ARENA Tutorial -3

ARENA Tutorial

1. Historisches
2. Basis-Elemente
3. Ergebnisanalyse
4. Modellierung von Transportvorgängen
5. Integration mit anderen Systemen
6. Customizing
7. Kontinuierliche und kombinierte Modelle

Eingabedaten

- Direkte Verwendung
  - Lesen von gespeicherten (beobachteten) Daten als Eingabedaten (Zwischenankunftszeiten, Bedienzeiten)
  - Alle Daten sind legal und relistisch
  - Dieser Wertebereich wird nicht verlassen
  - Oft sind es nicht genügend Daten für lange Simulationsläufe
- Verwendung von Verteilungsfunktionen
  - Ableitung von Verteilungsfunktionen aus empirischen Daten
  - Realisierte Werte können dann außerhalb der beobachteten Wertebereiche liegen

ARENA Input-Analyzer

- Input-Analyzer ist ein Tool zur Ableitung von Verteilungsfunktionen aus empirischen Daten
- Voraussetzungen:
  - Empirische Daten müssen Unabhängig (independent) und einer identischen Verteilung (identically distribution) entstammen
- Input-Analyzer kann als selbständiges Tool genutzt werden

WIRTSCHAFTS INFORMATIK

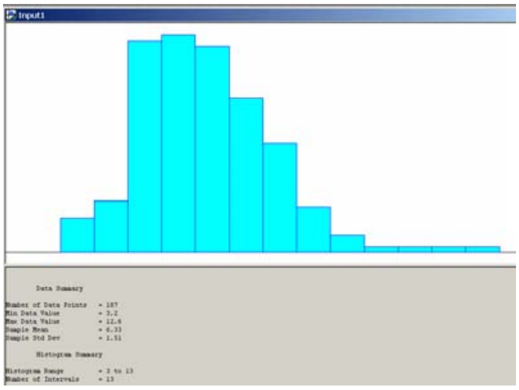
ARENA Input-Analyzer

- Sucht die „passendste“ Vertelungsfunktion und bestimmt deren Parameter
- Verwendet dafür unterschiedliche Methoden (Maximum likelihood, moment matching, least squares, …)
- Bewertung der gefundenen Verteilungen mittels Hypothesen-Tests
    - » H_0: die gefundene Funktion repräsentiert die Daten
    - » Berechnung eines *p* value zum Test (klein = schlechte Anpassung)
- Ermittlung von empirischen Verteilungen

WIRTSCHAFTS INFORMATIK

ARENA Inpit-Analyzer

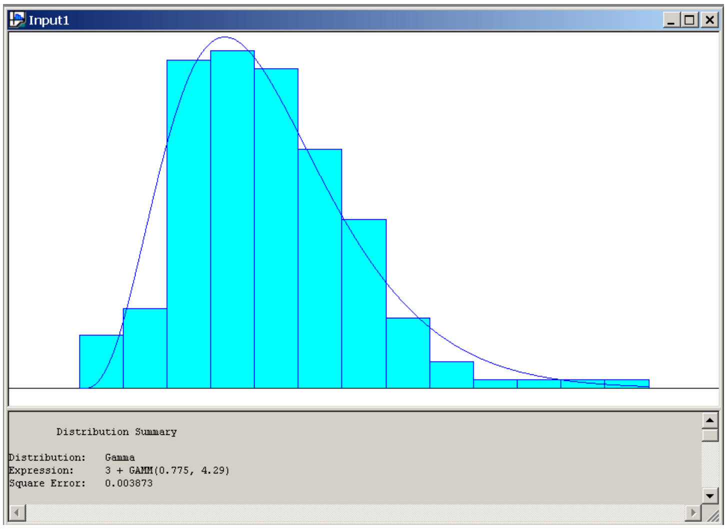
- File->New
    - – Anlegen eines „Projectes“
- File->Data File
    - – Laden einer Datei mit den empirischen Daten (partbprp.dst)



```
        Data Summary

Number of Data Points   = 187
Min Data Value          = 3.2
Max Data Value          = 12.6
Sample Mean             = 6.33
Sample Std Dev          = 1.51

        Histogram Summary

Histogram Range         = 3 to 13
Number of Intervals     = 13
```

WIRTSCHAFTSINFORMATIK

Gefundene Funktion



```
        Distribution Summary

Distribution:   Gamma
Expression:     3 + GAMM(0.775, 4.29)
Square Error:   0.003873
```

WIRTSCHAFTSINFORMATIK

Bewertung der Funktion

```
Distribution:        Gamma
Expression:          3 + GAMM(0.775, 4.29)
Square Error:        0.003873

Chi Square Test
  Number of intervals         = 7
  Degrees of freedom          = 4
  Test Statistic              = 4.68
  Corresponding p-value       = 0.337

Kolmogorov-Smirnov Test
  Test Statistic     = 0.0727
  Corresponding p-value       > 0.15

          Data Summary
Number of Data Points         = 187
Min Data Value                = 3.2
Max Data Value                = 12.6
Sample Mean                   = 6.33
Sample Std Dev                = 1.51

          Histogram Summary
Histogram Range      = 3 to 13
Number of Intervals = 13
```

Vergleich zwischen mehreren Funktionen

- Fit/Fit All
- Sortiert die Funktionen nach der Methode der kleinsten quadratischen Fehler
  - Unterschiede zwischen empirischen Häufigkeiten und den Häufigkeiten der ausgewählten Funktion
- Sensibel hinsichtlich der Anzahl der Intervalle
- Unbedingt auf den p-Value achten

Vergleich zwischen mehreren Funktionen

```
Function       Sq Error
-----------------------
Gamma          0.00387
Weibull        0.00443
Beta           0.00444
Erlang         0.00487
Normal         0.00633
Lognormal      0.00871
Triangular     0.0246
Uniform        0.0773
Exponential    0.0806
```

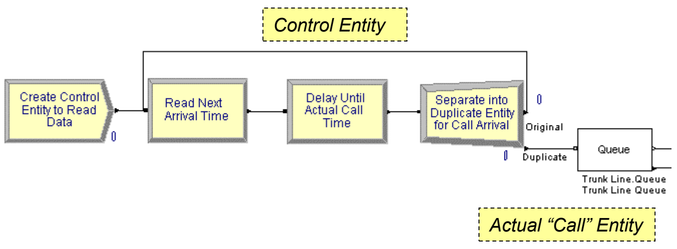
Generating Entities from Historical Data (ReadWrite)

- Trace-driven simulations
  - Model validation
  - Assumes historical data exist and can be transformed for use in simulation
- Model 09-01.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 In*** 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 In - TNOW*)
  - Create a duplicate (Separate module) to dispatch actual call into model. Original entity loops back to read next time.

Run Termination for Trace-Driven Scenario

- 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
  - Model 09-01: Resource schedules continue (time-based control process), so run terminates at replication length specified in run setup

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**

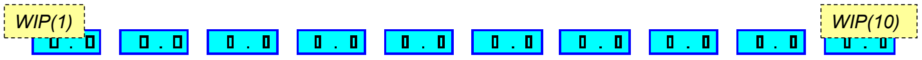
Module data available

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

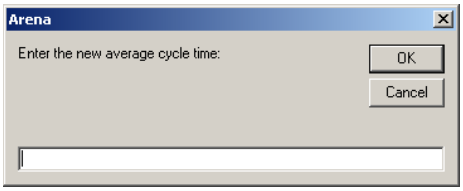
Sample: Create Ten Status Variables

```
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

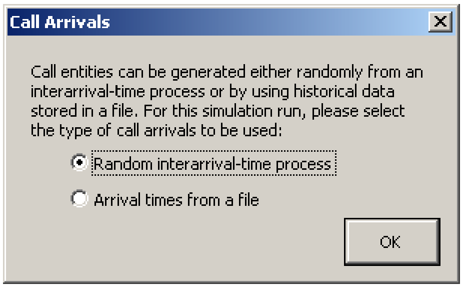
```
Dim oSIMAN As Arena.SIMAN
Dim nVarIndex As Long
Dim sNewValue As String

' 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

Enter the new average cycle time:

Model 9-2: 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



Call Arrivals

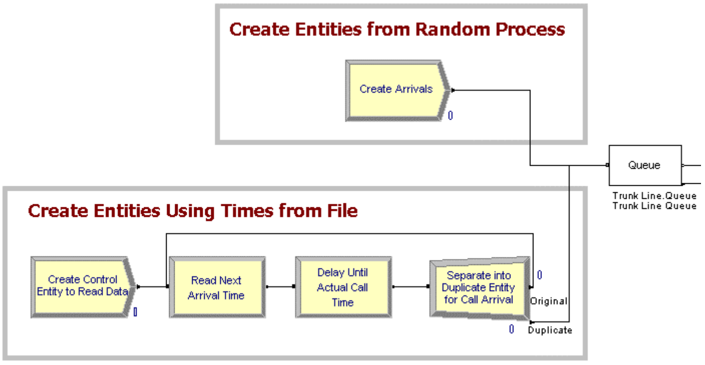
Call entities can be generated either randomly from an interarrival-time process or by using historical data stored in a file. For this simulation run, please select the type of call arrivals to be used:

[✓ Random interarrival-time process]

Arrival times from a file

Our Approach

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

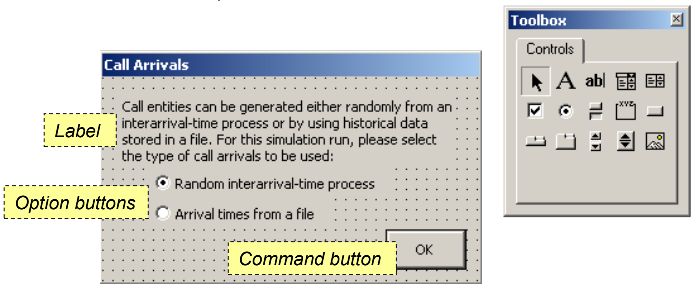

Create Entities from Random Process
Create Arrivals
0
Queue
Trunk Line.Queue
Trunk Line Queue
Create Entities Using Times from File
Create Control Entity to Read Data
0
Read Next Arrival Time
Delay Until Actual Call Time
Separate into Duplicate Entity for Call Arrival
0
Original
0
Duplicate

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 Arrivals*** module: ***Infinite***
  - ***Create Control Entity to Read Data*** module: ***0***
- Arrival times from a file
  - ***Create Arrivals*** 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)



Call entities can be generated either randomly from an interarrival-time process or by using historical data stored in a file. For this simulation run, please select the type of call arrivals to be used:

(●) Random interarrival-time process

( ) Arrival times from a file

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
```

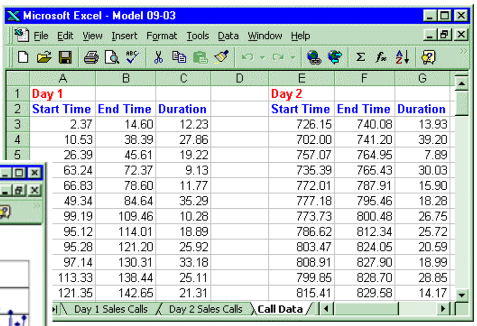
- **Program control passes to the form until it's closed**
- **Arena run "suspended" while form is in control**

Change Module Data On OK

- When user clicks OK button on form, modify the Create module data
  - Open the Create and Direct Arrivals submodel to gain access to the Create modules
  - Set the Max Arrivals fields
  - Display the top-level model's animation view
  - Play a sound
  - Close the UserForm
- When form is closed, simulation run commences with the new data values in the Create modules

Model 9-3: Record Call Data in Microsoft Excel

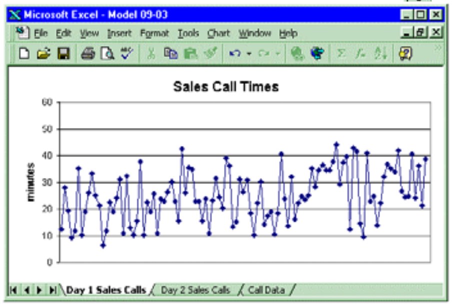
- Our goal:
  - Raw call data tables
  - Daily call duration charts



Microsoft Excel - Model 09-03

	A	B	C	D	E	F	G
1	Day 1				Day 2		
2	Start Time	End Time	Duration		Start Time	End Time	Duration
3	2.37	14.60	12.23		726.15	740.08	13.93
4	10.53	38.39	27.86		702.00	741.20	39.20
5	26.39	45.61	19.22		757.07	764.95	7.89
	63.24	72.37	9.13		735.39	765.43	30.03
	66.83	78.60	11.77		772.01	787.91	15.90
	49.34	84.64	35.29		777.18	795.46	18.28
	99.19	109.46	10.28		773.73	800.48	26.75
	95.12	114.01	18.89		786.62	812.34	25.72
	95.28	121.20	25.92		803.47	824.05	20.59
	97.14	130.31	33.18		808.91	827.90	18.99
	113.33	138.44	25.11		799.85	828.70	28.85
	121.35	142.65	21.31		815.41	829.58	14.17

Day 1 Sales Calls / Day 2 Sales Calls / Call Data



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

WIRTSCHAFTSINFORMATIK

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

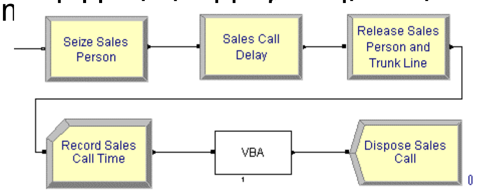
WIRTSCHAFTSINFORMATIK

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 n [illegible] alls



- **VBA Code**
  - ***VBA_Block_1_Fire*** event called each time an entity enters the VBA block in the model logic
  - 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*)

WIRTSCHAFTSINFORMATIK

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*