Traditional Cartography
What you really want
Cartographic Workflow
Esri Production Mapping
A collection of ArcMap Extensions
Map Automation

Production Mapping
Cartographic Rules
Map Automation

- Production Mapping
- Cartographic Rules
- Geoprocessing & Python
- Standard Output
Civilian Topographic Model (CTM)

- Based on the National System for Geospatial Intelligence Feature Data dictionary (NFDD)
- Includes:
  - Database schema
  - Editing Rules
  - Quality Control Rules
  - Cartographic templates for 25K
  - Cartographic templates for 50K
  - 50K Generalization Models
  - Workflow Manager Workflows
  - Distributed Generalization Workflows

https://github.com/esri/ctm
Cartographic Data
Cartographic Data

- Data and text used to enrich cartographic products
- Create data appropriate for your cartographic scale
Grids and Graticules

Design

- Feature based
- Fine Grain Custom Design
- Geographically aware
  - Scale
  - Coordinate system
  - Rotation
Grids and Graticules

Applying

# Define grid object
grid = arcpyproduction.mapping.Grid(grid_xml)

# Uses the appropriate XML for to create the grid
arcpy.AddMessage("Creating the Grid...")
output_layer = map_name + "." + grid.type
grid_result = arcpy.MakeGridsAndGraticulesLayer_cartography(grid_xml, aoi, gfds, output_layer, map_name)
grid_layer = grid_result.getOutput(0)

# Add the grid layer to the top of the map
arcpy.mapping.AddLayer(data_frame, grid_layer, "TOP")
arcpy.AddMessage("Grid Layer added to map...")

# Updates the data frame properties base on the grid
final_mxd.activeView = 'PAGE_LAYOUT'
grid.updateDataFrameProperties(data_frame, aoi)
Generalization

- Resolves the appearance of feature geometry at smaller scales
Feature Generalization

- Features assessed individually without regard to symbology or spatial relationships

1: 25000

1: 50000
Contextual Generalization

- Features are assessed collectively
  - Maintain pattern, density, and spatial relationships
Contextual Generalization

- Features from multiple layers assessed simultaneously
  - Maintain pattern, density, and spatial relationships
Find the tool and add to a model
Run the models
Demo:
Cartographic Generalization
Joe Sheffield
Symbology
Symbology
Intuitively displaying information
Feature Class Representations

- GIS Data
- Cartographic Edits

Feature Class Representations
Visual Specifications

- Create Symbology
- Know Your Rules
- Define Specifications
Visual Specifications
Applying
Cartographic Edits
Feature Class Representations
Conflict Resolution
Managing placement of symbolized features
Cartographic Refinement
Masking
Masking
Masking

```
# Masking the grid ladder values and annotations
arcpy.AddMessage("getting output of masks.")
mask_layer = arcpy.mapping.Layer(masks.getOutput(0))
arcpy.mapping.AddLayer(data_frame, anno_mask_layer, 'BOTTOM')
anno_mask = arcpy.mapping.ListLayers(final_mxd, mask_layer.name,
data_frame)[0]
arcpy.AddMessage("Annotation Mask layer added to the map...")
arcpyproduction.mapping.EnableLayerMasking(data_frame, 'true')
arcpyproduction.mapping.MaskLayer(data_frame, 'APPEND', anno_mask,gridline_layer)
arcpy.AddMessage("Masking applied to gridlines...")
```
Layout
Providing Context
Layout Elements
Types of Elements

- Static Elements
Layout Elements
Types of Elements

- Static Elements
- Dynamic Elements
  - Dynamic Text
  - Dynamic Graphics
- Template Page
Layout Elements

Automating

- Static Elements
- Dynamic Elements
  - Dynamic Text
  - Python
  - Graphic Table Element
- Page Size

```python
# Gets the list of layout elements
layout_elements = arcpy.mapping.ListLayoutElements(final_mxd)
for element in layout_element_list:
    # Update State Name text element
    if element.name == "State Name":
        element.text = state_name.upper()
```

```python
# Define variables
mxd = arcpy.mapping.MapDocument(r'C:\Project\Project.mxd')
# Set the page size
arcpy.mapping.mapping.SetPageSize(mxd, "CUSTOM",
                                   custom_width=60, custom_height=40, page_units="CENTIMETERS")
```
Layout Rules

Apply

mxd = arcpy.mapping.MapDocument("CURRENT")
arcpyproduction.mapping.ApplyLayoutRules(mxd, layout_rules.xml)
Output Product
Product/Map
Sharing with community, the way they need it
Production PDF
Color Separation and Transparency
Automating Output
Using Python

```python
if export == 'JPEG':
    filename = map_doc_name + '.jpg'
    outfile = os.path.join(outputdirectory, filename)

    # Run the export tool
    arcpy.mapping.ExportToJPEG(mxd, outfile)

elif export == "MAP PACKAGE":
    filename = map_doc_name + '.mpk'
    outfile = os.path.join(outputdirectory, filename)
    mxd = mxd.filePath

    # Run the export tool
    arcpy.PackageMap_management(mxd, outfile)

elif export == 'PRODUCTION PDF':
    filename = ap_doc_name + '.pdf'
    outfile = os.path.join(outputdirectory, filename)
    setting_file = os.path.join(product_location, "colormap.xml")

    arcpyproduction.mapping.ExportToProductionPDF(mxd, outfile, setting_file)

arcpy.AddMessage("Output is located: " + outfile)
```
Demo: Layout and Output
Joe Sheffield
Production Mapping for ArcGIS Server
Enabling self-service mapping for Authoritative products
Production Mapping for ArcGIS Server

- Database
- Production Mapping Rules & Templates
- GP Services & Python Scripts
- ArcGIS
- Web
- Mobile
- Public Access
Product on Demand

https://github.com/Esri/product-on-demand
Demo: POD
Joe Sheffield
Conclusions
Map Automation & Advanced Cartography

- Output
- Cartographic Data
- Layout
- Symbology
- Cartographic Edits
Map Automation

Production Mapping
Cartographic Rules

Geoprocessing
& Python

Standard Output
Production Mapping

Learn More:
Desktop: http://www.esri.com/productionmapping

Email us: productionmapping@esri.com

CTM
Get it:
https://github.com/esri/ctm

Product on Demand
Get it:
https://github.com/Esri/product-on-demand
Try it:
http://pod.arcgis.com/
Please Take Our Survey on the Esri Events App!

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the survey

Complete Answers and Select “Submit”