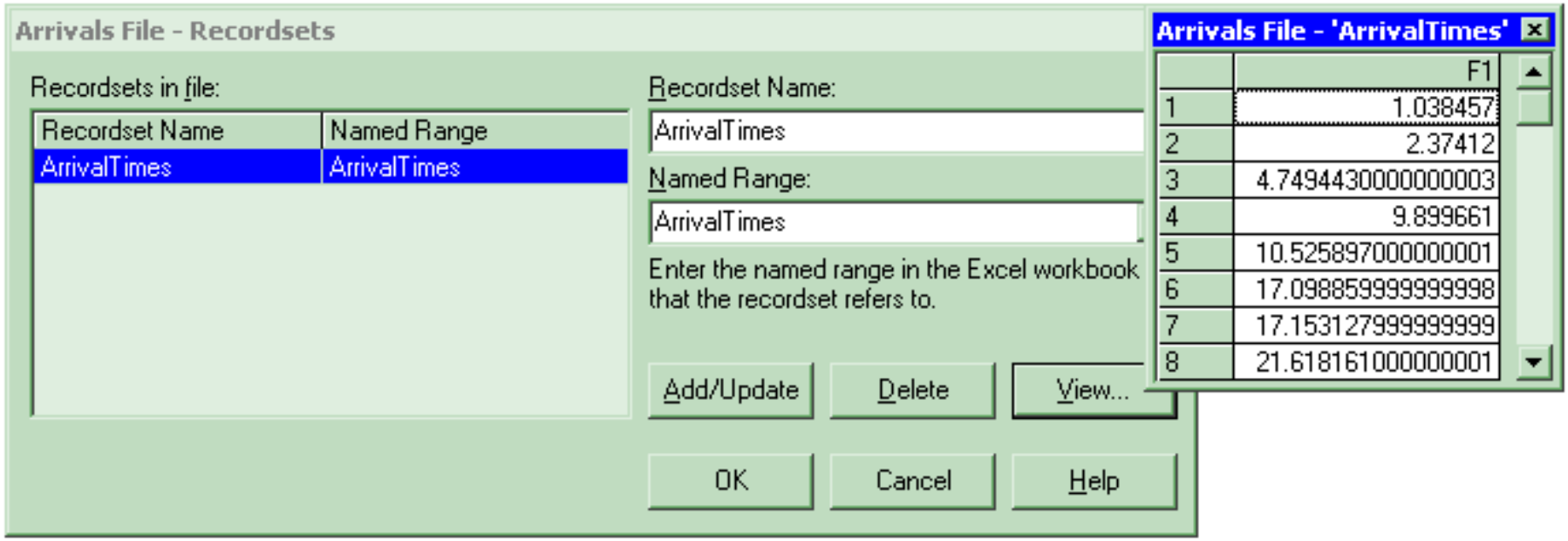
- **File data module (Advanced Process)**
  - *Access Type*: Microsoft Excel (*.xls)
  - *Operating System File Name*: Model 10-03 Input.xls
  - *Recordsets*: Click to load the Recordsets Editor

	Name	Access Type	Operating System File Name	End of File Action	Initialize Option	Recordsets
1	Arrivals File	Microsoft Excel (*.xls)	Model 10-03 Input.xls	Dispose	Hold	1 rows

For Help, press F1 (-593, 3618)

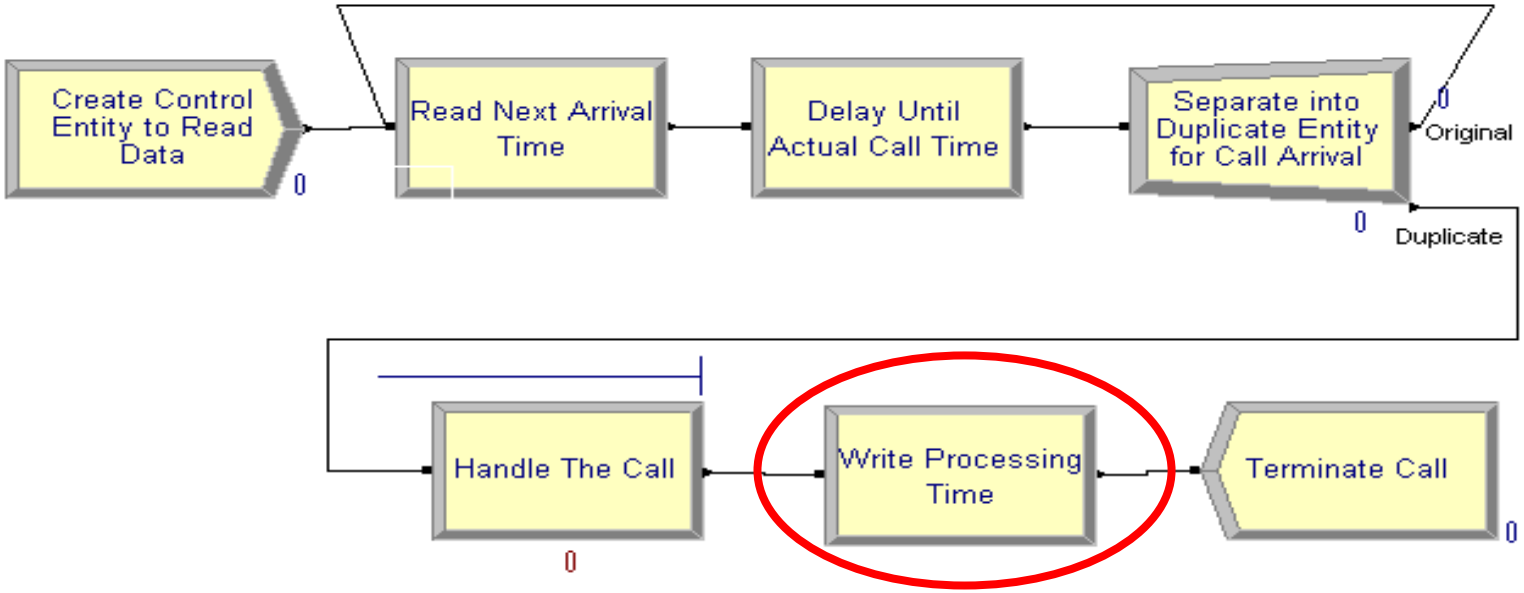
Reading Excel Files

- **Recordsets Editor**
  - Associates a recordset name with a named range
  - Named range must already exist
  - *View* allows you to see a sample of the real data



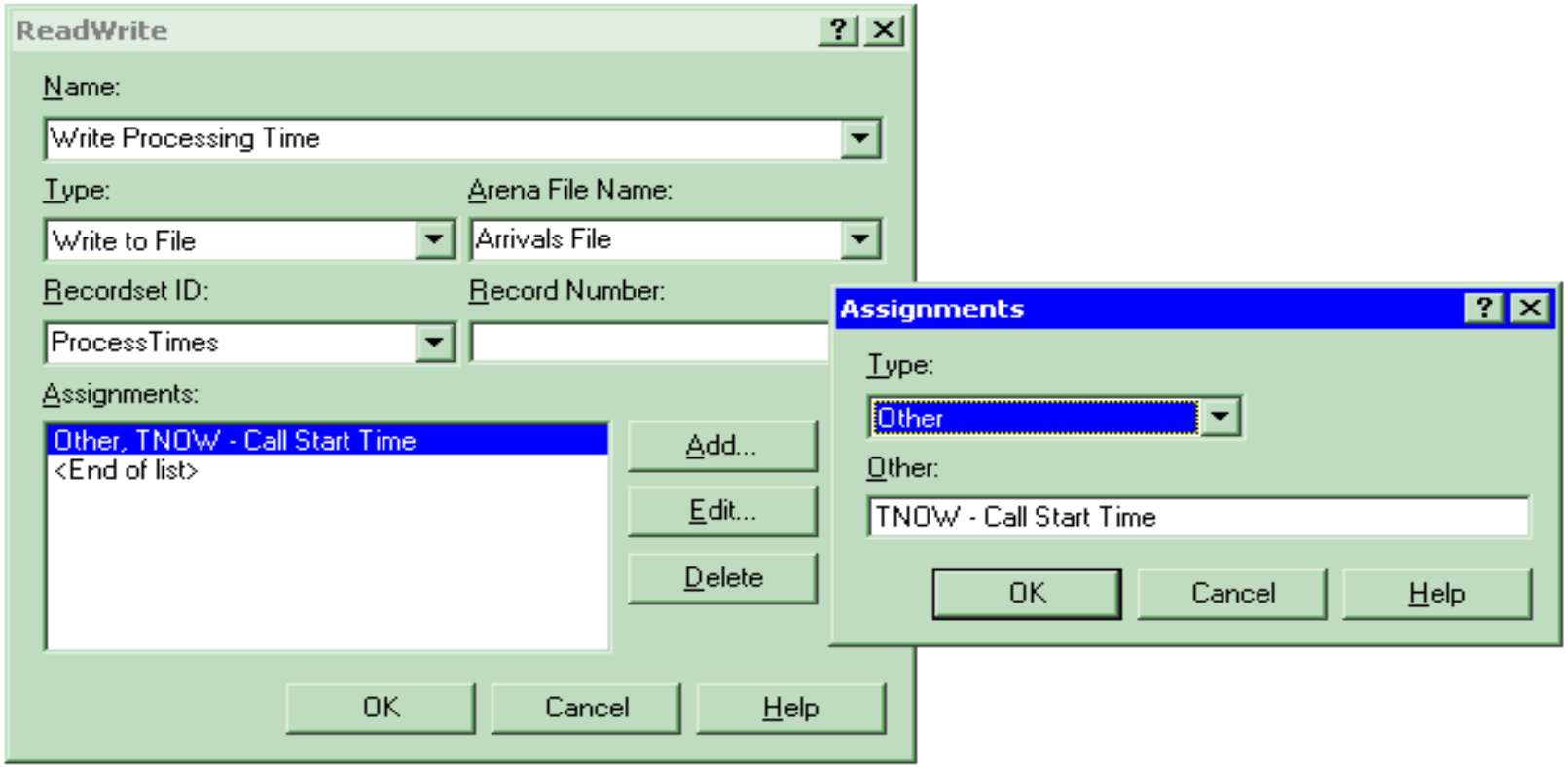
Writing Access and Excel Files

- ***The file:***
  - *The table or named range must already exist.*
  - *An Excel named range should be formatted as numeric.*
- **ReadWrite module:**
  - Add new module to specify which data to write.



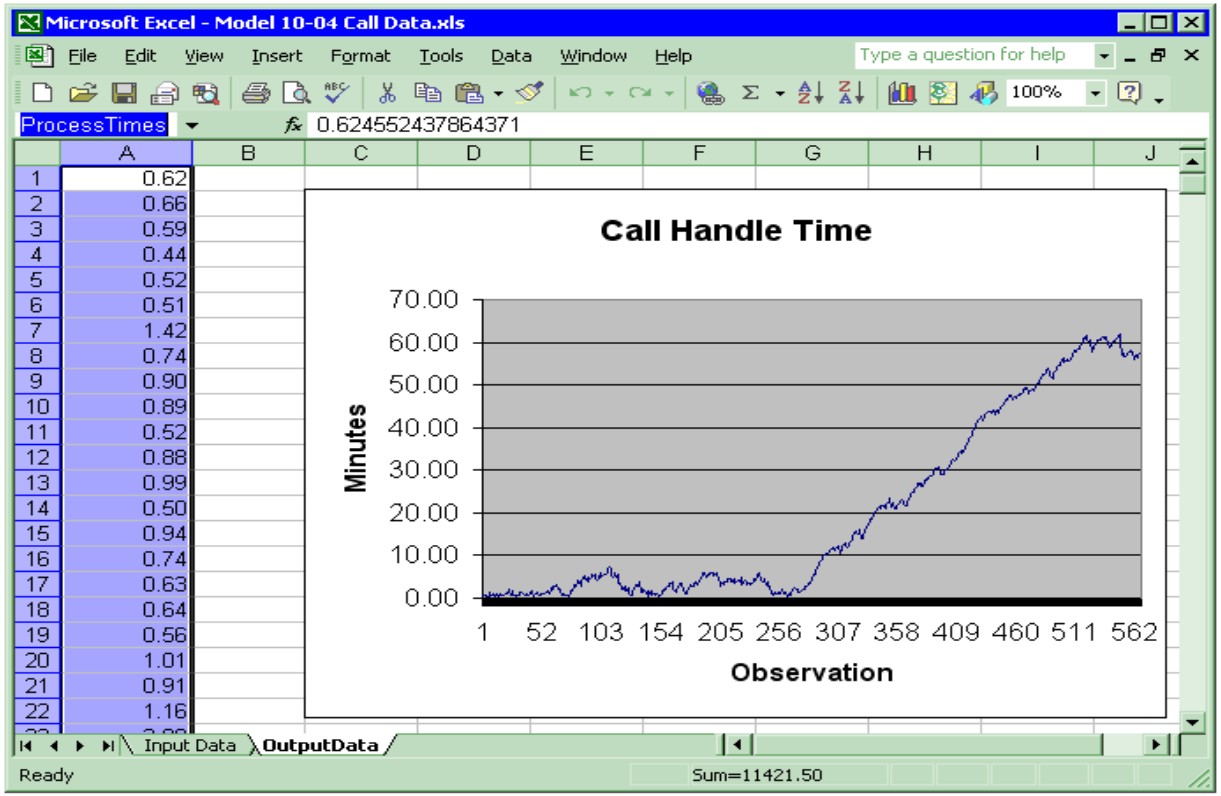
Writing Access and Excel Files

- **ReadWrite module:**
  - Use *Type* of *Write To File*
  - *Use Recordset ID as before.*



Writing Access and Excel Files

- **Spreadsheet options:**
  - You may predefine a plot on the named range and the plot will be built dynamically as data is added to the file.



Advanced Reading and Writing

- **Formatting can be used to handle text files with fields not delimited by spaces:**

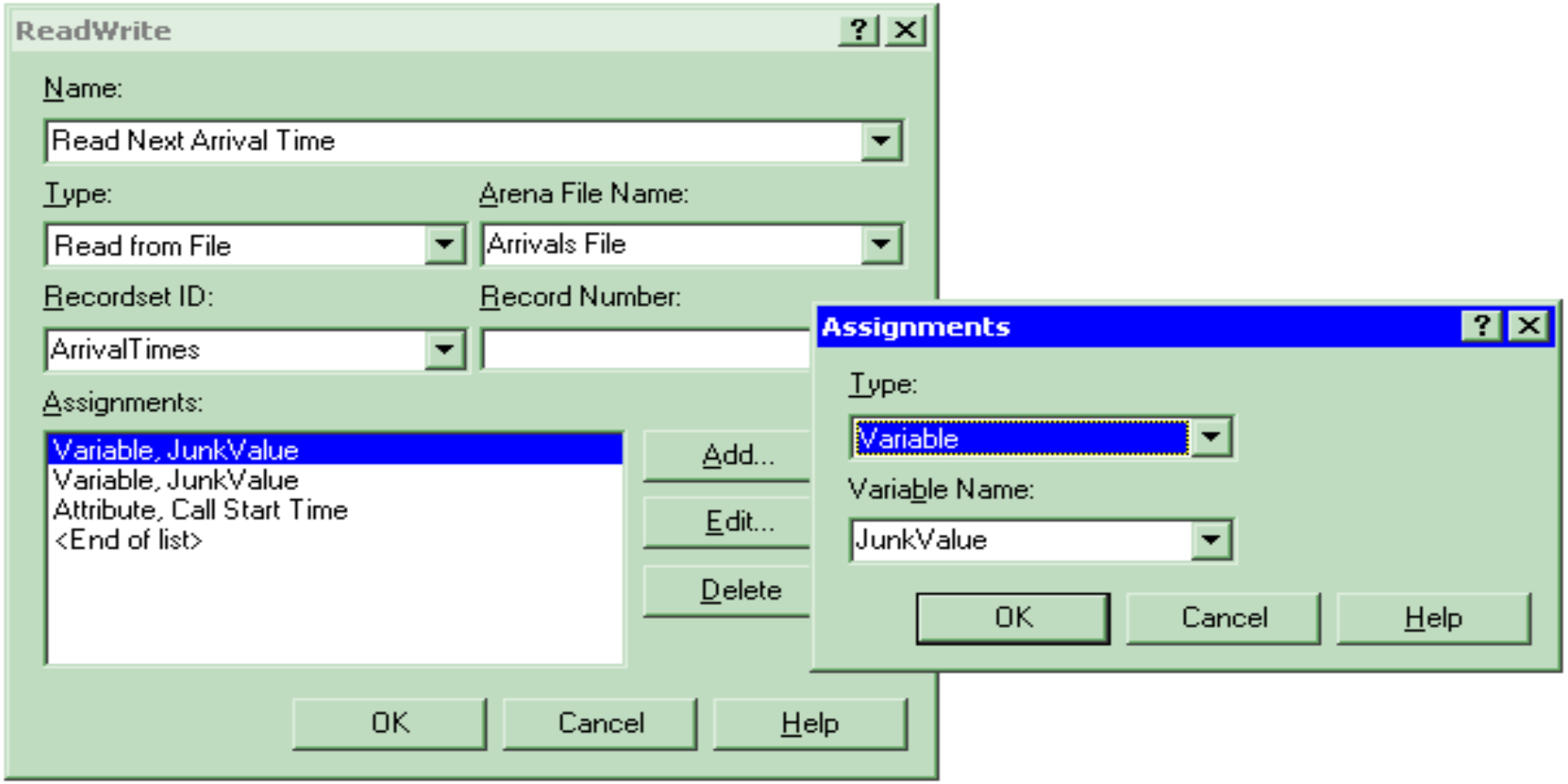
```
Part 1  1.038
Part 27 2.374
Part 5194.749
Part 67 9.899
Part 72 10.52
Part 16217.09
```

	Name	Access Type	Operating System File Name	Structure	End of File Action	Initialize Option	Comment Character
1	Arrivals	Sequential File	Model 10-02 Input.txt	"(8x,f8.3)"	Dispose	Hold	No

For Help, press F1 (145, 4932)

Advanced Reading and Writing

- **Skip columns in Access & Excel by using dummy variables:**



Advanced Reading and Writing

- **Advanced data access is available using *Access Type* of *Active Data Objects (ADO)* and a *Connection String:***
  - ***Excel With Headings Using ADO***

    Provider=Microsoft.JET.OLEDB.4.0;
    Data Source=C:\Documents\Book1.xls;
    Extended Properties=""Excel 8.0; HDR=Yes;""

  - ***SQL Commands Using ADO***

    Driver={SQL Server};
    Server=RSI-Joe; Database=BizBikes;
    Uid=BizWareUser; pwd=MyPassword

Use two double quotes for each embedded double quote

What We'll Do ...

- Reading and Writing Data Files (ReadWrite)
- ActiveX™ and Visual Basic® for Applications (VBA)
- Creating Modules with Arena Professional Edition

ActiveX Automation

- **Program applications to "automate" tasks**
  - Act on themselves (e.g., macros in Excel)
  - Act on other applications (e.g., Arena creating Excel file)
- **External programming languages**
  - C++, Visual Basic®, Java, etc.
- **Visual Basic for Applications (VBA) programming embedded in application**
  - Microsoft Office®, Visio®, AutoCAD®, Arena®, …
- **Both types work together (e.g., Arena VBA controlling Excel)**

Application Object Model

- **Objects: application *components* that can be controlled**
- **Properties: *characteristics* of objects**
- **Methods: *actions* performed on or by objects**

Arena Objects	Properties	Methods
Application	Visible	Show
Model	Name, State	Close, Go
View	Background Color	Zoom In
…		

Visual Basic for Applications (VBA)

- **Included with Arena**
- **Full Visual Basic programming environment**
- **Code stored with Arena model (*.doe*) file**
- **UserForms (dialogs) for custom interfaces**
- **Code-debugging tools**
- **Comprehensive online help**
- **Visual Basic Editor window: "child" of Arena (Tools/Show Visual Basic Editor)** 

Built-in Arena VBA Events

- **ThisDocument: accesses objects, events in Arena's object model**
- **Built-in VBA events: locations where VBA code can be activated**
  - Pre-run events (e.g., DocumentOpen)
  - Arena-initiated run events (e.g., RunBegin, RunEndReplication)
  - Model/user-initiated run events (e.g., UserFunction, VBA_Block_Fire)
- **Type code in Visual Basic Editor to populate an event**

Simulation Run VBA Events

- **Arena/VBA sequence of events when model runs:**

1. **RunBegin**

2. *Arena checks and initializes the model*

Module data available

3. **RunBeginSimulation**

4. **RunBeginReplication**

5. *Arena runs replication*

OnKeystroke, UserFunction, etc.

6. **RunEndReplication**

7. **RunEndSimulation**

Simulation run data available

8. *Arena terminates the simulation run*

9. **RunEnd**

Module data available

Arena's Object Model

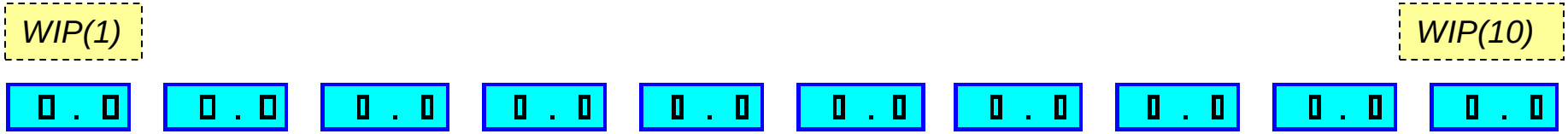
- **Model-window objects: items placed in model window, such as:**
  - Modules
  - Connections
  - Lines
- **SIMAN object: simulation run data, such as:**
  - Variable values
  - Queue lengths
  - Simulation time
- **Structural objects: access general functions**
  - Application
  - Panels

Sample: Create Ten Status Variables

```
Dim oModel As Arena.Model
Dim i As Integer
Dim nX As Long

' Add the status variables to this Arena model
Set oModel = ThisDocument.Model

nX = 0                    ' Start at x position 0
For i = 1 To 10
    ' Add a status variable to the model window
    oModel.StatusVariables.Create nX, 0, _
       nX + 400, 150, "WIP(" & i & ")", "**.*", False, _
       RGB(0, 0, 255), RGB(0, 255, 255), RGB(0, 0, 0), "Arial"
    ' Move over 500 world units for next position
    nX = nX + 500
Next i
```

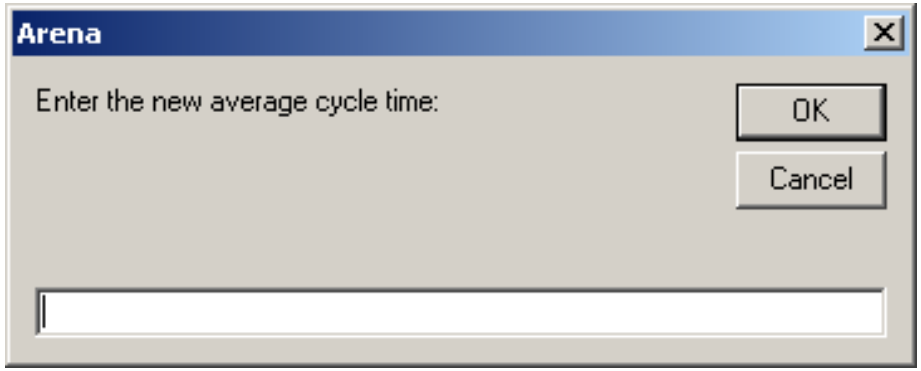


Sample: Assign Variable Value During Run

```
Dim oSIMAN As Arena.SIMAN
Dim nVarIndex As Long
Dim sNewValue As String

' Prompt for a new value
sNewValue = InputBox("Enter the new average cycle time:")

' Assign their answer to the Mean Cycle Time variable
Set oSIMAN = ThisDocument.Model.SIMAN
nVarIndex = oSIMAN.SymbolNumber("Mean Cycle Time")
oSIMAN.VariableArrayValue(nVarIndex) = sNewValue
```

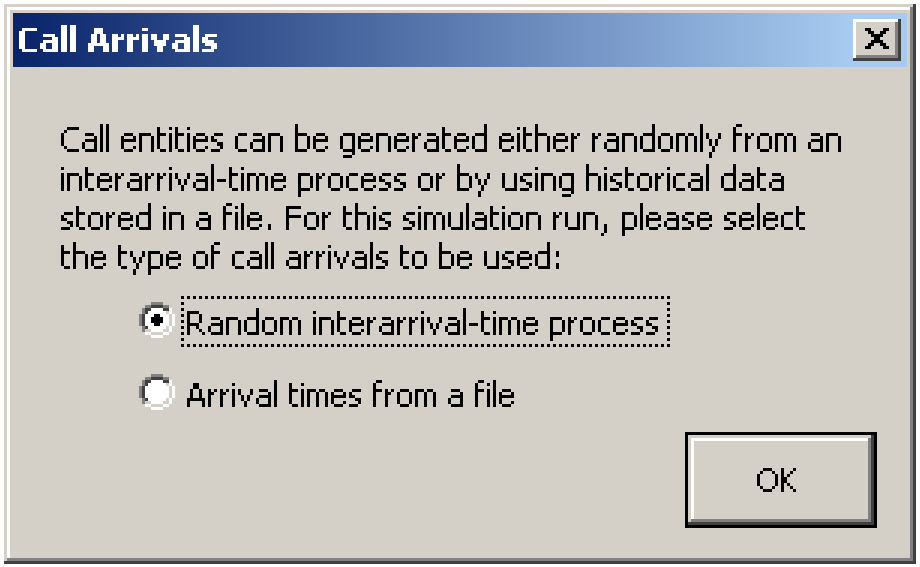


Arena Macro Recorder

- A **Macro** is a VBA function to perform a task.
- Macro recording automatically creates the VBA code to reproduce the actions you take while performing the steps in the task.
- Use the **Record Macro** toolbar to start/stop and pause/resume recording.
- Useful for automating repetitive tasks.
- Ideal for learning VBA commands and prototyping functions.

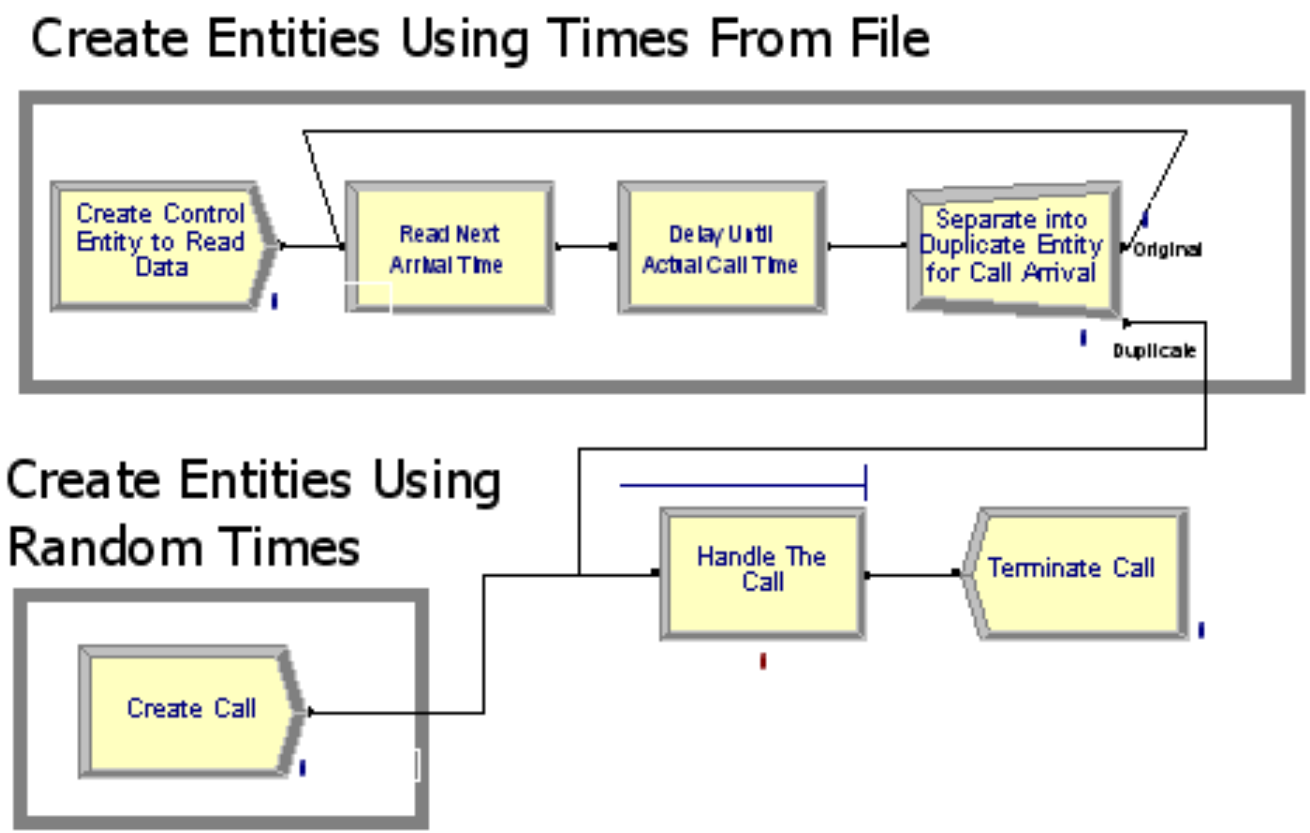
Model 10-05: Presenting Arrival Choices to the User

- **Prompt at beginning of run**
  - Generate entities via random process … or …
  - Generate based on arrival times stored in a file



Our Approach

- **Both sets of logic placed in model window and connected to start of call logic (Process module)**

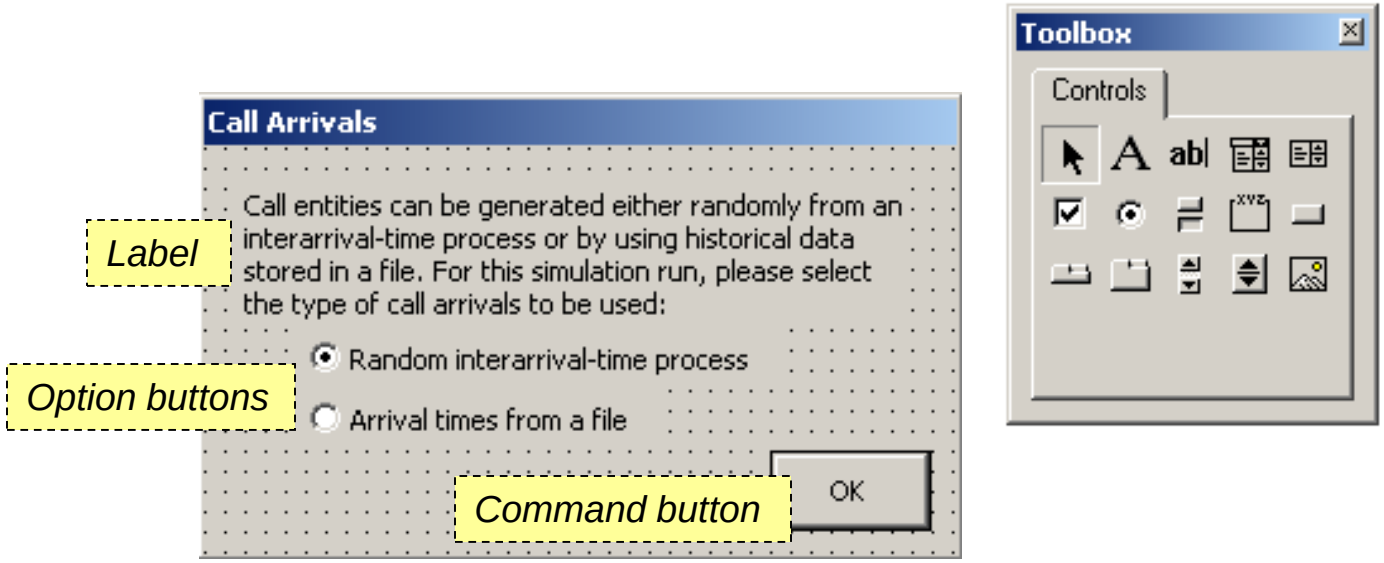


Our Approach (cont'd.)

- **Change Max Arrivals field in Create module to turn "on" or "off" its generation of entities**
- **Random interarrival-time process**
  - ***Create Call*** module: ***Infinite***
  - ***Create Control Entity to Read Data*** module: ***0***
- **Arrival times from a file**
  - ***Create Call*** module: ***0***
  - ***Create Control Entity to Read Data*** module: ***1***
- **Give unique "tag" to each Create module (so VBA code can find them)**

VBA UserForm

- **Insert/UserForm menu in Visual Basic Editor**
- **Drop controls from Control Toolbox (labels, option buttons, command button)**



Show the UserForm

- **At beginning of run (ModelLogic_RunBegin), show the form:**

```
Option Explicit
Private Sub ModelLogic_RunBegin()
    ' Display the UserForm to ask for the type of arrivals
    frmArrivalTypeSelection.Show

    Exit Sub
End Sub
```

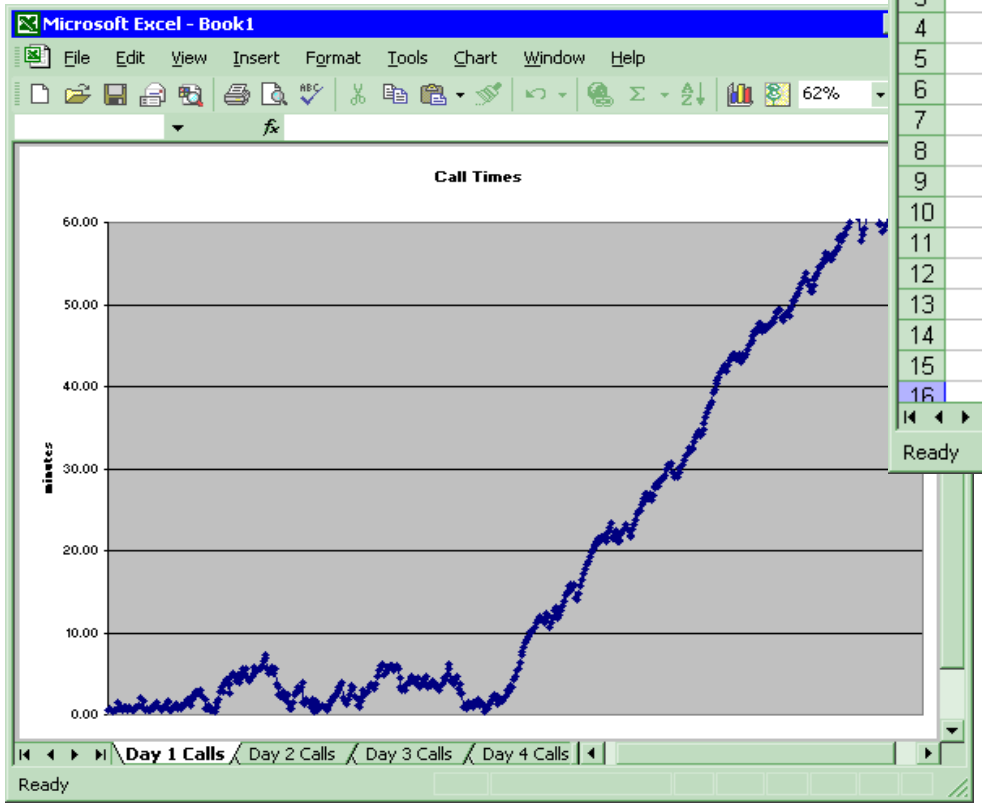
- **Program control passes to the form until it's closed**
- **Arena run "suspended" while form is in control**

Change Module Data On OK

- **When user clicks OK button on form, modify the Create module data**
  - Find the Create modules
  - Set the Max Arrivals fields
  - Play a sound
  - Close the UserForm
- **When form is closed, simulation run commences with the new data values in the Create modules**

Model 10-06: Record Call Data in Microsoft Excel

- **Our goal:**
  - Raw call data tables
  - Daily call duration charts



Microsoft Excel - Book1

File Edit View Insert Format Tools Data Window Help

100%

S16 | 0.567077248879592

	M	N	O	P	Q	R	S
1	Day 4				Day 5		
2	Start Time	End Time	Duration		Start Time	End Time	Duration
3	1.04	1.71	0.67		1.04	1.92	0.88
4	2.37	3.44	1.06		2.37	3.33	0.96
5	4.75	5.78	1.03		4.75	5.56	0.81
6	9.90	10.79	0.89		9.90	10.72	0.82
7	10.53	11.26	0.74		10.53	11.46	0.93
8	17.10	18.08	0.98		17.10	17.91	0.81
9	17.15	18.90	1.75		17.15	18.36	1.21
10	21.62	22.48	0.86		21.62	22.23	0.61
11	26.39	27.08	0.69		26.39	26.96	0.57
12	27.48	28.23	0.76		27.48	28.10	0.62
13	28.47	29.27	0.80		28.47	29.30	0.83
14	30.50	31.01	0.51		30.50	30.87	0.37
15	35.23	35.80	0.56		35.23	36.02	0.78
16	36.29	37.15	0.85		36.29	36.86	0.57

Day 1 Calls | Day 2 Calls | Day 3 Calls | D

Ready

Using ActiveX Automation in VBA

- **Reference the Excel Object Library**
  - Tools/References menu in Visual Basic Editor
  - Check the Microsoft Excel Object Library
  - Establishes link between Arena VBA and Excel
- **Object variables from application's object model**
  - *Excel.Application*, *Excel.Workbook*
  - *Arena.SIMAN*
- **Starting Excel**
  - *CreateObject*: starts application, returning "handle" to the program (stored in *oExcelApp* variable)
  - *oExcelApp.Workbooks.Add*: similar to "File/New" in Excel

Retrieving Simulation Data

- **ThisDocument**
  - Built-in variable accessing the Arena model
  - Use only within Arena's VBA

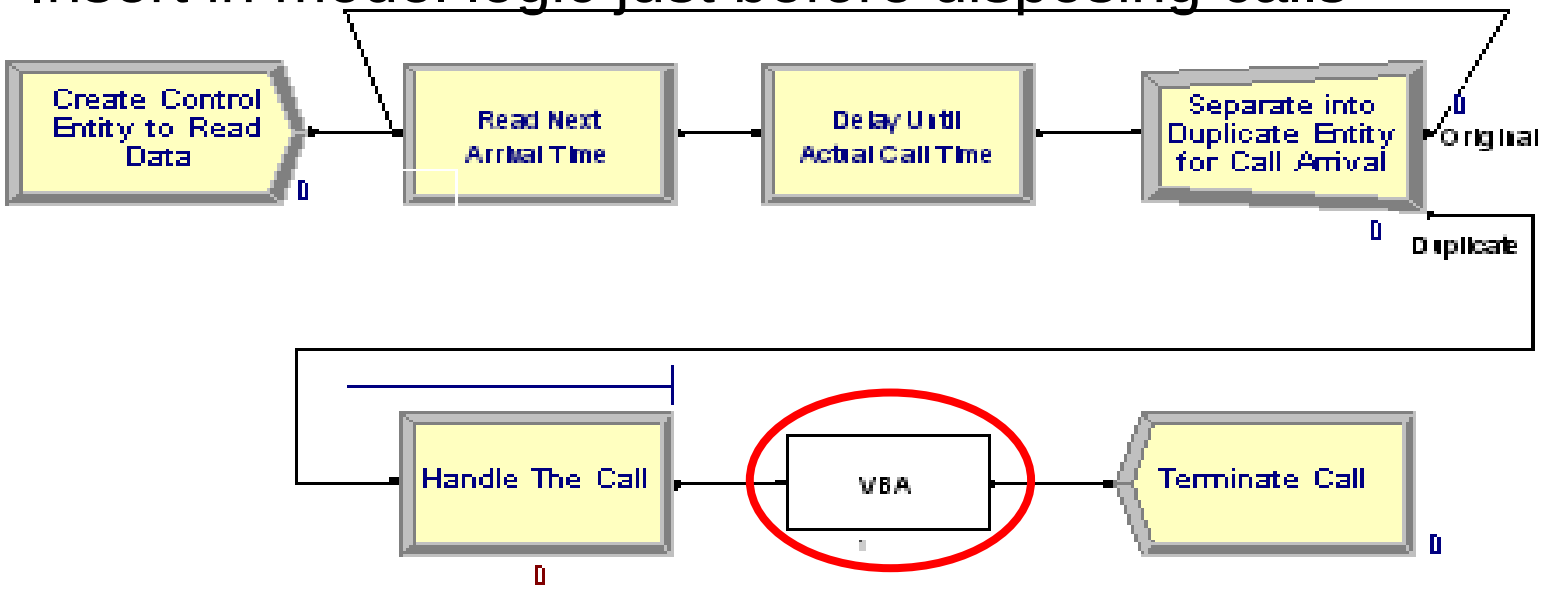
- **ThisDocument.Model.SIMAN**
  - Used to access simulation run data
  - Browse (**F2**) in VBA window for full list of variables
  - Active only when simulation run data is available -- i.e., built-in events:
    - after (and including) ModelLogic_RunBeginSimulation
    - before (and including) ModelLogic_RunEndSimulation

Our Approach

- **VBA ModelLogic_RunBeginSimulation**
  - Called ***once*** at the beginning of the simulation run
    - Start Excel with a new spreadsheet ("Workbook")
    - Format header rows for data worksheet
- **VBA ModelLogic_RunBeginReplication**
  - Called at the beginning of ***each replication***
    - Write headers for the three columns and the Day
    - Format the data columns

Our Approach (cont'd.)

- **VBA Module (Blocks panel)**
  - Insert in model logic just before disposing calls



- **VBA Code**
  - VBA modules numbered as they're placed, with corresponding VBA_Block_<n>_Fire events in VBA

Our Approach (cont'd.)

- **VBA_Block_1_Fire**
  - Called each time an entity enters the VBA Block in the model
  - Retrieve data from running simulation via SIMAN object (stored in *oSIMAN* variable)
  - Row and columns into which to write data stored in global VBA variables (*nNextRow*, *nColumnA*, *nColumnB*, *nColumnC*)

Our Approach (cont'd.)

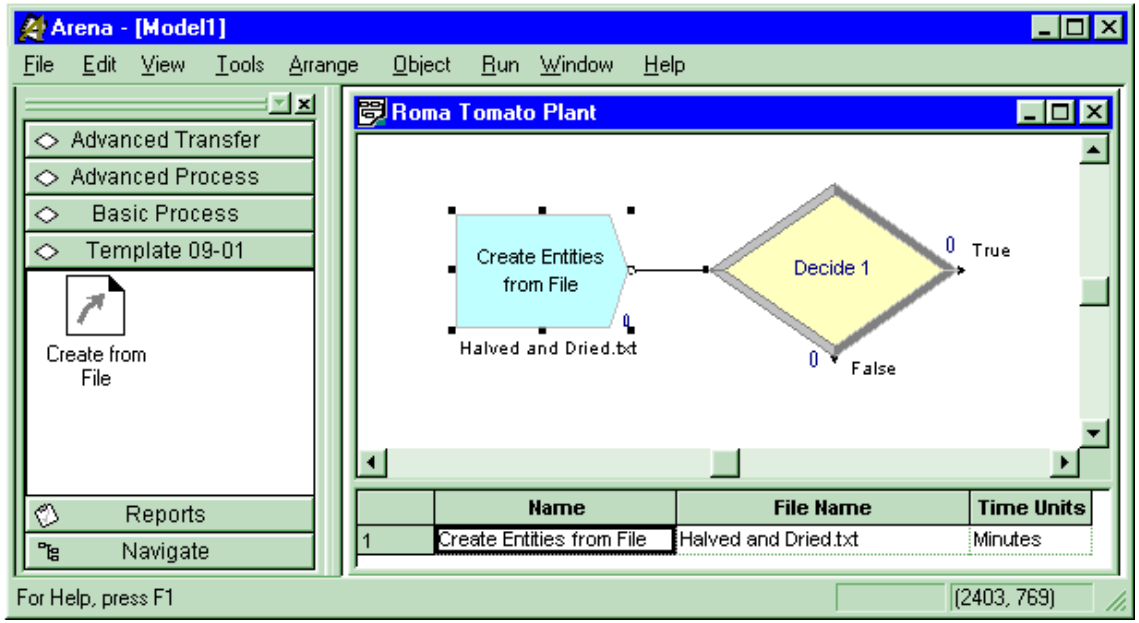
- **ModelLogic_RunEndReplication**
  - Called at end of each replication
  - Creates the chart and updates the global variables
  - Hint: Use Excel macro recording for "skeleton" code (e.g., for formatting commands, creation of chart); copy into Arena VBA and adjust variable names (e.g., add *oExcelApp* to access Excel)
- **ModelLogic_RunEndSimulation**
  - Turn *DisplayAlerts* off (overwrites *.xls* file if it exists)
  - *SaveAs* method to give filename
  - Excel still running. Could use *oExcelApp.Quit*

What We'll Do ...

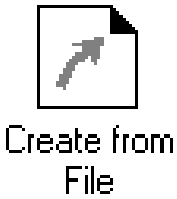
- Reading and Writing Data Files (ReadWrite)
- ActiveX™ and Visual Basic® for Applications (VBA)
- **Creating Modules with Arena Professional Edition**

Creating Modules with Arena Professional Edition

- **Template 10-01: *Create from File* module**
  - Template (*.tpo*) can be attached and used in Academic Arena
  - Place and edit like any other module
  - Need Research/ Professional Arena to create/edit template source (*.tpl*)



Panel Icon and User View

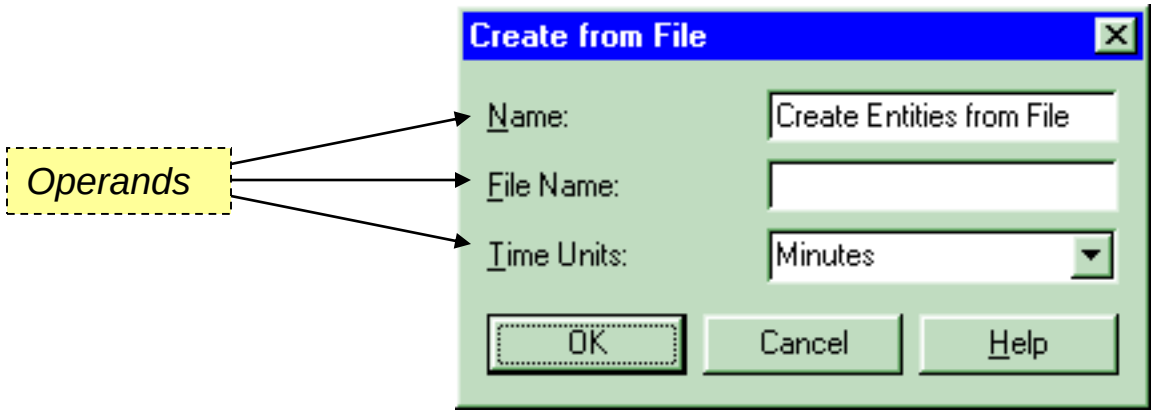


- ***Panel Icon*: "Button" displayed in template panel to represent the module**

- ***User View*: Graphics objects placed in the model window with an "instance" of the module**
  - Module handle
  - Entry, exit points
  - Animation
  - Operand values

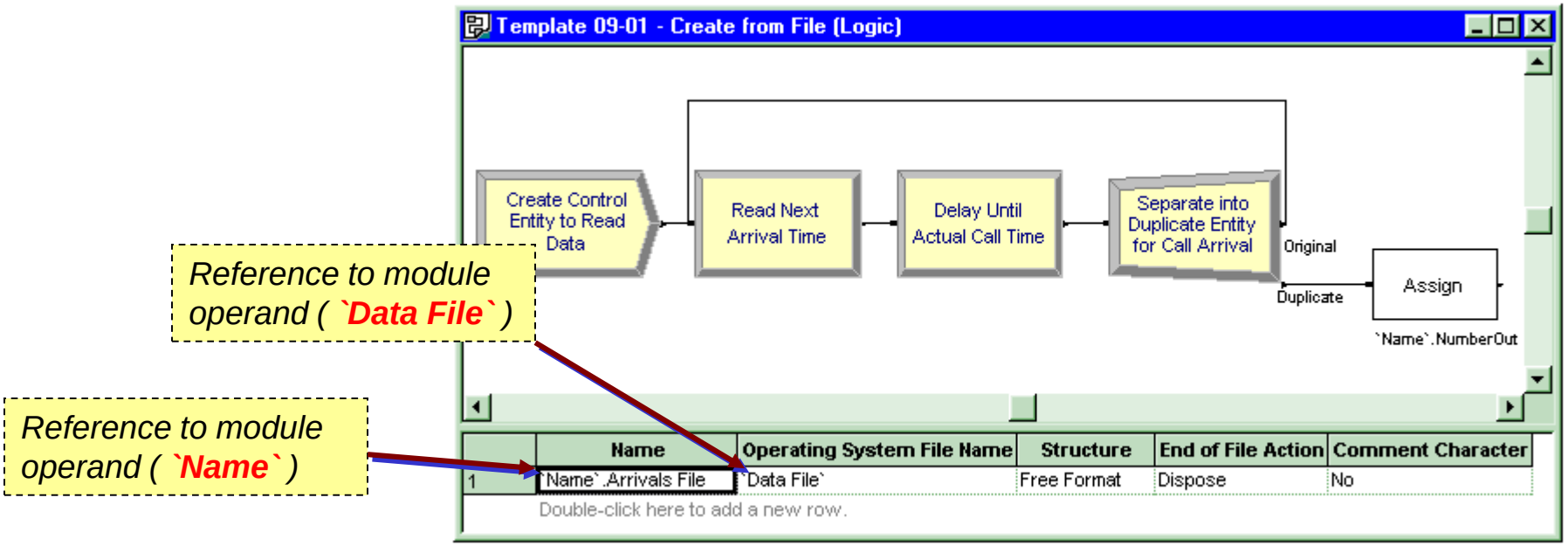
Operands

- ***Operands*: Fields that can be filled out by the modeler**
  - Allow different data for each instance of the same module type in a model
  - Passed down to logic via back quotes (`)
  - Entry and exit point operands permit entities to move through underlying module logic



Module Logic

- ***Module Logic*: A "submodel" containing module instances**
  - Can paste from a model into a module definition's logic window
  - Same interface as for building models



Uses of Templates

- **Commercial templates**
  - Arena templates (Basic Process, Advanced Process, etc.)
  - Contact Center, Packaging templates
  - …
- **Application-focused templates**
  - Terminology, modeling capabilities designed for a particular application environment (e.g., mining, material handling, order processing)
  - “Personal” / utility templates
  - Reuse what you learn
  - Share your modeling techniques

Summary

- **We have just barely scratched the surface of Arena customization and interactions with other software including:**
  - Reading and writing various types of external data files.
  - Visual Basic for Applications
  - Interacting with Microsoft Office
  - Building custom applications with templates.
- **There are many other ways of customizing Arena and allowing Arena to interact and exchange data with other software.**