Automated mapping of distribution routes using Python

Mathieu Klapwijk
FASTENAL

Esri UC 2015
Agenda

- Fastenal, distribution routes
- Process and scripts
- Benefits
Fastenal

- Industrial and construction supplies
- HQ in Winona, MN
- Worldwide operation
- 2,700 stores, primarily in North America
- 14 distribution centers ("hubs")
- 10 manufacturing facilities
Fastenal - Distribution Routes

- USA: 2,300+ stores and 11 hubs
- Canada: 200+ stores and 2 hubs
- Semi’s, straight trucks, sprinters
- Hub-to-store delivery routes
- Routes start at hub, store or terminal
- 400+ routes; each 1 or more times per week
- Total 1,400+ route instances per week
## Fastenal - Distribution Routes

### Route configuration

<table>
<thead>
<tr>
<th>DOW</th>
<th>SUN</th>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>543</td>
<td>15</td>
<td>543</td>
<td>15</td>
<td>543</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPR</td>
<td>SEM</td>
<td>SPR</td>
<td>SEM</td>
<td>SPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23:30 SUN</td>
<td>23:30 MON</td>
<td>23:30 TUE</td>
<td>23:30 WED</td>
<td>23:30 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MN100</td>
<td>MN100</td>
<td>MN100</td>
<td>MN100</td>
<td>MN100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>IAMAS</td>
<td>IAAME</td>
<td>IAMAS</td>
<td>IAAME</td>
<td>IAMAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IAIOA &amp; IACAR</td>
<td>IADE1</td>
<td>IADE1</td>
<td>IADE1</td>
<td>IADE1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IAAME</td>
<td>IADE1</td>
<td>IAGRM</td>
<td>IADES</td>
<td>IAGRM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>IADE1</td>
<td>IADES</td>
<td>IAFTD &amp; IAPEA</td>
<td>IAFTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IADES &amp; IAPEA</td>
<td>IAHUM &amp; IA005</td>
<td>IAHUM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IAGRM &amp; IA005</td>
<td>IA006</td>
<td>IAGRM</td>
<td>IA006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>IAFTD</td>
<td>IAGRM</td>
<td>IA003</td>
<td>IAFTD</td>
<td>IA003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>IAHUM</td>
<td>IAFTD</td>
<td>MN100</td>
<td>IAHUM</td>
<td>MN100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>IA006</td>
<td>IAHUM</td>
<td></td>
<td>MN100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>IA003</td>
<td>MN100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>MN100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fastenal – Distribution Routes

Related tables

**ROUTE_MAIN_SCHEDULED**
- ROUTE_MAIN_ID
- ROUTE_MAIN_NUMBER
- DESCRIPTION
- STATUS
- DATE_CREATED
- VERSION
- VERSION_DATE
- HUB

**ROUTE_SCHEDULED**
- ROUTE_SCHEDULED_ID
- ROUTE_NUMBER
- ROUTE_MAIN_ID
- ROUTE_PARENT_ID
- TRUCK
- DAY
- TEAM
- BUNK
- ROUTE_START_TIME
- ROUTE_START_DAY
- DOW_WMS

**ROUTE_STOP_SCHEDULED**
- ROUTE_STOP_SCHEDULED_ID
- ROUTE_SCHEDULED_ID
- COMPANY_STOP
- DROPPED
- VENDOR_STOP
- STOP_TYPE
- STOP_SEQUENCE_NUMBER
- START_ROUTE
- END_ROUTE
- LEG_TIME
- LEG_DISTANCE
- STOP_TIME
- ARRIVAL_TIME_SCHEDULED
- ARR_DAY_SCHED
- DEPARTURE_TIME_SCHEDULED
- DEP_DAY_SCHED
- MILES_TYPE
- Latitude
- Longitude
- LegTime_VB
- StopTime_VB
Fastenal – Distribution Routes

Routes wall map
Process - Overview

- Per Hub
- Two steps:
  1. Data Preparation
  2. Main Python Script

- Required: Network Analyst Extension
Process - 1. Data Preparation

• Create folder structure

• Create spreadsheet [Hub]_Routes
  tab RoutelIDs (route ID, route nr, day, truck type)
  tab RouteStops (route ID, stop order, lat/long, ...)

• Copy values for RoutelIDs and RouteStops from Routes database to spreadsheet
Process - 1. Data Preparation (ArcMap)

• New map document
• Create folder structure
  Hub_Routes
  AHUB_Routes
  DHUB_Routes
  HHUB_Routes
  IHUB_Routes
  ...
  Hub_Symbology
  Hub_RouteStops_shp

• Add symbology template shapefiles
• Drag tabs RouteIDs and RouteStops from Catalog to map
Process - 1. Data Preparation (Python window)

- ExcelToDBF.py
- RemoveExcel_RouteStops_RoutelDs.py
- MakeXYEventLayer.py
- XYEventLayerToShapefile.py

Example:

```python
SpatialReference = "D:/ESRI/ArcGIS102DataMaps/usa/background/sm_cntry.sdc.prj"

hubs = ['AHUB', 'DHUB', 'HHUB', 'IHUB', 'KHUB', 'LHUB', 'MHUB', 'NHUB', 'OHUB', 'SHUB', 'UHUB', 'VHUB', 'WINO']

for hub in hubs:
    print hub

RouteStopsTable = (hub) + "_RouteStops"
EventsLayerName = (hub) + "_RouteStops_Events"

arcpy.MakeXYEventLayer_management(RouteStopsTable,"Longitude","Latitude",EventsLayerName, SpatialReference)
```
Process - 1. Data Preparation (Result)

For each hub:

• Table RouteIDs
• Shapefile RouteStops (Stores)
• Shapefile Symbology Template
Process – 2. Main Python Script (Overview)

• Setup environment, parameters
• Create a cursor for table RouteIDs
• For each RouteID:
  Select RouteStops
  Make route layer
  Add stops to route layer
  Solve route
  Save route as shapefile
  Add route shapefile to map
  Apply symbology
import arcpy
from arcpy import env

# Check out Network Analyst extension
arcpy.CheckOutExtension("Network")

# Define parameters
# parameter(0) = Hub 4-digit code (datatype = String)
# parameter(1) = Project Folder (to store Hub Folders) (datatype = Folder)
# parameter(2) = Routes/Scops Layer (datatype = Feature Layer)
# parameter(3) = shapefile, not XY Event
# parameter(4) = Routes/Scops Layer (datatype = Feature Layer)
# parameter(5) = Network Dataset (datatype = Feature Layer)
# parameter(6) = Network Analyst Output Layer
# parameter(7) = Apply same Symbology to all Routes (datatype = Boolean)

Hub = arcpy.GetParameter(0)
ProjectFolder = arcpy.GetParameter(1)
RoutesScopsLayer = arcpy.GetParameter(2)
RoutesScopsTable = arcpy.GetParameter(3)
NetworkDataset = arcpy.GetParameter(4)
inNetworkDataset = arcpy.GetParameter(5)
ApplySymbologyFlag = arcpy.GetParameter(7)
Process – 2. Main Python Script

- Create a cursor for table RouteIDs

```python
# Create SearchCursor to loop through RouteIDs
with arcpy.da.SearchCursor(RouteIDsTable, ["RouteID", "MainNr", "RouteNr", "Day", "Vehicle"])
```

```python
for row in cursor:
    # Set attributes for Select statement, 
    # convert RouteID and RouteNr to string in WhereClause
    RouteID = row[0]
    MainNr = row[1]
    RouteNr = row[2]
    Day = row[3]
    Vehicle = row[4]
```

- “Do something” with each row
Process – 2. Main Python Script

- For each RouteID:
  - Select RouteStops (Stores)
  - Make route layer
  - Add stops to route layer
  - Solve route
  - Save route as shapefile
  - Add route shapefile to map
  - Apply symbology
Process – 2. Main Python Script

- For each RouteID:
  Select RouteStops

```python
# Create SearchCursor to loop through RouteIDs
with arcpy.da.SearchCursor(RouteIDsTable, ["RouteID", "MainNr", "RouteNr", 
                                          "Day", "Vehicle"]) as cursor:
    for row in cursor:
        # Set attributes for Select statement,
        # convert RouteID and RouteNr to string in WhereClause
        RouteID = row[0]
        MainNr = row[1]
        RouteNr = row[2]
        Day = row[3]
        Vehicle = row[4]

        # Exclude freigted routes, only run through code if Vehicle <> FRT
        if Vehicle <> "FRT":

            # Select RouteStops from Route Stops Layer
            WhereClause = (" " + str(int(RouteID)) + 
                                 " AND " + "Drop" + " <> -1 "
                                 
            arcpy.SelectLayerByAttribute_management(RouteStopsLayer, 
                          "NEW_SELECTION", 
                          WhereClause)
```
Process – 2. Main Python Script

- For each RouteID:
  - Select RouteStops
  - **Make route layer**
  - Add stops to route layer
  - Solve route
  - Save route as shapefile
  - Add route shapefile to map
  - Apply symbology

```python
# Zoom to selected RouteStops
#DataFrame.zoomToSelectedFeatures()
#arcpy.RefreshActiveView()

# Make Route Layer - assign output layer (outNALayer) to parameter
outNALayer = str(int(RouteID)) + "-" + str(RouteNr) + "_" + (Day)
arcpy.AddMessage(outNALayer)
impedanceAttribute = "Time"
arcpy.MakeRouteLayer_na(inNetworkDataset, outNALayer, impedanceAttribute)
arcpy.SetParameterAsText(6, outNALayer)
```
Process – 2. Main Python Script

- For each RoutelID:
  Select RouteStops
  Make route layer
Process – 2. Main Python Script

- For each RouteID:
  Select RouteStops
  Make route layer
  **Add stops to route layer**
  Solve route
  Save route as shapefile
  Add route shapefile to map
  Apply symbology

```python
106   # Add selected RouteStops to Route
107   # The AddLocations_na tool automatically adds SELECTED RECORDS ONLY from
108   # the in table (RouteStopsLayer)
109   arcpy.AddLocations_na(outNALayer,"Stops",RouteStopsLayer)
```
Process – 2. Main Python Script

- For each RouteID:
  Select RouteStops
  Make route layer
  Add stops to route layer
  **Solve route**
  Save route as shapefile
  Add route shapefile to map
  Apply symbology

```python
# Solve Route: arcpy.Solve_na(outNALayer, "SKIP")
# (option "SKIP" to ignore invalid locations)
arcpy.Solve_na(outNALayer,"SKIP")
```
Process – 2. Main Python Script

- For each RouteID:
  Select RouteStops
  Make route layer
  Add stops to route layer
  Solve route
  **Save route as shapefile**
  Add route shapefile to map
  Apply symbology

```python
# Set RoutesLayer Name from outNALayer + "\Routes"
RoutesLayerName = str(outNALayer) + "\Routes"

# Create layer variable RoutesLayer as 'In Features' for CopyFeatures_management
RoutesLayer = arcpy.mapping.Layer(RoutesLayerName)

# Create 'Out Feature Class' as output for CopyFeatures_management
RoutesLayerCopy = str(workspace) + "\"\" + (Hub) + "_" + \
                 str(RouteNr) + "_" + (Day)

# Copy Routes sublayer to Hub folder
arcpy.CopyFeatures_management(RoutesLayer, RoutesLayerCopy)

# Delete Output Layer, RoutesLayer from ArcMap's in-memory workspace to prevent
# consumption of all physical memory (RAM)
arcpy.Delete_management(outNALayer)
arcpy.Delete_management(RoutesLayer)
```
Process – 2. Main Python Script

- For each RouteID:
  - Select RouteStops
  - Make route layer
  - Add stops to route layer
  - Solve route
  - Save route as shapefile
  - Add route shapefile to map
  - Apply symbology

```python
# Add Routes Shapefile to Map Document
# Set Routes Shapefile Name including workspace (Hub Folder)
RoutesShapefileName = str(workspace) + "\" + (Hub) + "_" + \
    str(RouteNr) + "_" + (Day) + ".shp"

# Create Feature Layer from Routes Shapefile
RoutesFeatureLayer = arcpy.mapping.Layer(RoutesShapefileName)

# Add Routes Feature Layer to HubRoutesGroupLayer in Map Document
arcpy.mapping.AddLayerToGroup(DataFrame, HubRoutesGroupLayer, \
    RoutesFeatureLayer)
```
Process – 2. Main Python Script

• For each RouteID:
  Select RouteStops
  Make route layer
  Add stops to route layer
  Solve route
  Save route as shapefile
  Add route shapefile to map
  Apply symbology

```python
# Apply Symbology from Symbology Layer to all routes if flag is set to True
if ApplySymbologyFlag == True:
    # Get RouteLayer from GroupLayer to apply symbology to
    # (cannot apply symbology to RoutesFeatureLayer that is still in memory)
    RouteLayerName = Hub + "_" + RouteNr + "_" + Day
    RouteLayer = arcpy.mapping.Layer(RouteLayerName)

    # Apply Symbology from Symbology Layer to Routes Shapefile
    arcpy.ApplySymbologyFromLayer_management(RouteLayer, SymbologyLayer)

elif RouteNr == PreviousRouteNr:
    # If current record has the same RouteNr as the previous record, _
    # apply symbology from Previous routeNr so all records with the_
    # same RouteNr get the same color
    RouteLayerName = Hub + "_" + str(RouteNr) + "_" + Day
    RouteLayer = arcpy.mapping.Layer(RouteLayerName)
    arcpy.ApplySymbologyFromLayer_management(RouteLayer, PreviousRouteLayer)
```
Process – 2. Main Python Script

- For each RouteID:
  Select RouteStops
  Make route layer
  Add stops to route layer
  Solve route
  Save route as shapefile
  Add route shapefile to map
  Apply symbology

*Set values for next iteration*

```python
# Set current RouteNr and RouteLayer as previous for use in next record
PreviousRouteNr = RouteNr
PreviousRouteLayerName = Hub + "_" + str(RouteNr) + "_" + Day
PreviousRouteLayer = arcpy.mapping.Layer(PreviousRouteLayerName)

# Make PreviousRouteLayer NOT visible to save redraw time
PreviousRouteLayer.visible = False

else:
    message = str(int(RouteID)) + "-" + str(RouteNr) + "_" + (Day) + " is Freighted"
    arcpy.AddMessage(message)
```
Process – 2. Main Python Script

- Clean up

```python
# Make HubRoutesGroupLayer NOT visible to save redraw time
HubRoutesGroupLayer.visible = False

# Refresh Active View (to apply making the HubRoutesGroupLayer not visible)
arcpy.RefreshActiveView()

# Clear selection
arcpy.SelectLayerByAttribute_management(RouteStopsLayer,"CLEAR_SELECTION")

# Clear parameters and variables
del Hub, ProjectFolder, RouteStopsLayer, RouteIDsTable, SymbologyLayer,
    inNetworkDataset
del row, cursor
del HubFolderName, workspace, DataFrame, HubRoutesGroupLayer, RouteID,
    MainNr, RouteNr, Day, Vehicle, WhereClause, outNALayer, impedanceAttribute,
    RoutesLayerName, RoutesLayer, RoutesLayerCopy, RoutesShapefileName, RoutesFeatureLayer

MapDoc.save()
del MapDoc
```
Process – 2. Main Python Script

- Completed
Process – 2. Main Python Script

Result
Process – 2. Main Python Script

Batch mode
Benefits

- Reduction of mapping time: *from 10 days to less than 8 hours*
- More frequent and timely map updates
- More versatile ~ any selection of routes
- Basis for further analysis
Questions