



Configure-to-Order on the Web with Autodesk Inventor Engineer-to-Order

Sanjay Ramaswamy, Product Manager
Jon Balgley, User Experience Designer

Class Summary

Learn how ETO enables the creation of web applications to help with bid and order processes of configurable products.

Learning Objectives

At the end of this class, you will be able to:

- Understand how Inventor ETO works at a high level
 - Describe how ETO drives Inventor models with rules
 - Describe how ETO uses/reuses CAD files and interacts with Vault
- Understand the capabilities of Inventor ETO Server
 - Identify 3 modes of accessing Inventor ETO Server for web applications
 - Understand the applicability of each mode for your workflows

Design Landscape

- Engineered to Order

- Configured to Order

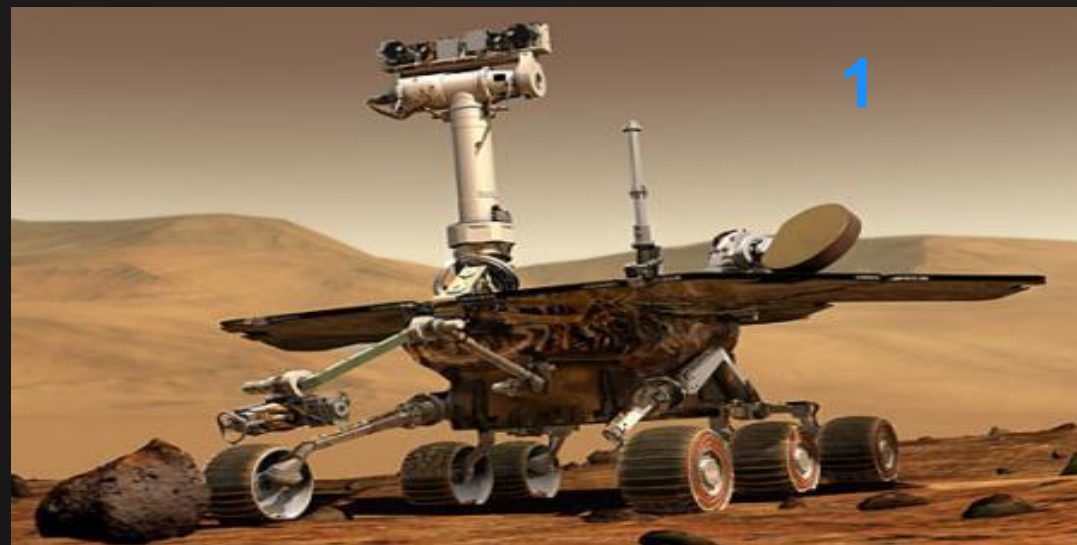
0%

25%

50%

75%

100%



100%

75%

50%

25%

0%



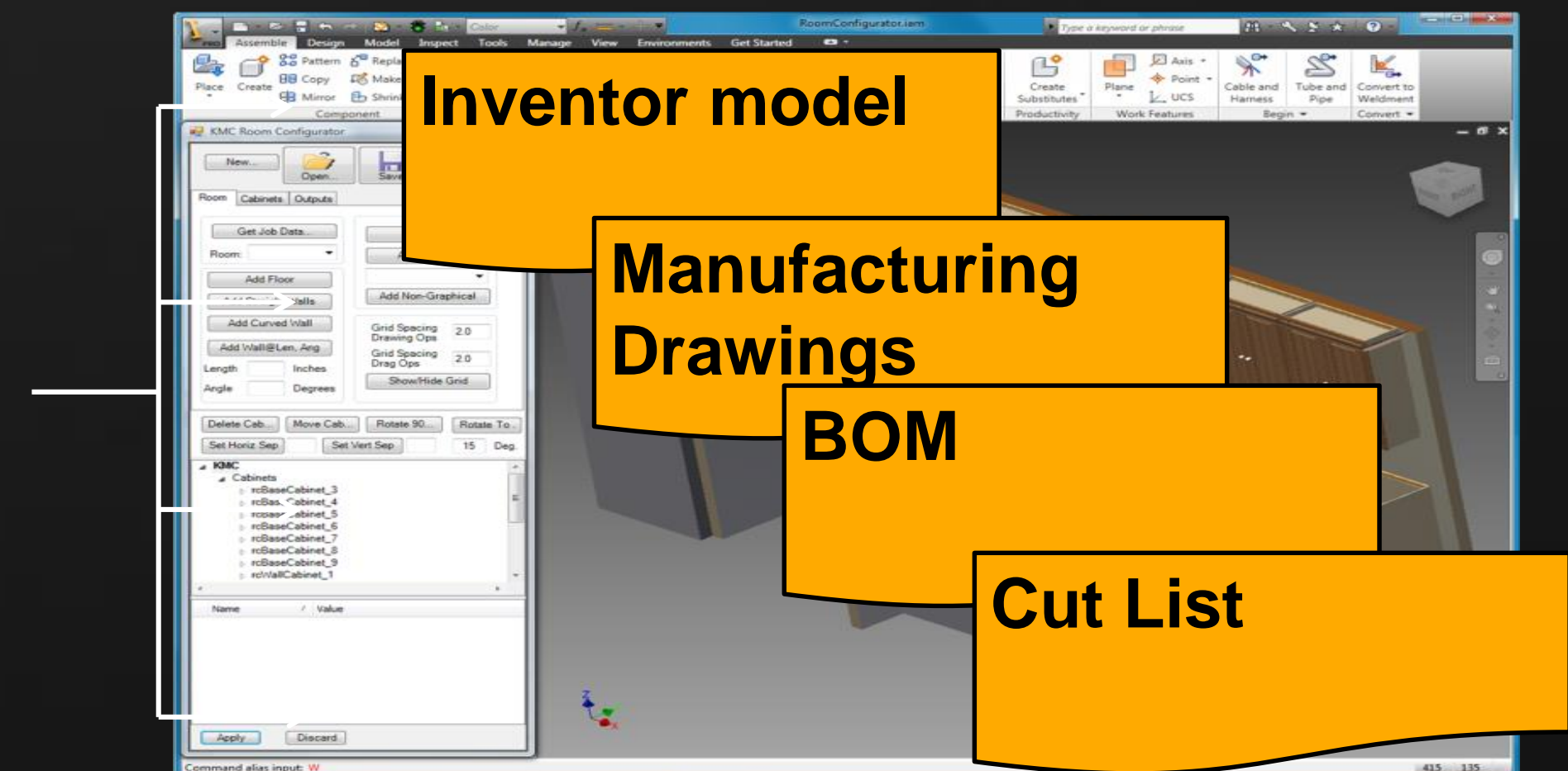
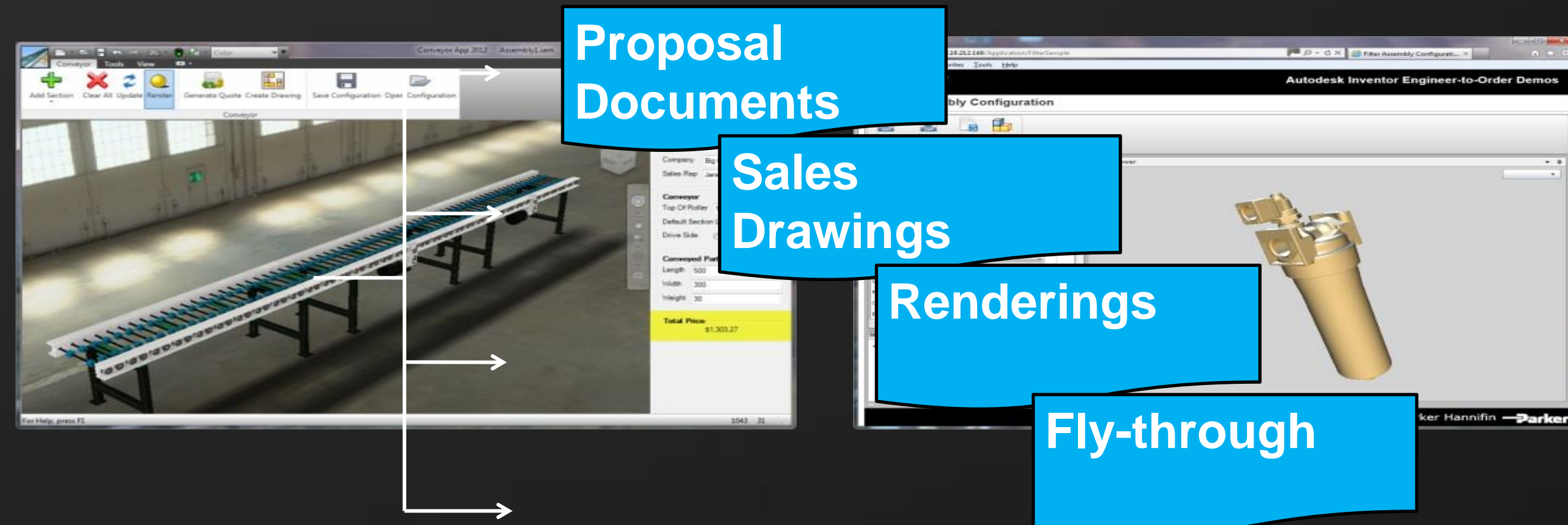
Solution Overview

Sales Automation

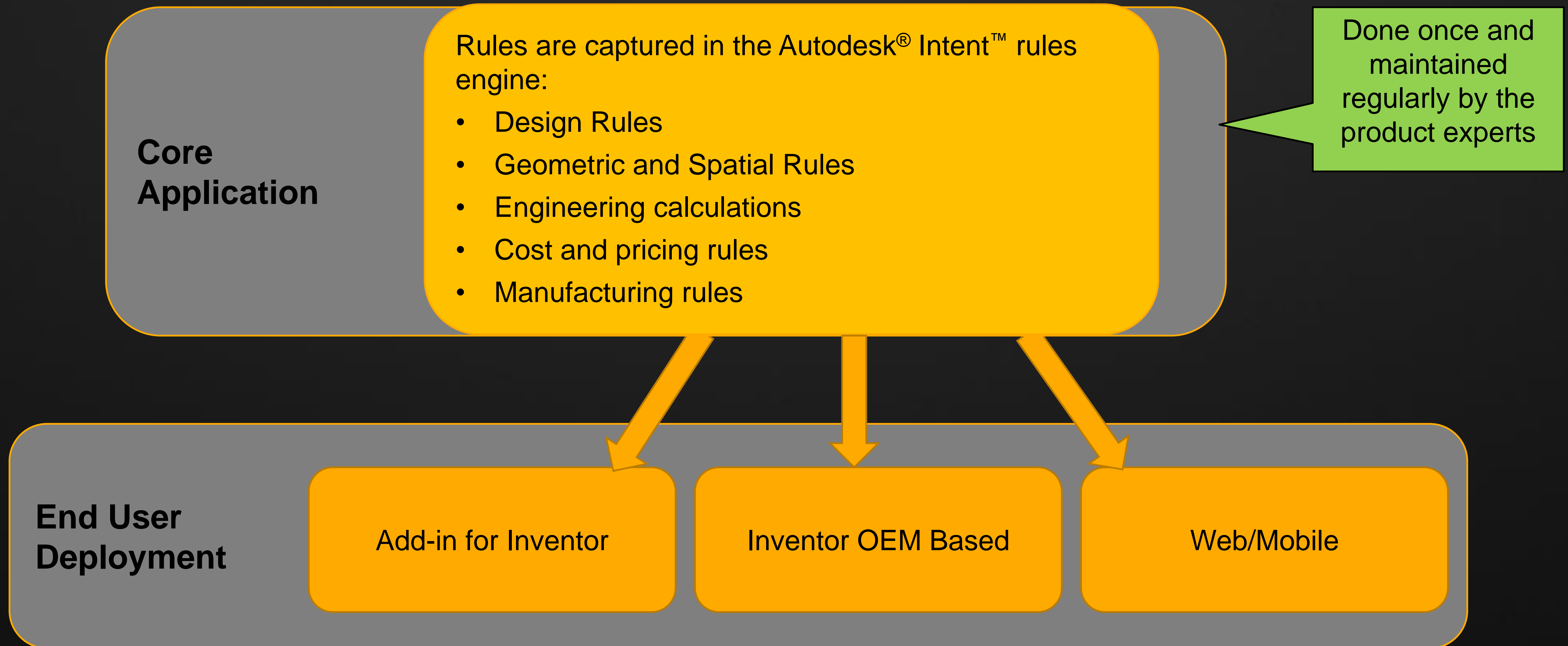
Bring engineering to the point of sale

Engineering Automation

Automate order engineering tasks



Solution Architecture



How Inventor ETO Works

How Inventor ETO Works

- It's a little like programming, ...
 - Language, compiler, IDE, run-time, etc.
 - This is "Intent"
- But different ...
 - Declarative (like Excel)
 - Tied in tightly to Inventor
- So it is easy to create highly parameterized models and drawings

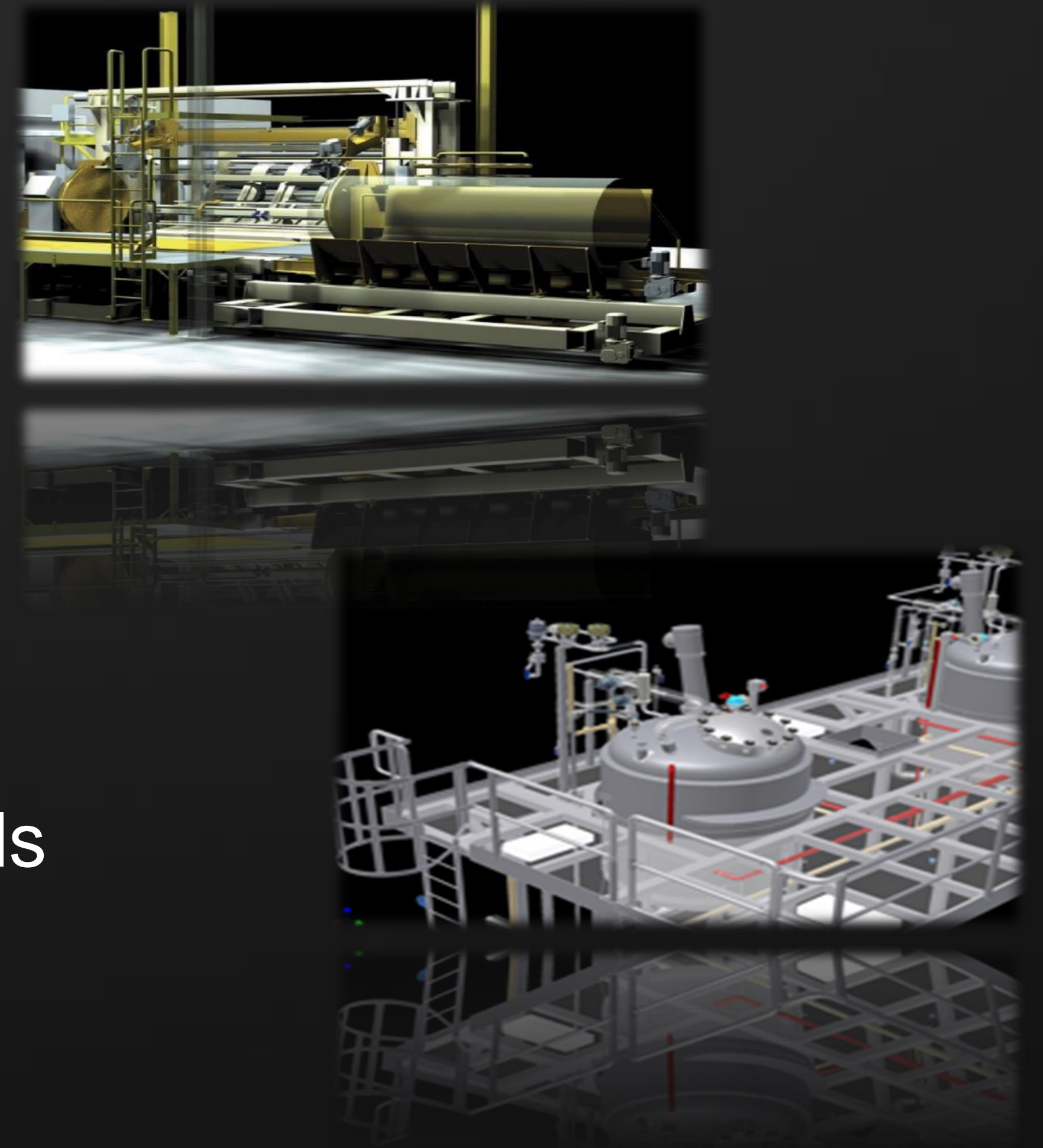
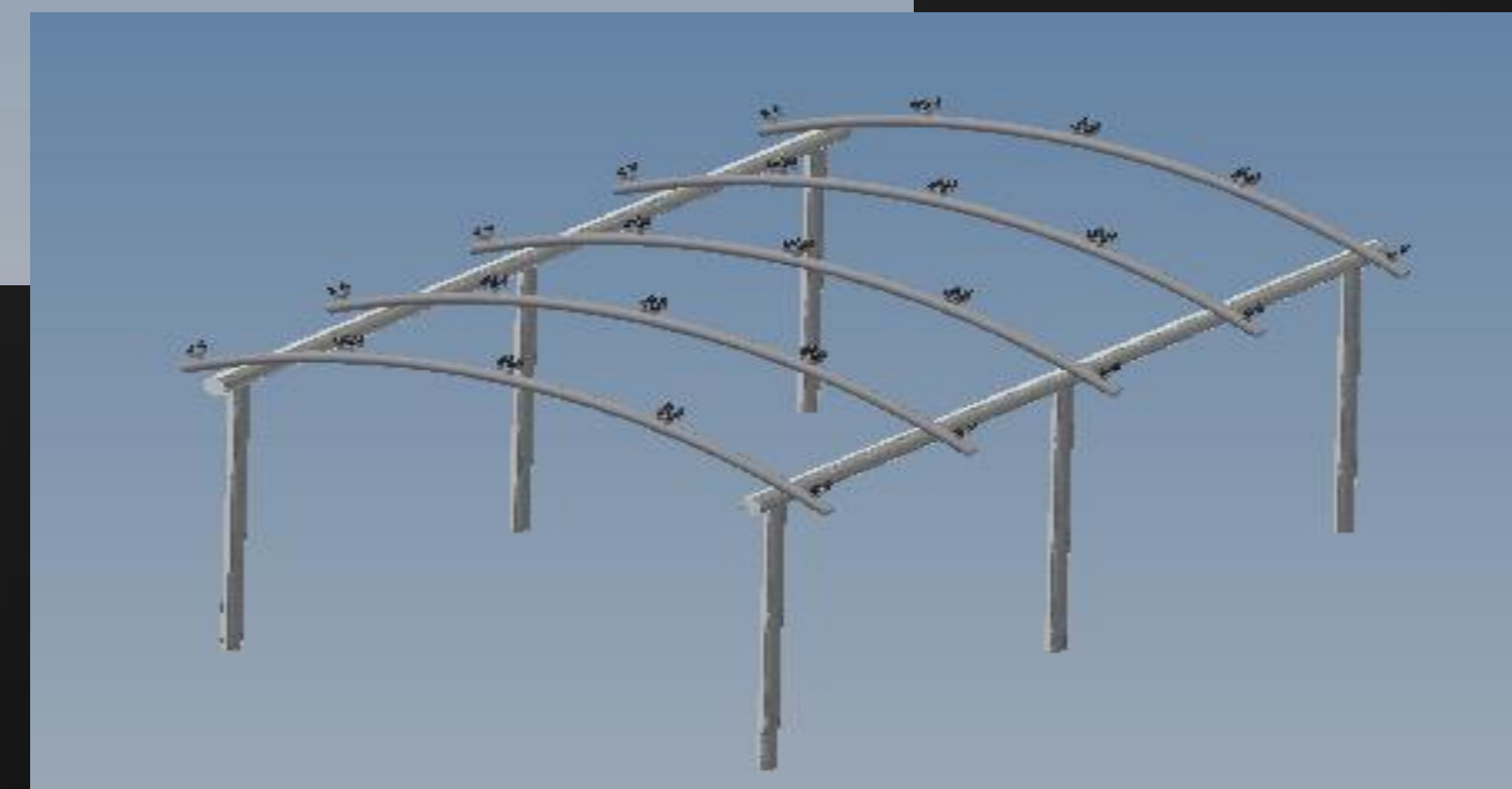
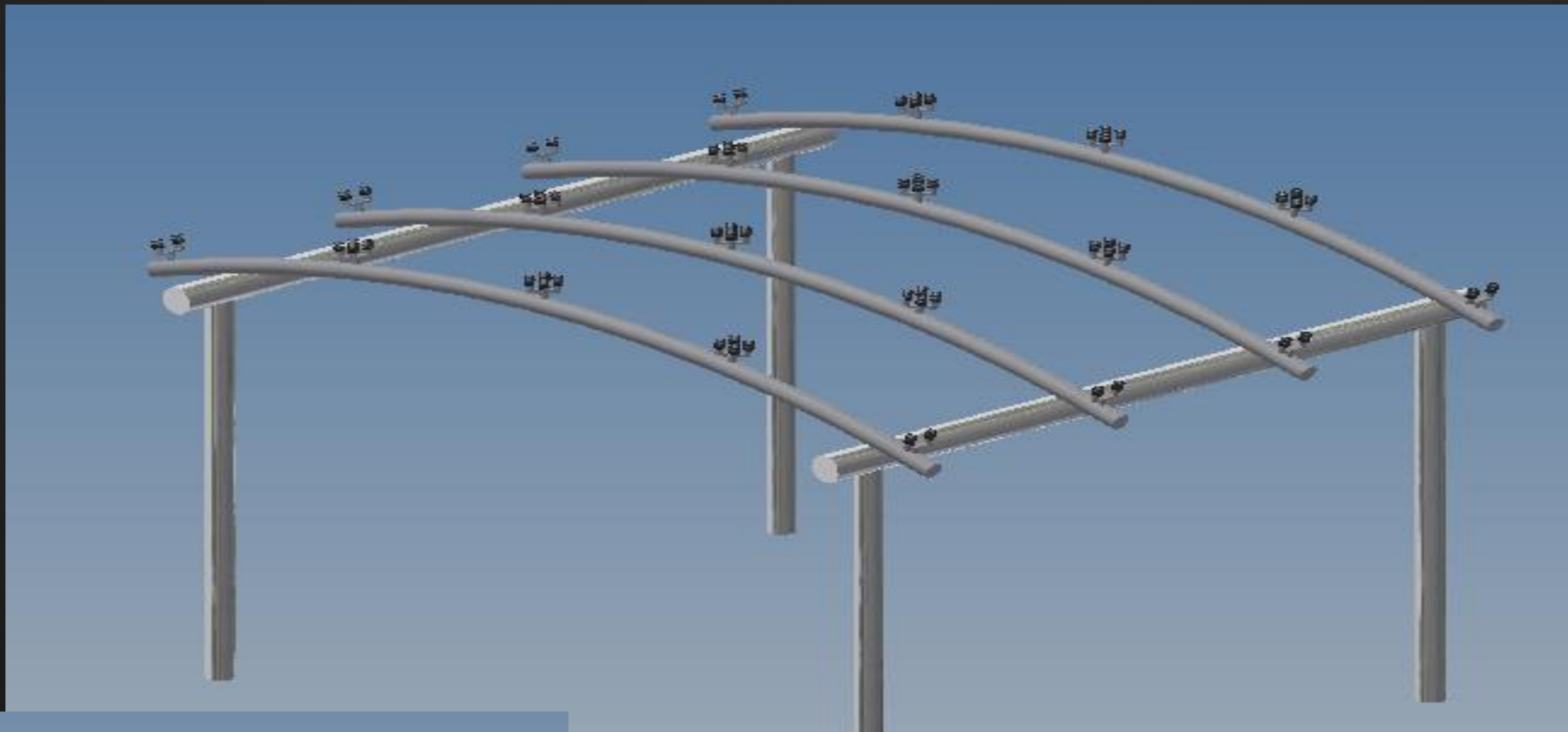


Image courtesy of Brimrock, BCD



ETO “Designs”

- Like a “class”
- Represent / manage Inventor objects:
 - In assembly modeling:
 - Assembly and part documents
 - Components
 - Constraints and patterns
 - In drawing “modeling”:
 - Drawing documents
 - Sheets, views
 - Dimensions, other annotations
 - Typically only “indirect” part modeling
- Also other non-Inventor things:
 - Databases
 - XML files
 - Word/Excel documents
 - Vault operations

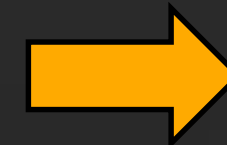
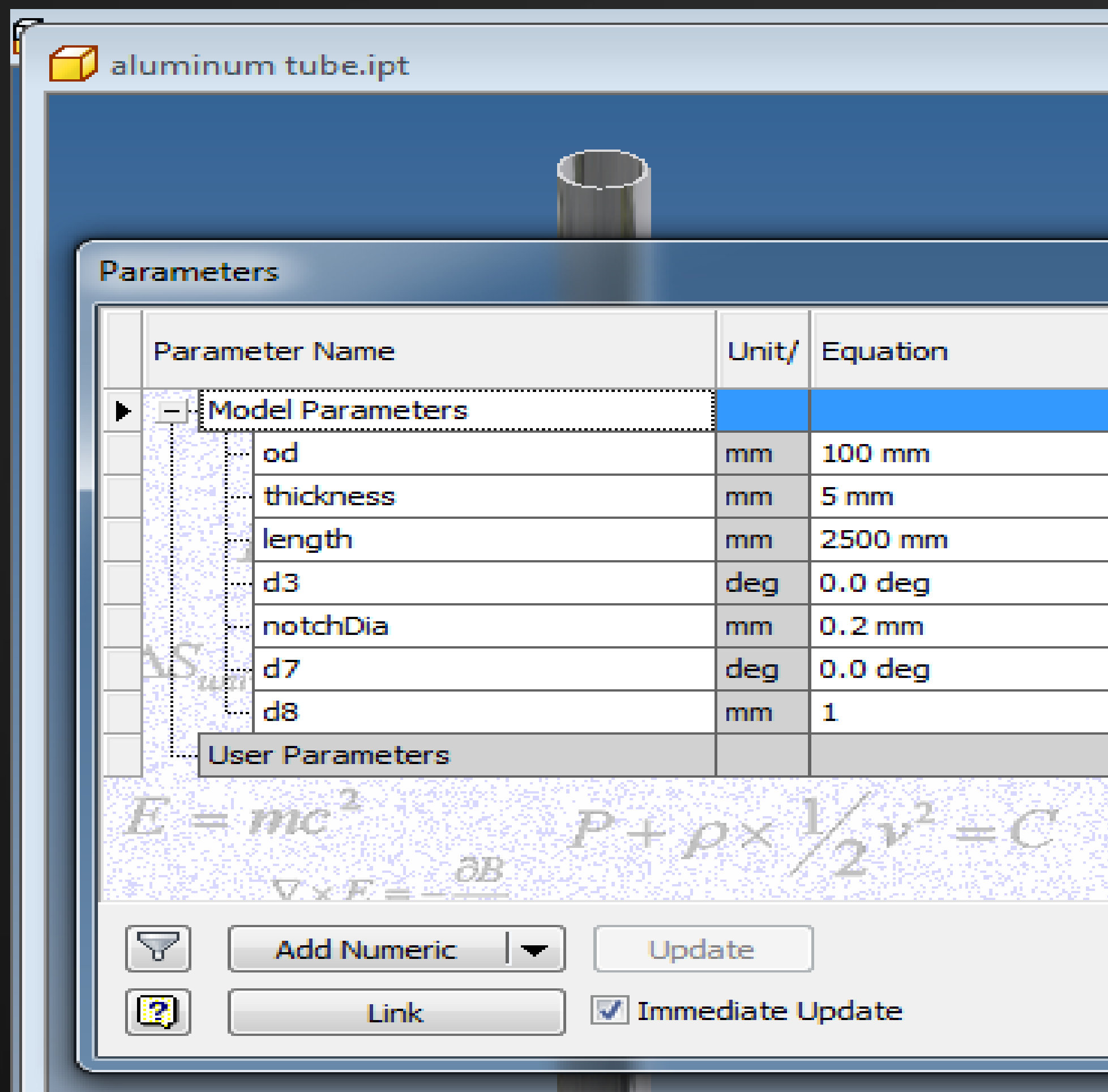
Assembly Modeling

- Part documents are “adopted”
 - “Wrapper” designs created automatically
 - Controls part parameters
 - Manages “member” files
 - Like iParts, but automatic member management
- Designs instantiated by “Child” rules

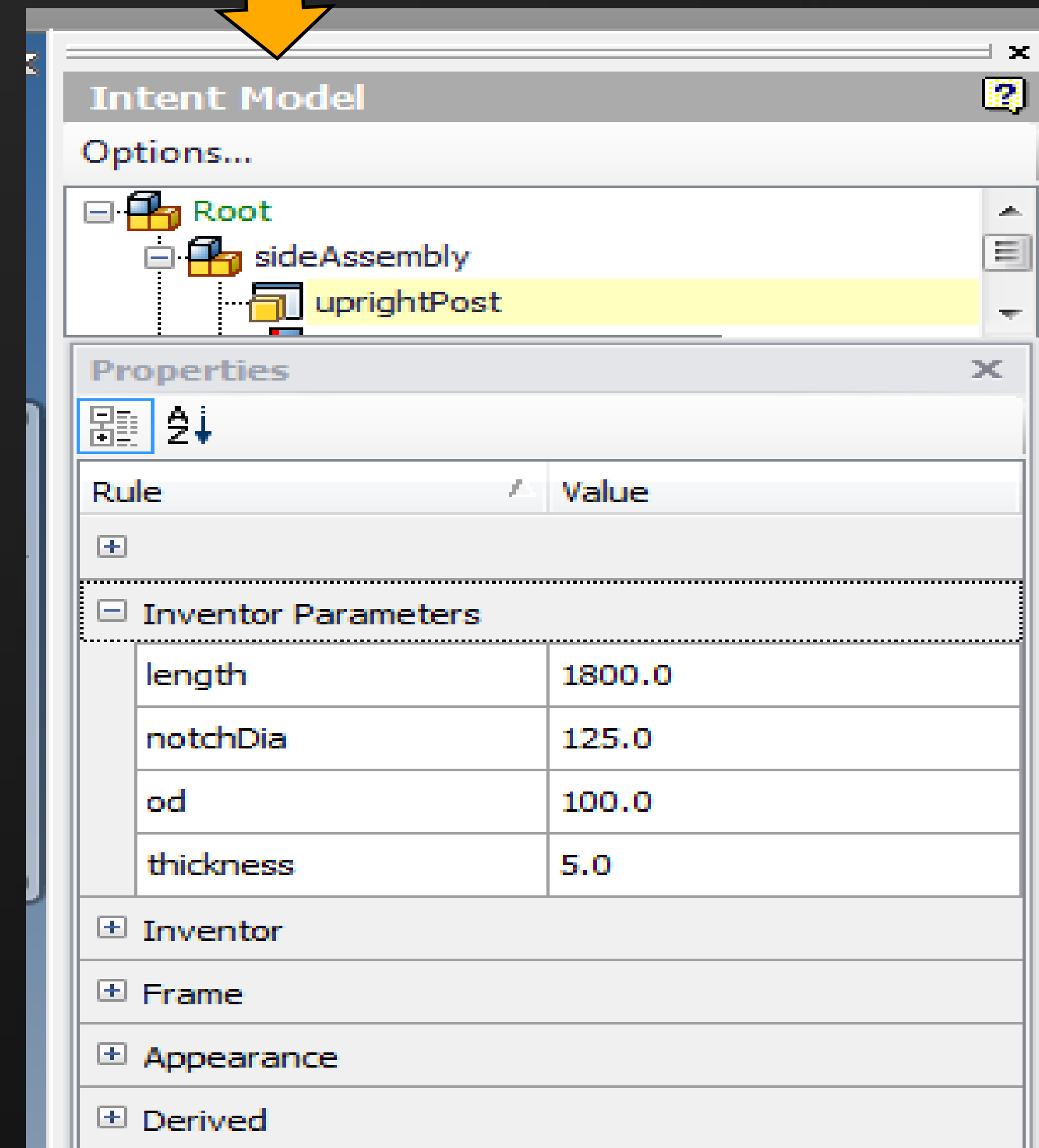
The screenshot shows the 'Inventor ETO Adoption Wizard' dialog box. The title bar reads 'Inventor ETO Adoption Wizard'. The main area is titled 'Component Settings' with the instruction 'Specify design name and parameters to adopt'. Below this is a table with columns: Type, Component Name, Intent Design, Parent Design, and File Name. The first row shows 'Part' as the type, 'aluminum tube.ipt' as the component name, 'aluminum_tube' as the intent design, 'Adopted Components' as the parent design, and 'C:\work\eto\samples\ET...y Files\aluminum tube.ipt' as the file name. Below this table is a 'Parameters' section with a table of parameters to be adopted. The parameters table has columns: Adopt?, Read Only?, Type, Name, Intent Rule, Expression, Value, and Comments. The parameters listed are: od (100 mm), thickness (5 mm), length (2500 mm), d3 (0.0 deg), notchDia (0.2 mm), d7 (0.0 deg), and d8 (1.000 mm). The 'Adopt?' column has checkboxes, and the 'Read Only?' column has checkboxes. The 'Intent Rule' column contains the names of the rules being adopted. The 'Expression' column contains the expressions for the rules. The 'Value' column contains the current values for the rules. The 'Comments' column is empty. At the bottom of the dialog are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

Type	Component Name	Intent Design	Parent Design	File Name
Part	aluminum tube.ipt	aluminum_tube	Adopted Components	C:\work\eto\samples\ET...y Files\aluminum tube.ipt

Adopt?	Read Only?	Type	Name	Intent Rule	Expression	Value	Comments
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Model	od	od	100 mm	100.000 mm	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Model	thickness	thickness	5 mm	5.000 mm	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Model	length	length	2500 mm	2500.000 mm	
<input type="checkbox"/>	<input type="checkbox"/>	Model	d3	d3	0.0 deg	0.00 deg	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Model	notchDia	notchDia	0.2 mm	0.200 mm	
<input type="checkbox"/>	<input type="checkbox"/>	Model	d7	d7	0.0 deg	0.00 deg	
<input type="checkbox"/>	<input type="checkbox"/>	Model	d8	d8	1	1.000 mm	



```
Child uprightPost As :aluminum_tube
    notchDia = crossBeam.dia
    length = height
    od = uprightPostDiameter
End Child
```



Rules control the content

- Rules can be “basic” or “Child”
- Basic rules can be
 - Parameter (independent variable)
 - or non-parameter (dependent variable)
- Child rules define “hierarchy”

```
Design busShelter : ETO_Canopy2Root IvAssemblyDocument
```

```
Parameter Rule height As Number = 2200
Parameter Rule width As Number = 2000
Parameter Rule depth As Number =
Parameter Rule depthInset As Num
Parameter Rule sideOverhang As N
Parameter Rule centerIncremental
Parameter Rule nTiles_lateral As
Parameter Rule maxDepthBetweenRo
```

```
Rule beamAvailableDepth As Numbe
Rule nBeams As Integer = 1 + cei
Rule roofBeamSpacing As Number =
```

```
Child sideAssembly As :sideAssem
    grounded? = True
    height = height
    length = depth
    inset = depthInset
End Child
```

sideAssembly x

```
Design sideAssembly : ETO_Canopy2Root IvAssemblyDocument
```

```
Parameter Rule height As Number = 2200
Parameter Rule length As Number = 2000
Parameter Rule inset As Number = 100
Parameter Rule maxLengthBetweenPosts As Number = 3000
Parameter Rule uprightPostDiameter As Number = 100
Parameter Rule crossBeamDiameter As Number = 125
```

```
Rule postAvailableLength As Number = length-(inset*2)
Rule nPosts As Integer = 1 + ceiling(postAvailableLength/maxLengthBetweenPosts)
Rule actualSpacing As Number = postAvailableLength/(nPosts-1)
```

```
Child uprightPost As :aluminum_tube
    notchDia = crossBeam.dia
    length = height
    od = uprightPostDiameter
End Child
```

Use IDE to edit/debug rules

```
sideAssembly : ETO_Canopy2Root IvAssemblyDocument

Parameter Rule height As Number = 2200
Parameter Rule length As Number = 2000
Parameter Rule inset As Number = 100
Parameter Rule maxLengthBetweenPosts As Number = 3000
Parameter Rule uprightPostDiameter As Number = 100
Parameter Rule crossBeamDiameter As Number = 125

Rule postAvailableLength As Number = length-(inset*2)
Rule nPosts As Integer = 1 + ceiling(postAvailableLength/maxLengthBetweenPosts)
Rule actualSpacing As Number = postAvailableLength / (nPosts-1)

Child uprightPost As :aluminum_tube
    notchDia = crossBeam.dia
    length = height
    od = uprightPostDiameter
End Child
```

Intent Model

Options...

- Root
 - sideAssembly
 - uprightPost
 - postVerticalConstraint
 - postPattern
 - crossBeam
 - crossBeamVerticalConstraint
 - crossBeamLateralConstraint
 - insetConstraint
 - sidePattern
 - roofBeam

Properties

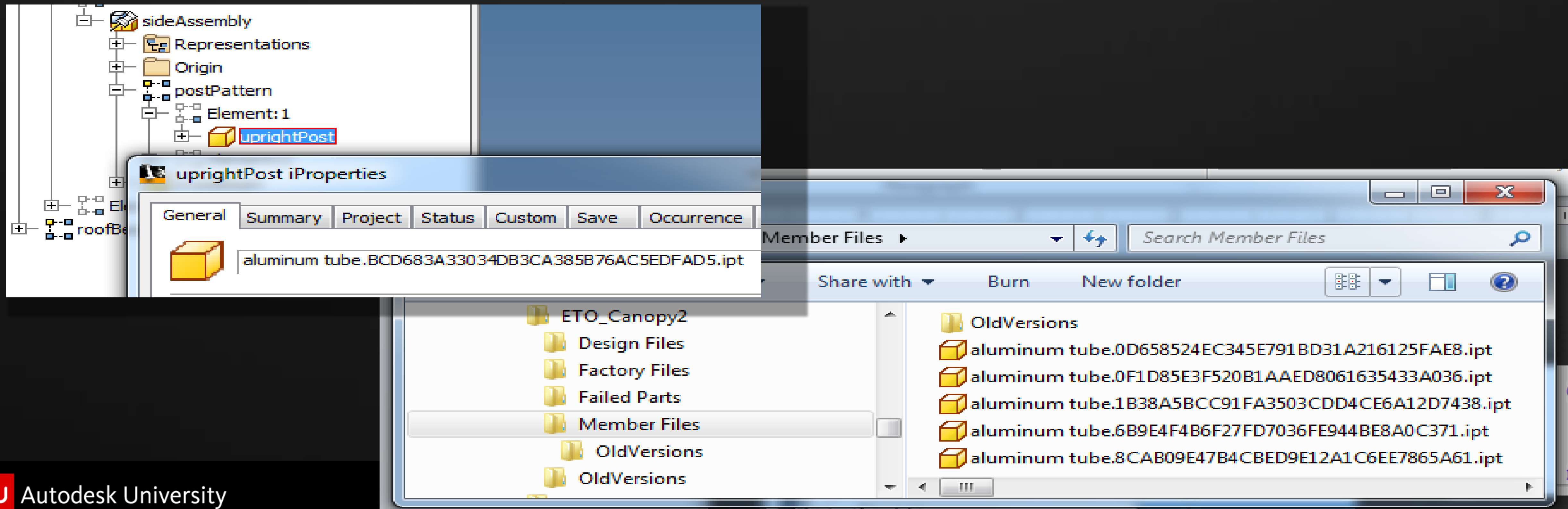
Rule	Value
actualSpacing	2600.0
crossBeamDiameter	125.0
height	1800.0
inset	200.0
length	3000.0
maxLengthBetweenP...	3000.0
nPosts	2
postAvailableLength	2600.0
uprightPostDiameter	100.0

Inventor

Frame

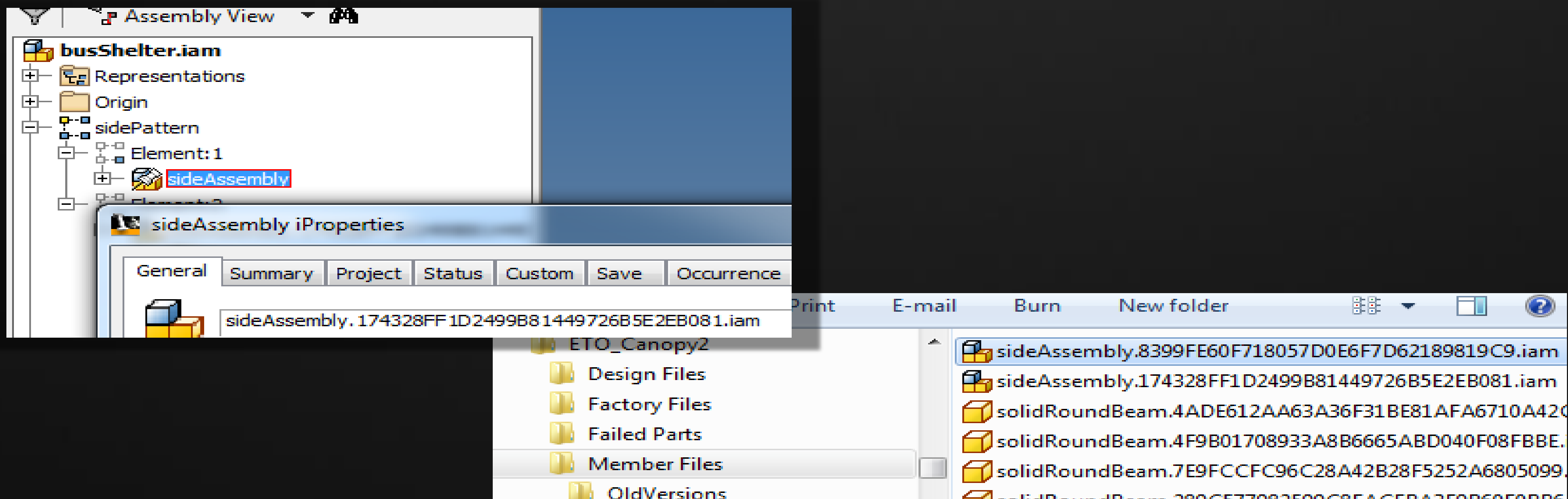
Member File Management

- As parts' values change:
 - new member files are created when necessary
 - occurrences are replaced with the “correct” member file
- *ETO keeps track of all this, so you don't have to*



Ditto for Assemblies

- As assemblies' content changes (occurrences, constraints, etc.):
 - new member files are created when necessary
 - occurrences are replaced with the “correct” member file
- *ETO keeps track of all this, so you don't have to*



Ditto for Drawings

- As drawings' contents changes (e.g. sheets):
 - sheets are added or deleted as necessary
- As sheet content changes (views, annotations)
 - Views are added/deleted/updated as necessary
 - Annotations are added/deleted/updated as necessary
- *ETO keeps track of all this, so you don't have to*

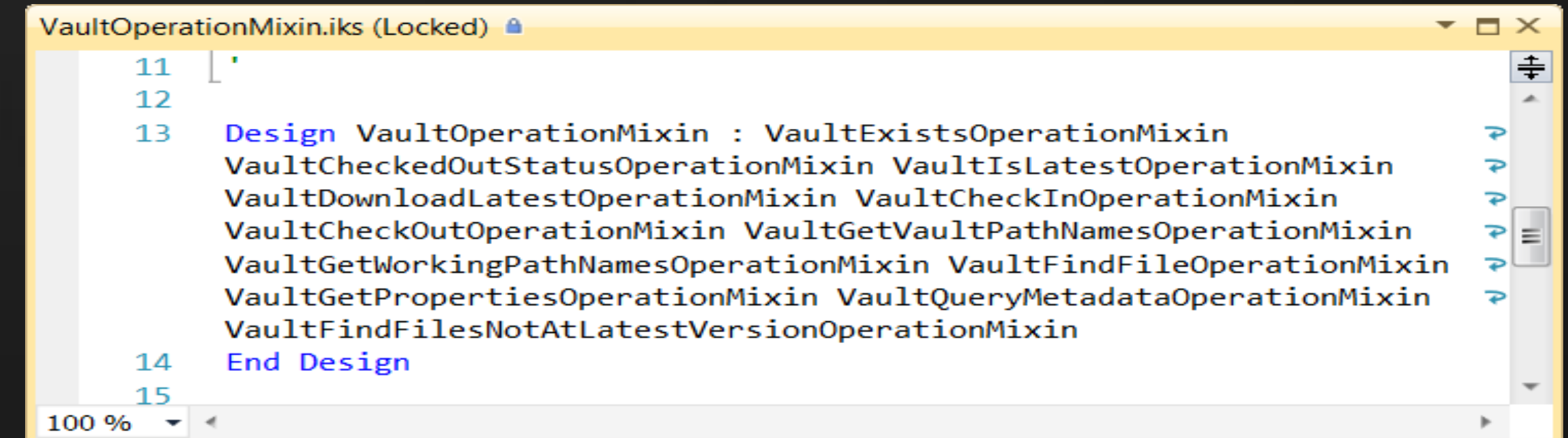
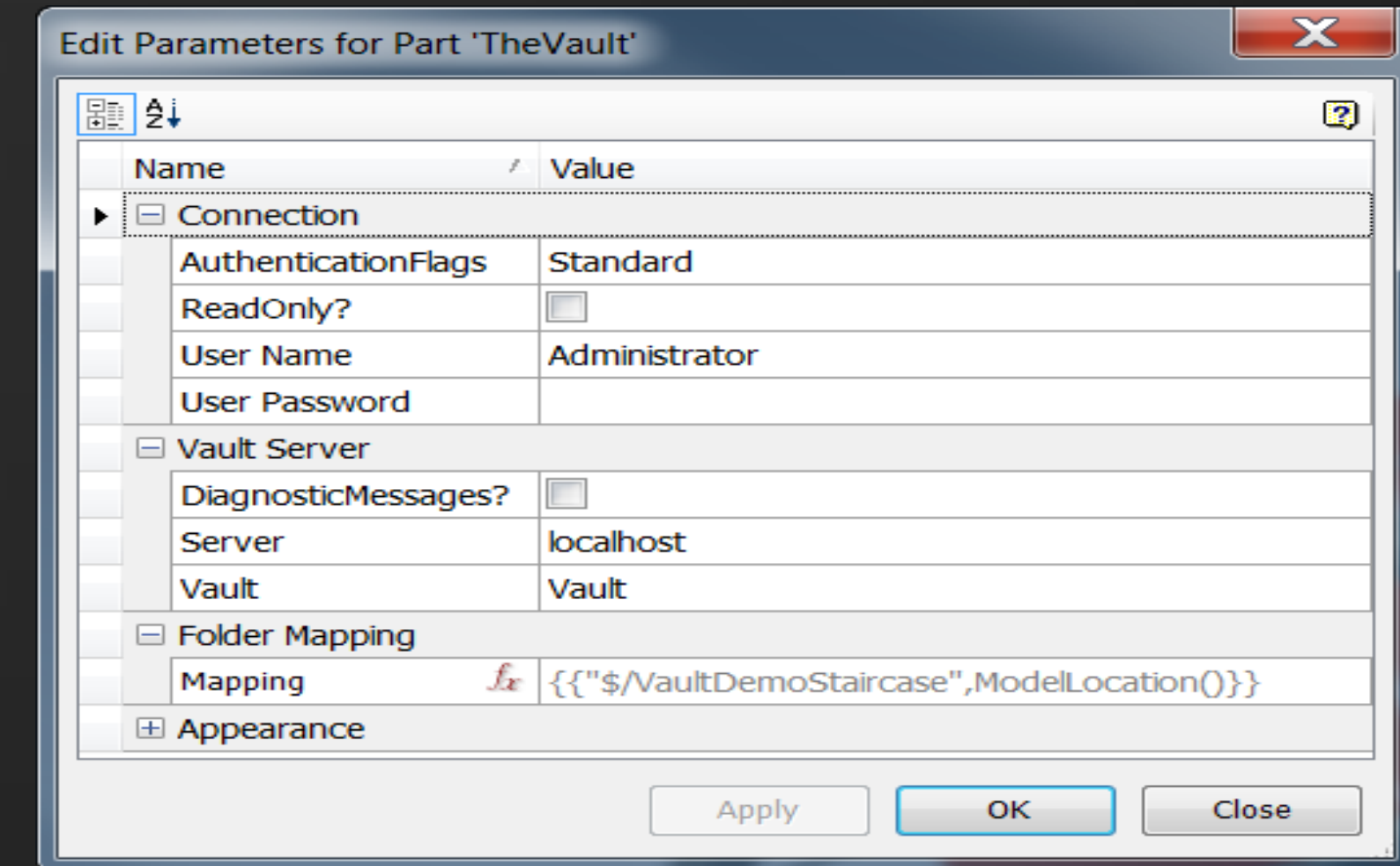
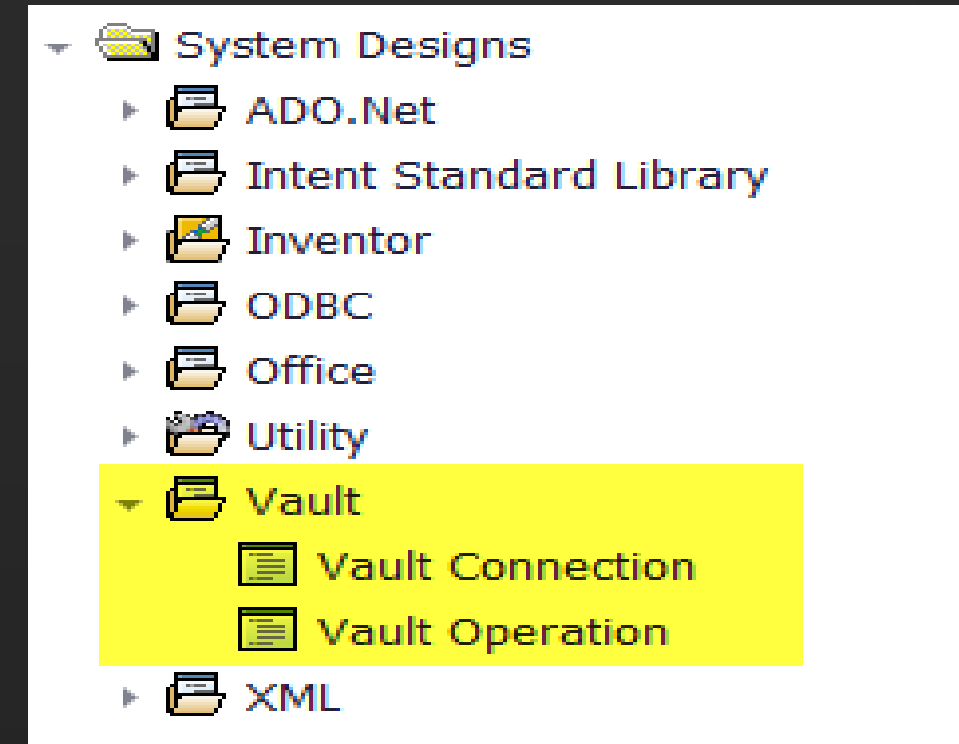
Vault Integration

Vault Connection

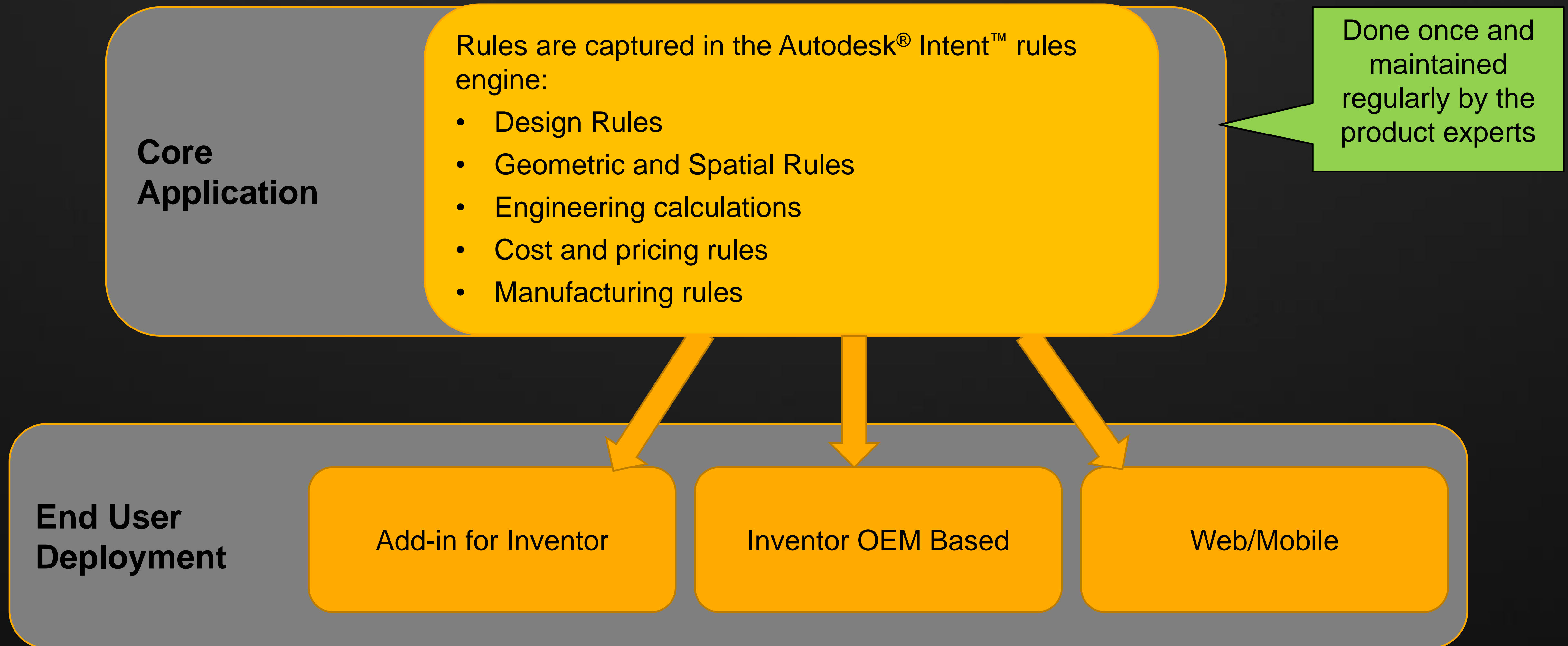
- Log in
- Log out

Vault Operations

- Check documents in and out of the vault
- Check if files exist in the vault and use existing rather than generating new members
- Query the vault:
 - Check if a file exists
 - Find a file
 - Get file properties from files in the Vault
 - Check for a files checked out status
 - Check if a file is the latest version
 - Get the full path of a named file from the Vault
 - Get Vault Meta Data: date created, date modified, lifecycle state (Released etc.)
- Get the full Working Path of a named file in the local workspace
- Find files in the local workspace that are not the latest version and replace with the latest version from the Vault
- Download the latest version of a file from the vault



Solution Architecture



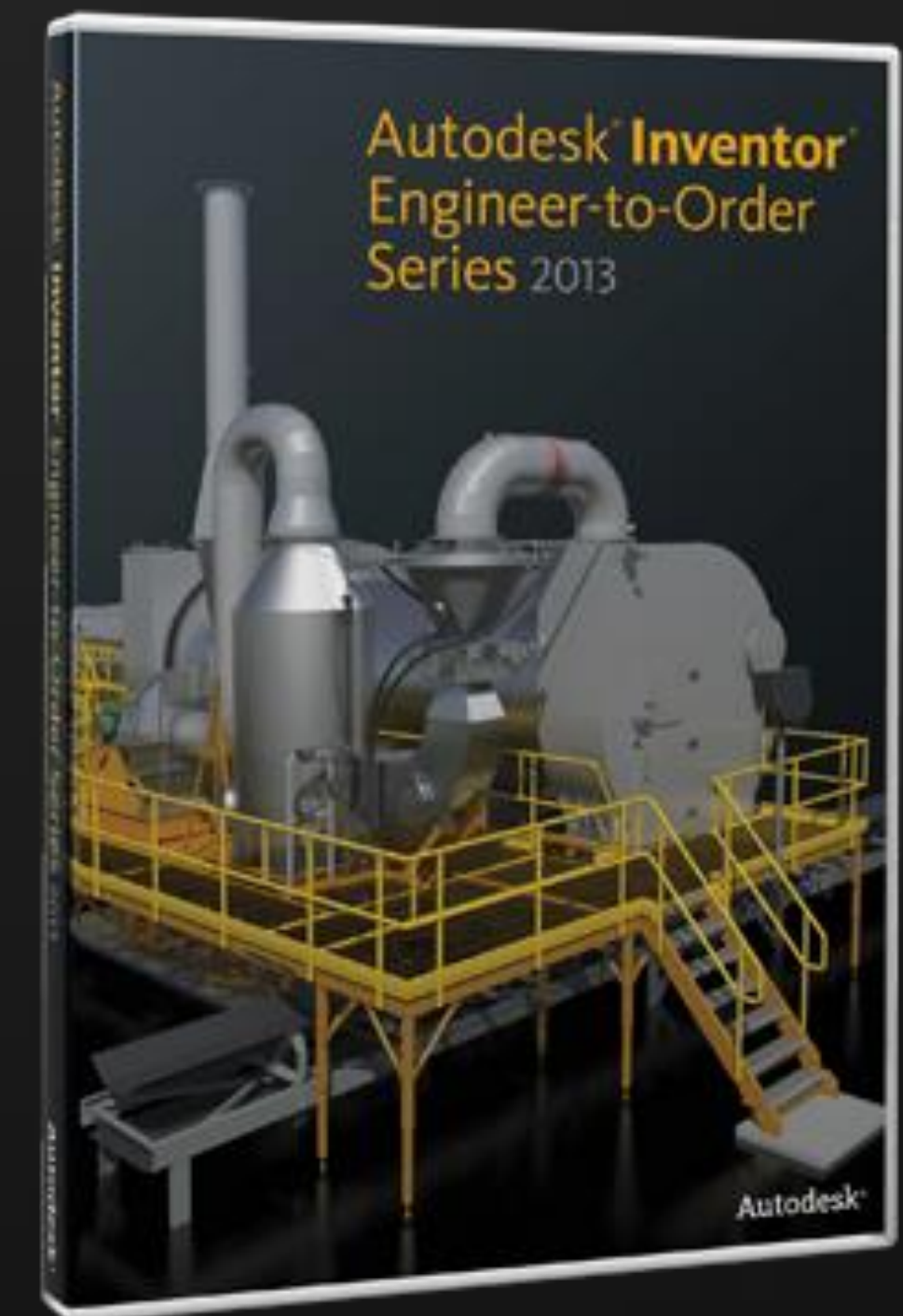
Inventor ETO Server

Inventor ETO Server

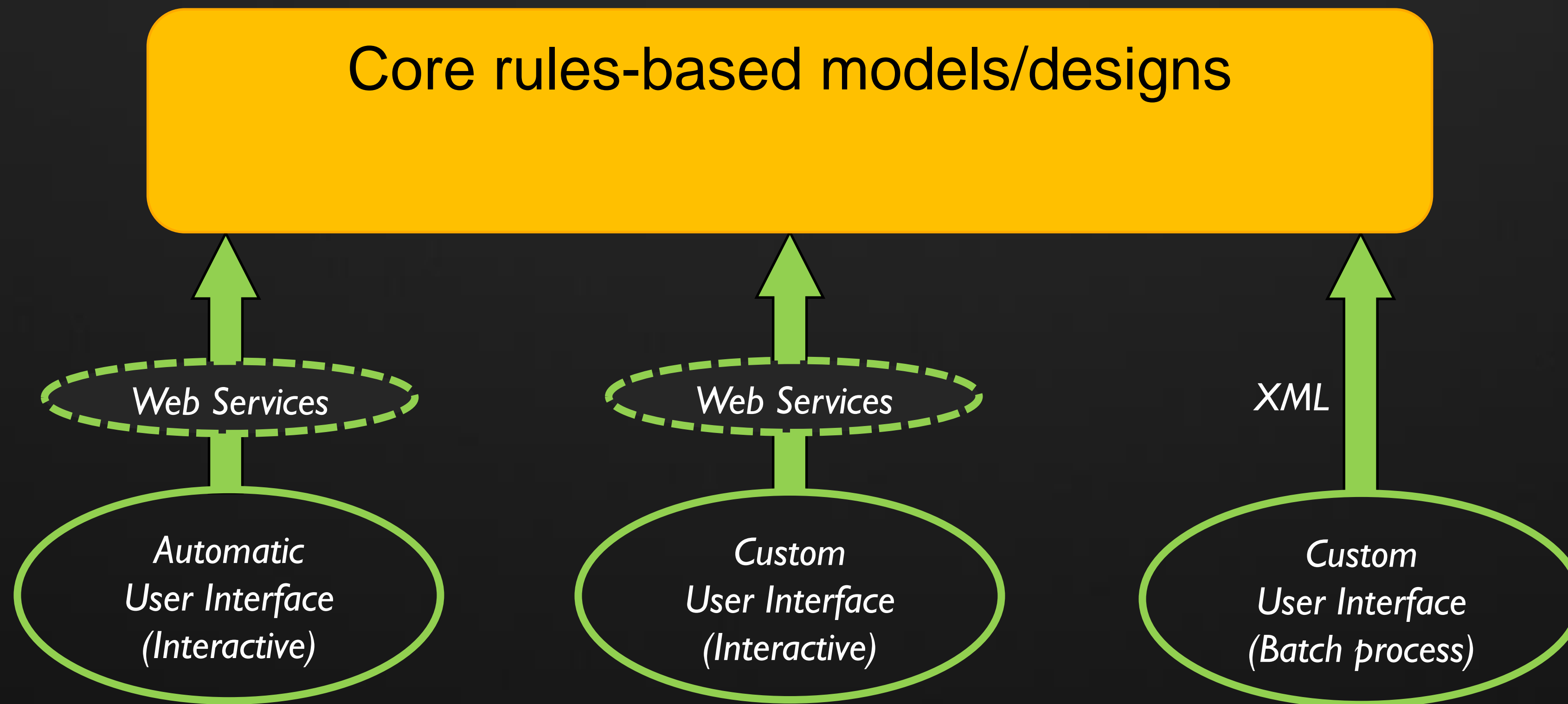
Platform for deploying web, mobile and client-server applications

Contains

- Inventor Server
- ETO Rules Engine
- iLogic
- Web Services
- Server Farm Management



3 Modes of Accessing Inventor ETO Server



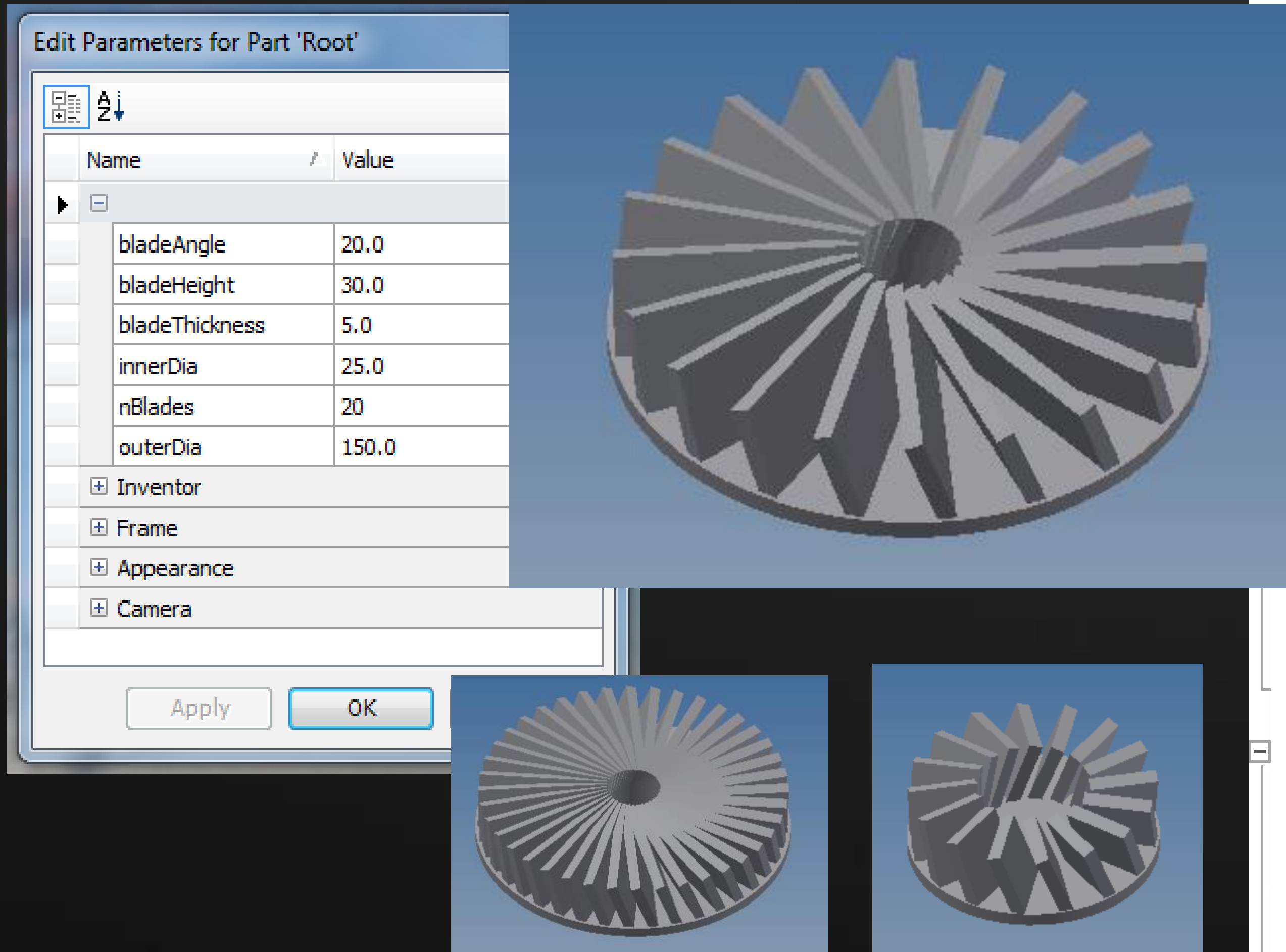
Demos

Mode 1: Custom UI, Batch

Applicability:

- Workflow is such that all inputs can be batched up
 - Some other means of gathering configuration input already exists
 - No requirement for real-time visualization based on user inputs
-
- Easiest to implement
 - No web services knowledge required
 - End user experience not as good
 - Web front end development effort required

Really Simple Example



```
Design turbine : turbineVariations IvAssemblyDocument
```

```
Parameter Rule innerDia As Number = 25  
Parameter Rule outerDia As Number = 150  
Parameter Rule bladeThickness As Number = 5  
Parameter Rule bladeHeight As Number = 30  
Parameter Rule nBlades As integer = 20  
Parameter Rule bladeAngle As Number = 20
```

```
Rule bladeLength As Number = (outerDia - innerDia)/2
```

```
Method rotateBlade(idx As Integer) As Frame ...
```

```
Child blade As :IvBlock, Quantity = nBlades  
    width = bladeLength  
    height = bladeThickness  
    length = bladeHeight  
    ReferenceFrame = rotateBlade(child.index)  
    ignorePosition? = False  
    grounded? = True  
End Child
```

```
Child base As :IvCylinder  
    diameter = outerDia*1.02  
    height = 5  
    bottomPoint = origin -vector(0,0,bladeHeight /2)  
    ignorePosition? = false  
    grounded? = true  
End Child
```

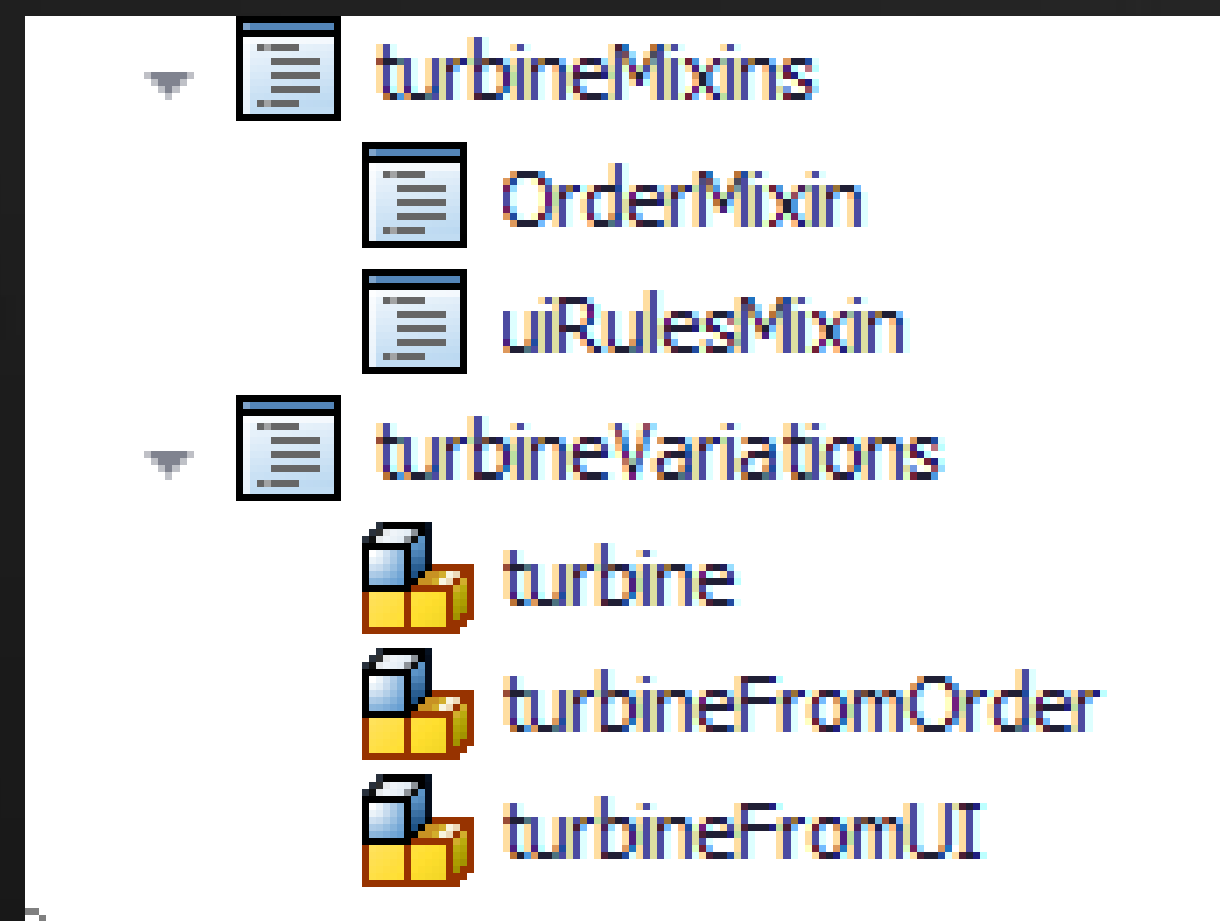

Make JPEG from rules

```
Rule cameraFit? As Boolean = True
Rule cameraViewOrientation As Name = :isoTopLeft
Uncached Rule makeJPEG As boolean = saveImage("c:\temp\output.jpg", 600, 600)
```

ETO Console

```
ETO Series Studio 6.1.113
Copyright © 2012 Autodesk, Inc.
Intent >makeJPEG
True
Intent >|
```

Variations for Deployment Modes



```
Design turbine : turbineVariations IvAssemblyDocument
```

```
Parameter Rule innerDia As Number = 25
Parameter Rule outerDia As Number = 150
Parameter Rule bladeThickness As Number = 5
Parameter Rule bladeHeight As Number = 30
Parameter Rule nBlades As integer = 20
```

```
'! Intent Language(tm) 3.0
```

```
Design turbineFromOrder : turbineVariations orderMixin turbine
```

```
End Design
```

```
'! Intent Language(tm) 3.0
```

```
Design turbineFromUI : turbineVariations uiRulesMixin UiPartMixin turbine
```

```
End Design
```

Scenario: Pre-Configured Orders

- XML “order file” from some other source

```
<?xml version="1.0" encoding="utf-8"?>
<order orderID="123">
  <customer>
    <company name="Acme Power Transfer, Inc"/>
    <contact name="Jon Balgley"/>
  </customer>
  <parameters>
    <parameter name="nBlades" value="22"/>
    <parameter name="bladeAngle" value="35"/>
    <parameter name="innerDia" value="50"/>
    <parameter name="outerDia" value="100"/>
  </parameters>
</order>
```


Reading Data from XML

```
<?xml version="1.0" encoding="utf-8"?>
<order orderID="123">
  <customer>
    <company name="Acme Power Transfer, Inc"/>
    <contact name="Jon Balgley"/>
  </customer>
  <parameters>
    <parameter name="nBlades" value="22"/>
    <parameter name="bladeAngle" value="35"/>
    <parameter name="innerDia" value="50"/>
    <parameter name="outerDia" value="100"/>
  </parameters>
</order>
```

Design OrderMixin : turbineMixins

innerDia	50
jpeg	
nBlades	22
orderFile	"C:\work\eto\samples\fo
outerDia	100
paras	{Root.orderData.order.pa
paraVals	{:nBlades, 22, :bladeAngl

End C

▼ Inventor

▼ Frame

'Retr

Rule paras As List = OrderData.order.parameters

Rule paraVals As List

For Each Para In paras

Dim attrs As List = Para.attributes

'Turn XML data into Intent 'plist'

'Note: no error checking on input data in this example

paraVals = paraVals + {makeName(nth(2, attrs)), EvaluateString(nth(4, attrs))}

Next Para

End Rule

Root[paraVals]

:nBlades
22
:bladeAngle
35
:innerDia
50
:outerDia
100

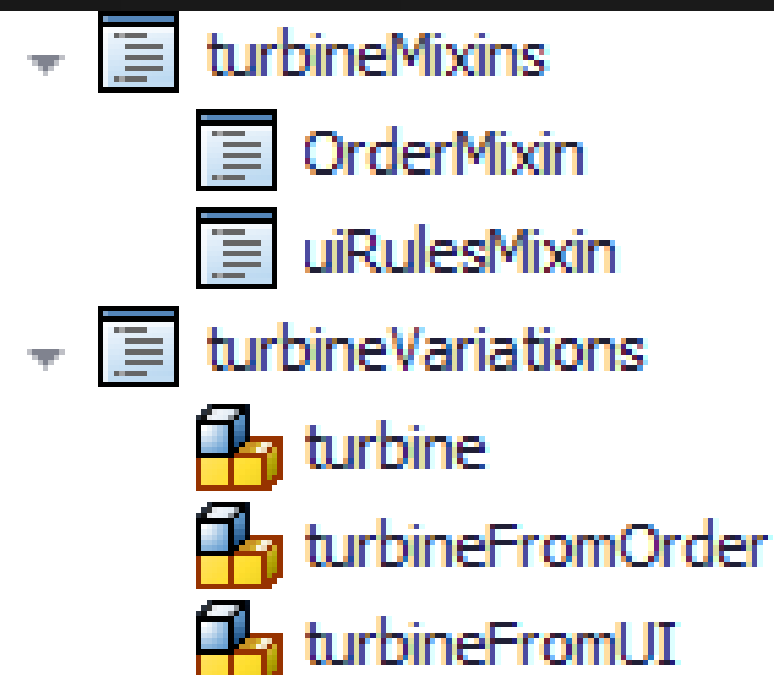
“Adapter” from XML to individual params

```
<?xml version="1.0" encoding="utf-8"?>
<order orderID="123">
  <customer>
    <company name="Acme Power Transfer, Inc"/>
    <contact name="Jon Balgley"/>
  </customer>
  <parameters>
    <parameter name="nBlades" value="22"/>
    <parameter name="bladeAngle" value="35"/>
    <parameter name="innerDia" value="50"/>
    <parameter name="outerDia" value="100"/>
  </parameters>
</order>
```

```
Design OrderMixin : turbineMixins
```

'Overrides

```
Rule nBlades As Integer = findInPlistEx(:nBlades, paraVals, default := 20)
Rule innerDia As Integer = findInPlistEx(:innerDia, paraVals, default := 25)
Rule outerDia As Integer = findInPlistEx(:outerDia, paraVals, default := 150)
Rule bladeThickness As Integer = findInPlistEx(:bladeThickness, paraVals, default := 5)
Rule bladeAngle As Integer = findInPlistEx(:bladeThickness, paraVals, default := 20)
```



```
Design turbineFromOrder : turbineVariations orderMixin turbine
```

```
End Design
```

Mode 1a: Using ETO Server Directly

Applicability:

- Testing
 - Hand-processing
-
- Not using any web stuff yet!

Anatomy of a Task File

Setup:

- IntentServer
- Project (IPJ)
- NewModel, etc. – sets root design (and IAM)

Commands:

- SetRuleValue
- Evaluate
 - RenderSelf

Case-sensitive!

```
<?xml version="1.0" encoding="utf-8"?>
<IntentServer name="Turbine" outputDir="c:\temp\Output">
  <Project name="Fractals">
    <NewModel name="TestTurbine" baseDesign="Turbine">
      <SetRuleValue rule="nBlades" expression="40"/>
      <Evaluate expression="renderSelf"/>
      <Evaluate expression="jpeg"/>
    </NewModel>
  </Project>
</IntentServer>
```

Run ETO from command prompt

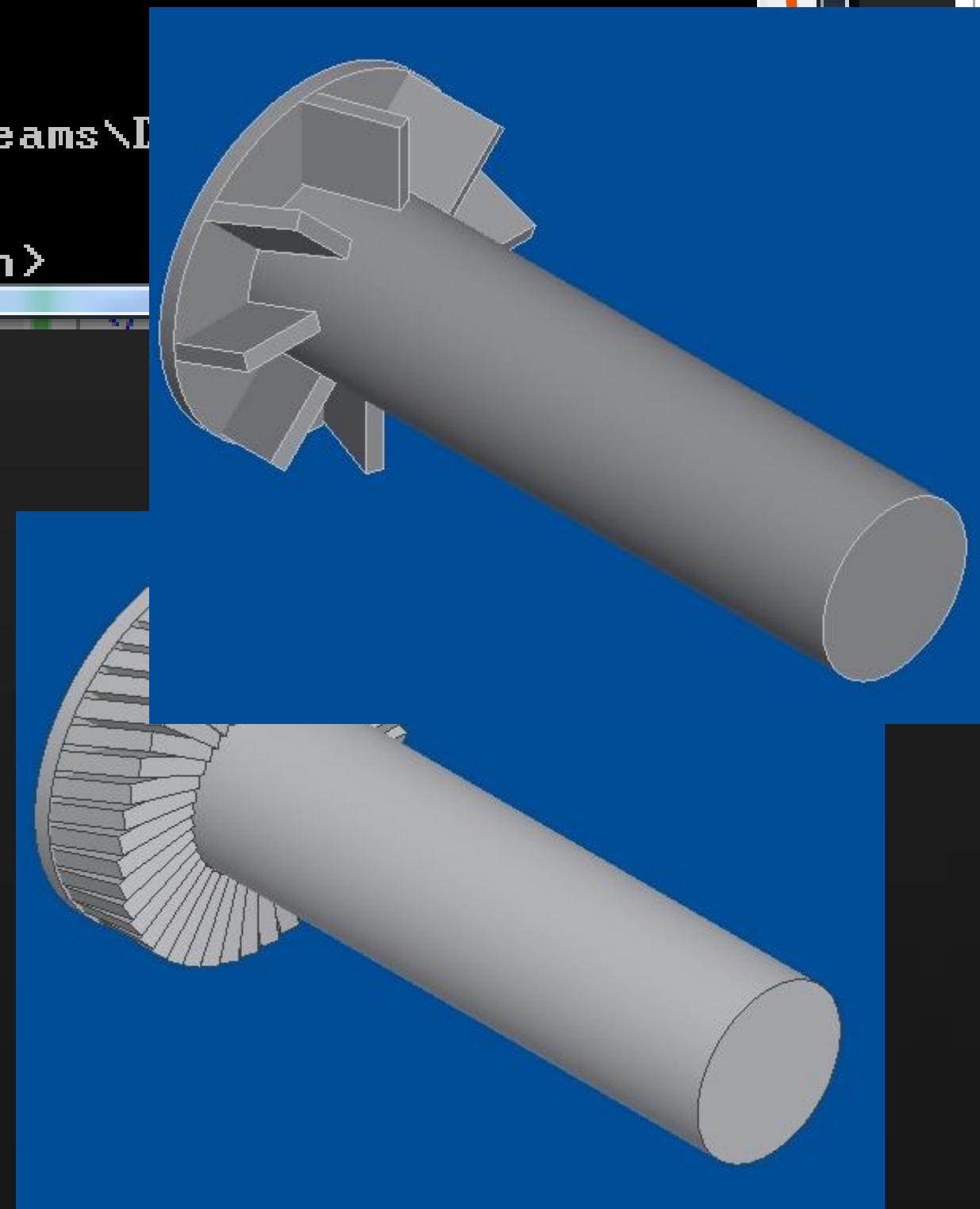
```
C:\Program Files\Autodesk\Inventor ETO Server 2013\Bin>InventorETOServer.exe -b  
c:\work\eto\src\Intent\Streams\Deimos\7.0\dev\tp1\Samples\Internal\turbine\order  
Data\task.xml  
Autodesk Inventor ETO Server 6.1.0.106  
Copyright (c) 2012 Autodesk, Inc.
```

```
Creating Inventor Server object...  
Initializing Inventor Server...
```

```
<?xml version="1.0" encoding="utf-8"?>  
<order orderID="456">  
  <customer>  
    <company name="Acme Power Transfer, Inc"/>  
    <contact name="Jon Balgley"/>  
  </customer>  
  <parameters>  
    <parameter name="nBlades" value="8"/>  
    <parameter name="bladeAngle" value="35"/>  
    <parameter name="innerDia" value="50"/>  
    <parameter name="outerDia" value="100"/>  
  </parameters>  
</order>
```

```
<?xml  
<order  
  <cu  
    <  
    <  
  </cu  
  </parameters>  
<pa  
</order>  
<  
  <parameter name="bladeAngle" value="35"/>  
  <parameter name="innerDia" value="50"/>  
  <parameter name="outerDia" value="100"/>  
</parameters>  
</order>
```

```
<?xml version="1.0" encoding="utf-8"?>  
<IntentServer name="Turbine" outputDir="C:\temp\output">  
  <Project name="turbine">  
    <NewModel name="TestTurbine" baseDesign="turbineFromOrder">  
      <SetRuleValue rule="orderFile" expression="C:\...\order123.xml"/>  
      <Evaluate expression="renderSelf"/>  
      <Evaluate expression="makeJPEG"/>  
    </NewModel>  
  </Project>  
</IntentServer>
```



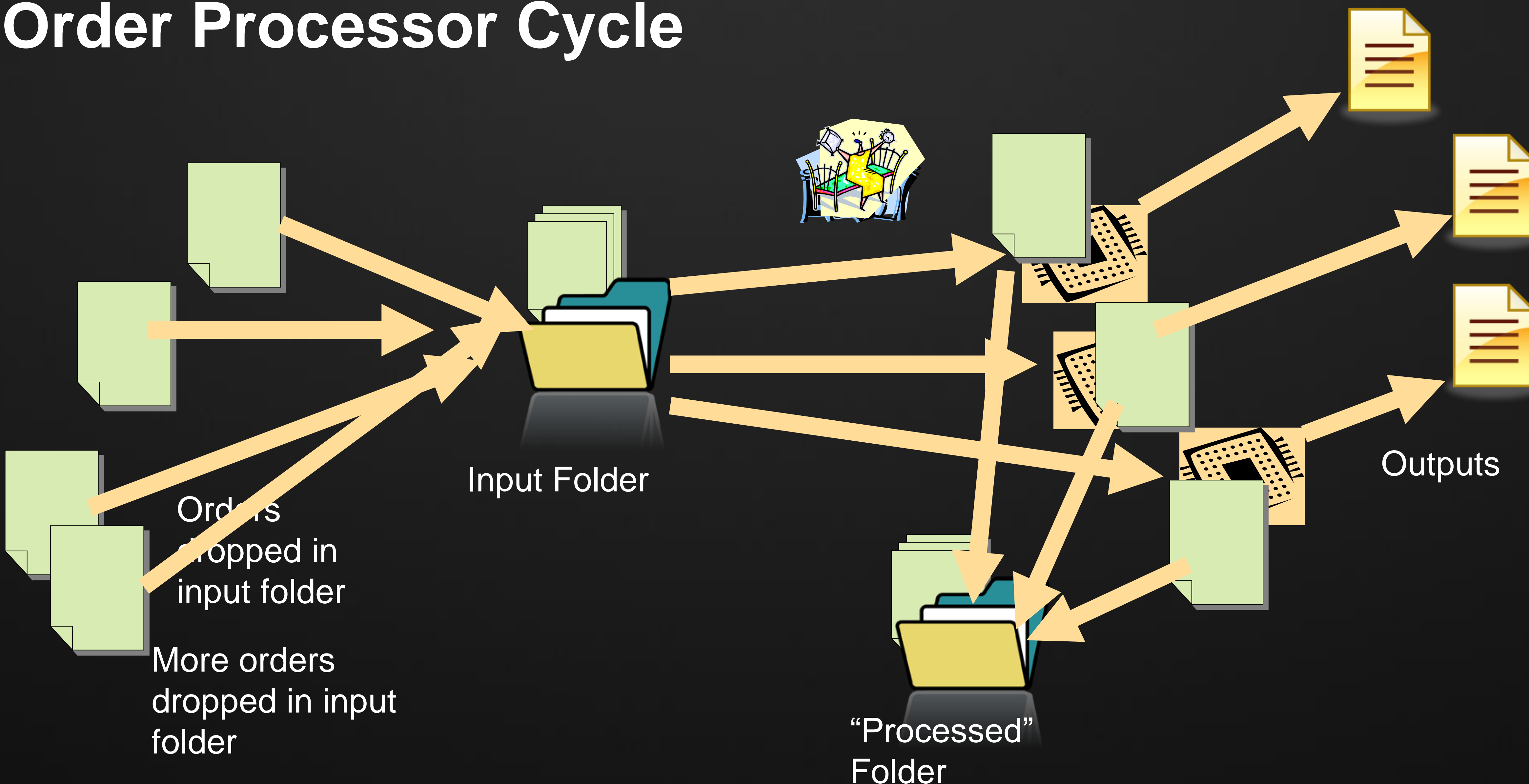
Summary

- Simple “batch” processor
 - Takes “task file” with simple ETO commands
 - Often “real” data comes from separate data file
- No connection to web server

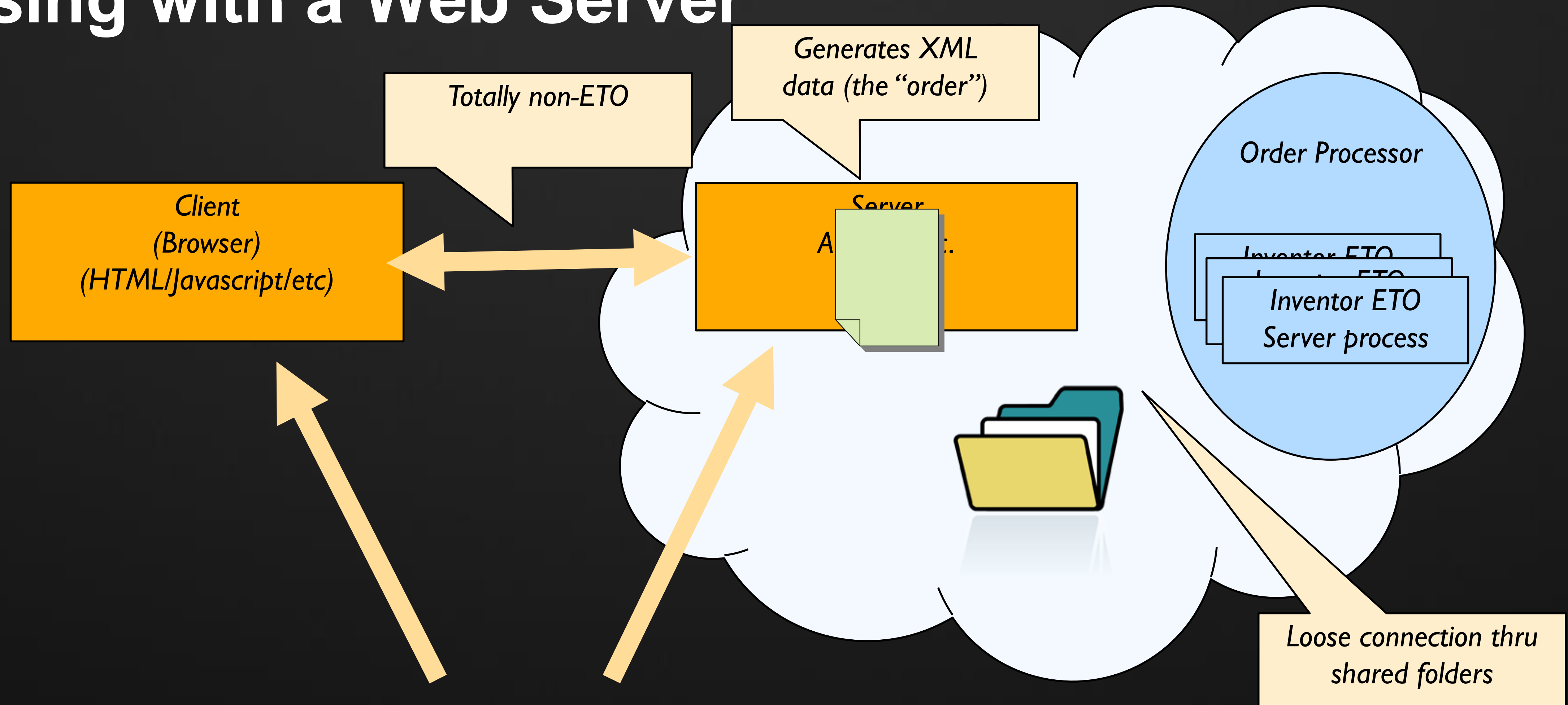
Mode 1b: Order Processor

- Watches folder for new files
- Runs multiple servers (on one host) with desired parameters
- Typically “real” parameters come from the “order file”
- Runs multiple orders in same session to optimize performance

Order Processor Cycle



Using with a Web Server



These are required, but not part of ETO

Summary

- Simple “infinite loop” wrapper around “batch” processor
- No direct connection to web server

Mode 2: Custom UI, Interactive

Applicability:

- Real-time visualization based on user input
 - Validation of every user input
 - Interactive application
 - Workflow requires a custom web interface
-
- Limited viewing technology:
 - Images
 - DWF (DWF viewer supported only on Internet Explorer, requires download)
 - Web front end development effort required

Autodesk Inventor Engineer-to-Order Demos

Equipment placement and throughput optimization

Reset Generate Quote Show Drawing

Components

Customer

Contact: John Doe

Company: Big Company

Sales Rep: Jane Doe

Conveyor

Top Of Roller: 1000

Section Length: 3000

Drive Side: ☐ Left ☒ Right

Conveyed Part

Length: 500

Width: 300

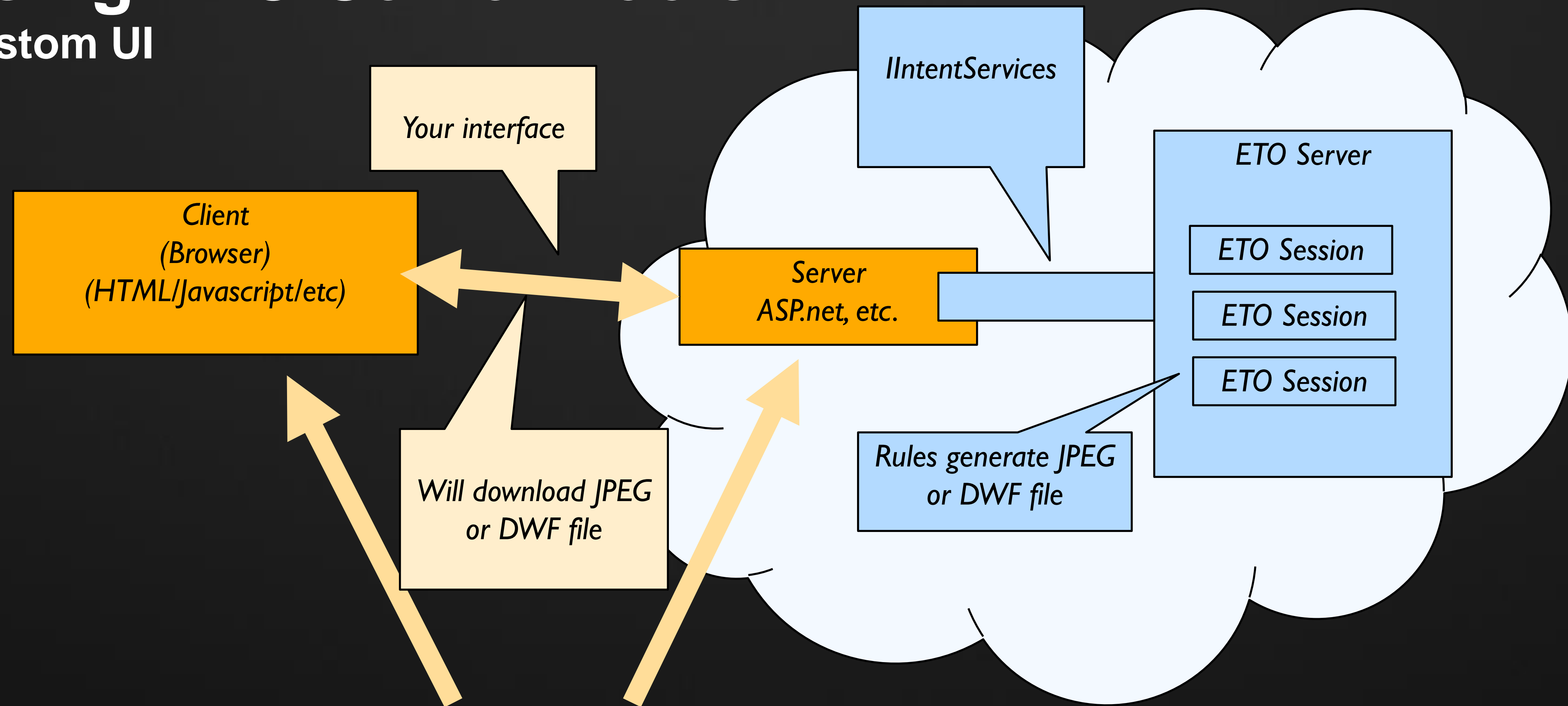
Weight: 30

Total Price

\$360.15

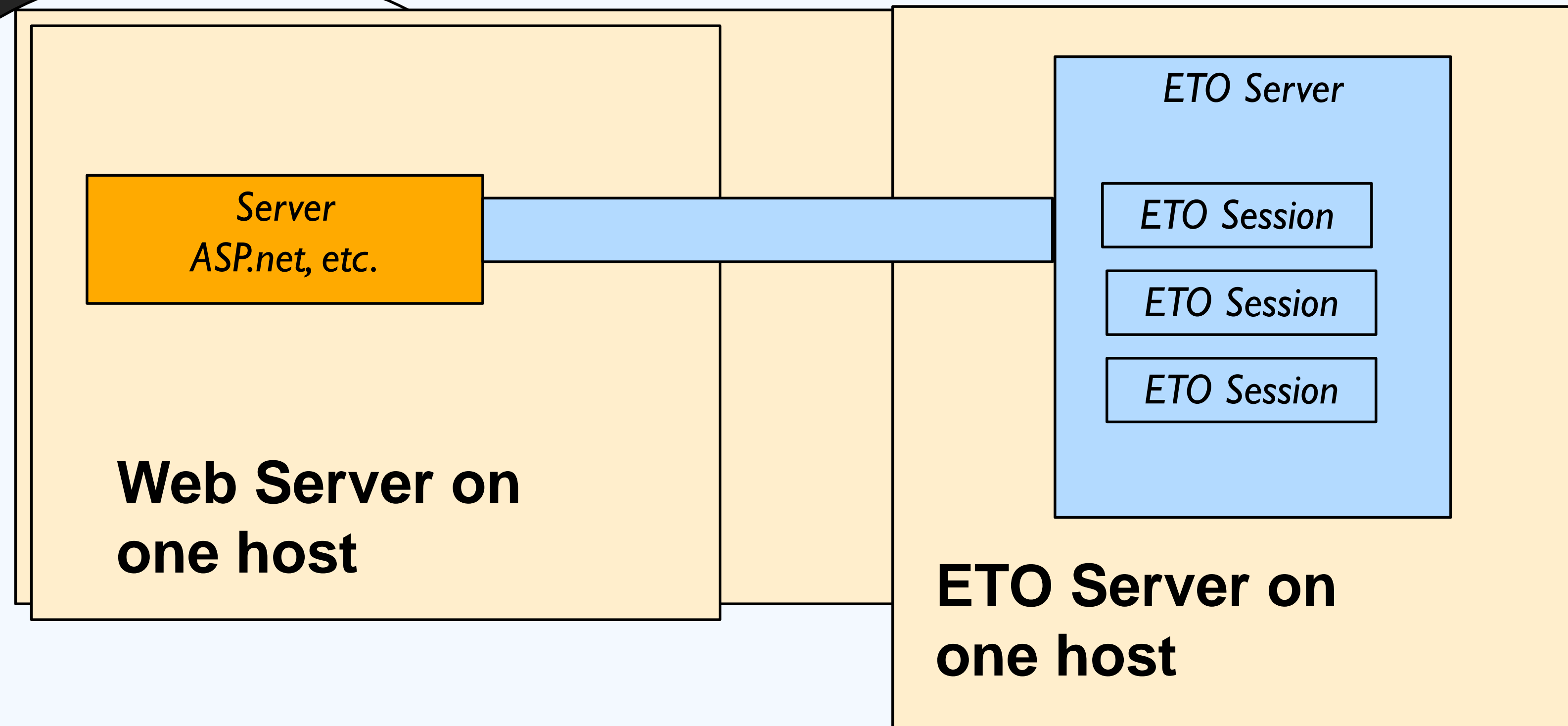
Using ETO Server Tools

Custom UI

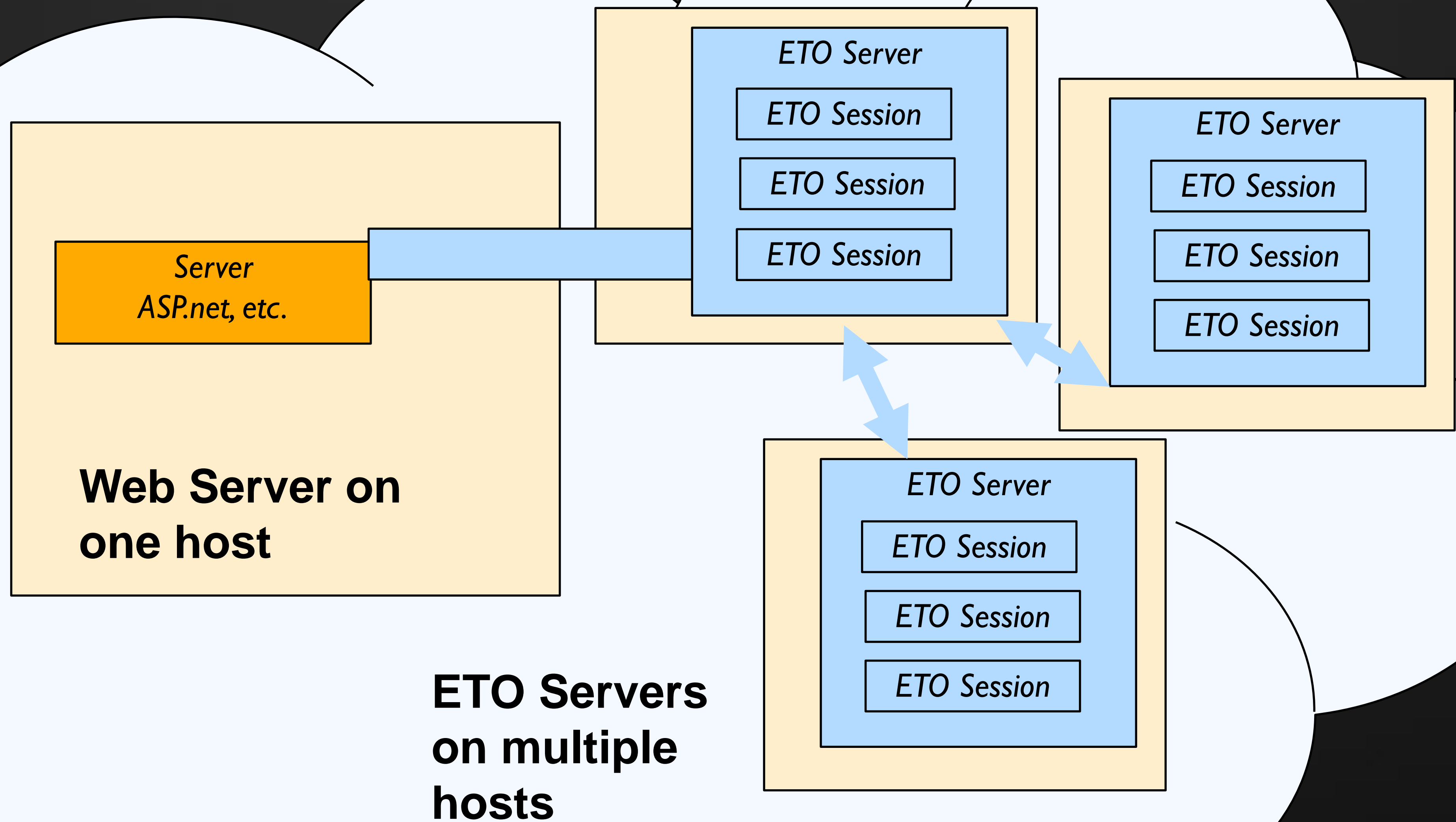


These are required, but not part of ETO

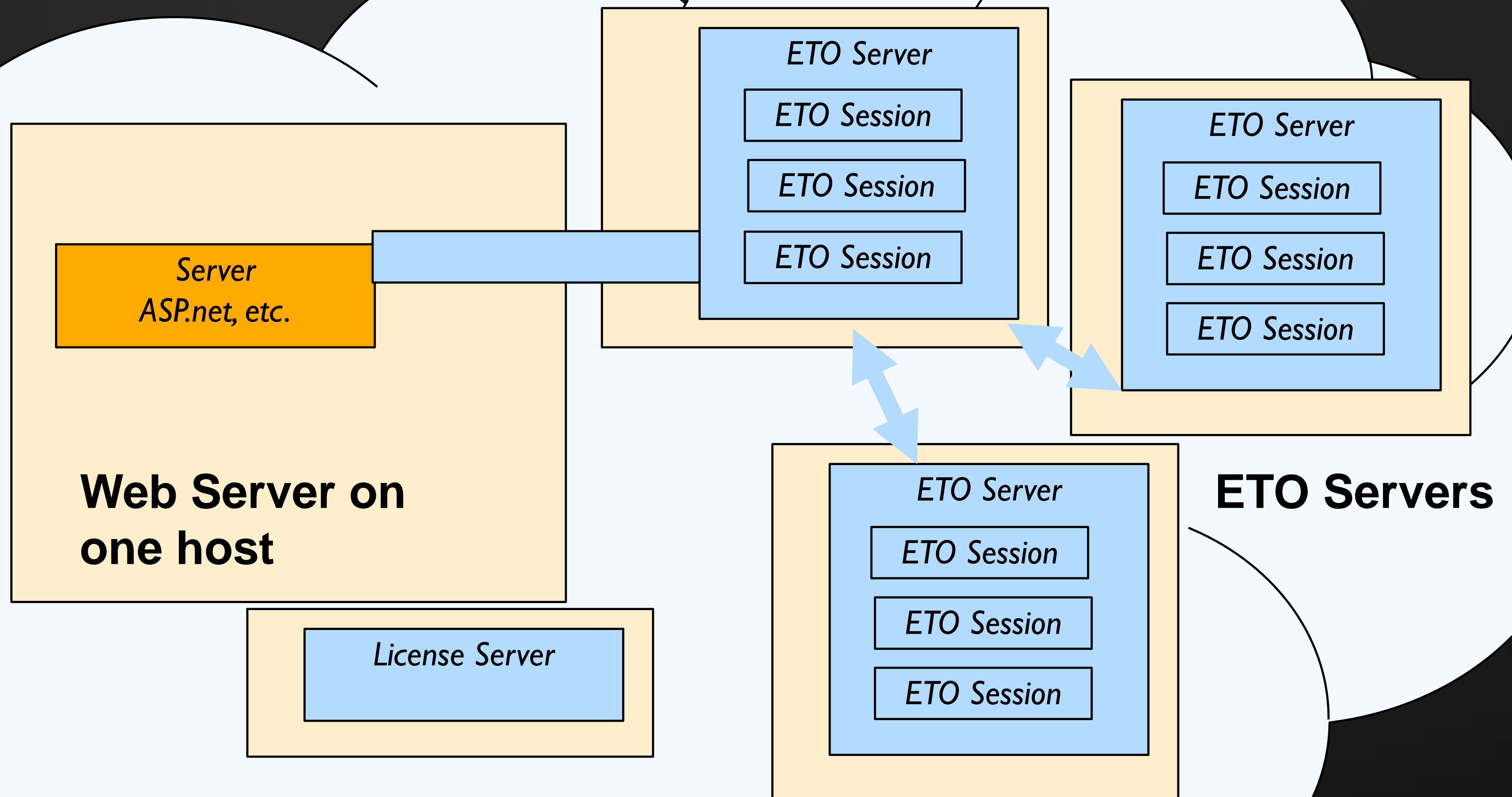
Hosting Options



Hosting Options



Licensing



Licensing

- Always network licensed
- Always “per session”
- 10 simultaneous sessions per license-unit
- Applies to all ETO Server deployment modes
- License server cannot run on same host as ETO server

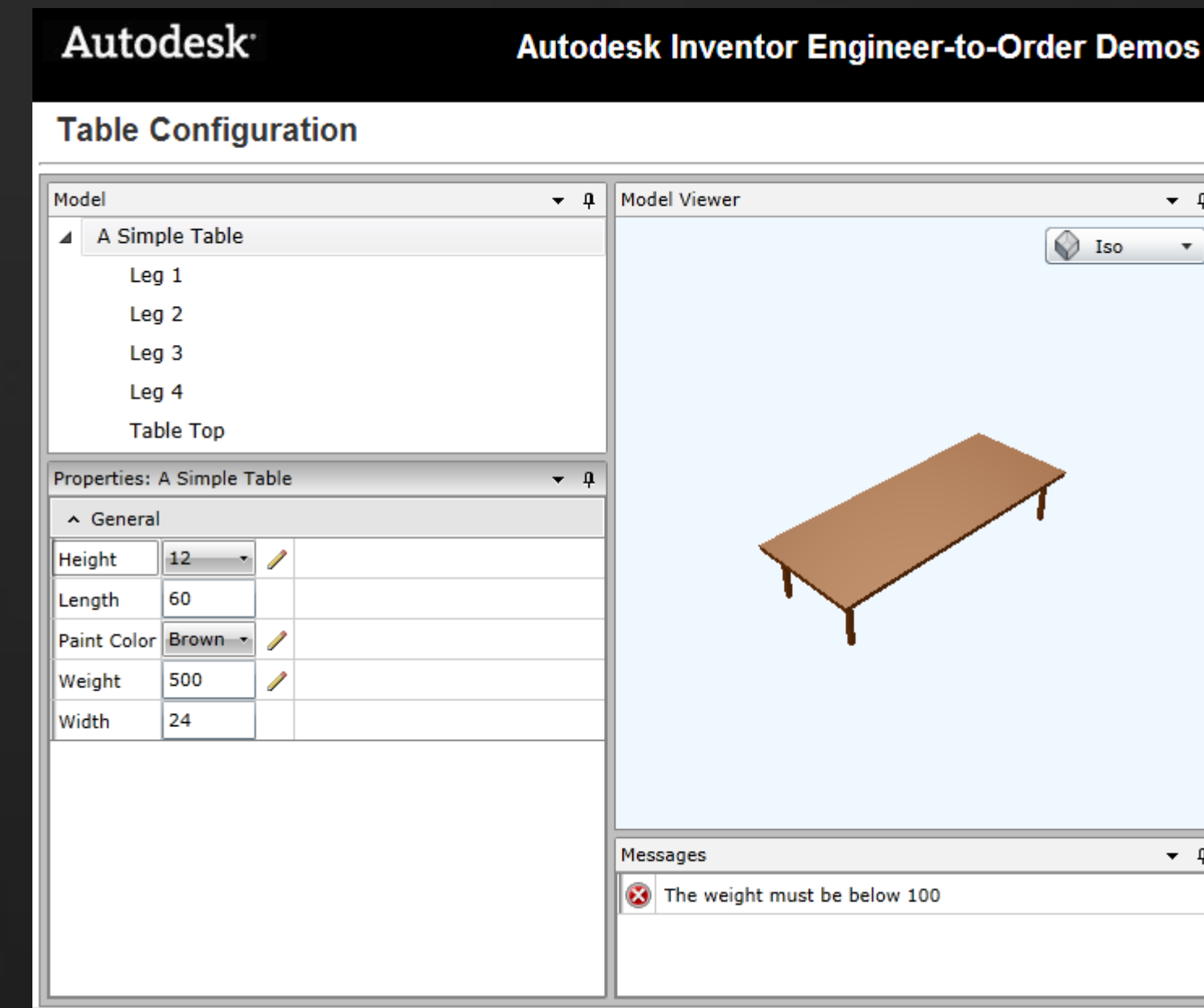
Summary

- Most flexibility
- Most non-ETO effort required
- Viewer limitations

Mode 3: Automatic UI, Interactive

Applicability:

- Real-time visualization based on user input
- Validation of every user input
- Interactive application



- ETO-provided front end interface; no web UI development required
- ETO-provided viewer; works on all browsers
- Lightweight graphics technology for web viewing
- Automatically drive web and Inventor UI

Modeling for Faster Web Performance

- Host does not run Inventor – only ETO/Intent
- Uses pre-generated “asset” files for geometry
 - Can use “fixed” IPT files in Inventor for testing
- Uses “frame-based positioning” (FBP)

Automatic UI from “UI Rules”

- UI Rules define:
 - What the user can enter
 - Input format
 - Min/Max values or other validation
 - Contents of model browser
 - Messages to user
 - How to handle illegal states

Other rules

- Support custom “action” DLLs

A

Assembly

PRO Assemble Design Model Inspect

Show UI

Whitepaper UI

Intent Model

Options...

- Root <Table>
- HeightMessage <UIMess
- WeightMessage <UIMes
- TableTop <TableTop>
- Leg_1 <Leg>
- Leg_2 <Leg>
- Leg_3 <Leg>
- Leg_4 <Leg>

Whitepaper UI

- A Simple Table
- Table Top
- Leg 1
- Leg 2
- Leg 3
- Leg 4

Properties

Rule / Value Owner

- Inventor
- Frame
- Appearance

Properties Immediate

For Help, press F1

5 3

http://localhost/Engenious.UI.Whitepaper.Silverlight.Web/Engenious.I

Engenious.UI.Whitepa...

A Simple Table

General


Height	24	
Length	60	
Paint Color	Brown	
Weight	150	
Width	24	

Table Top

Leg 1


Leg 2

Leg 3

Leg 4

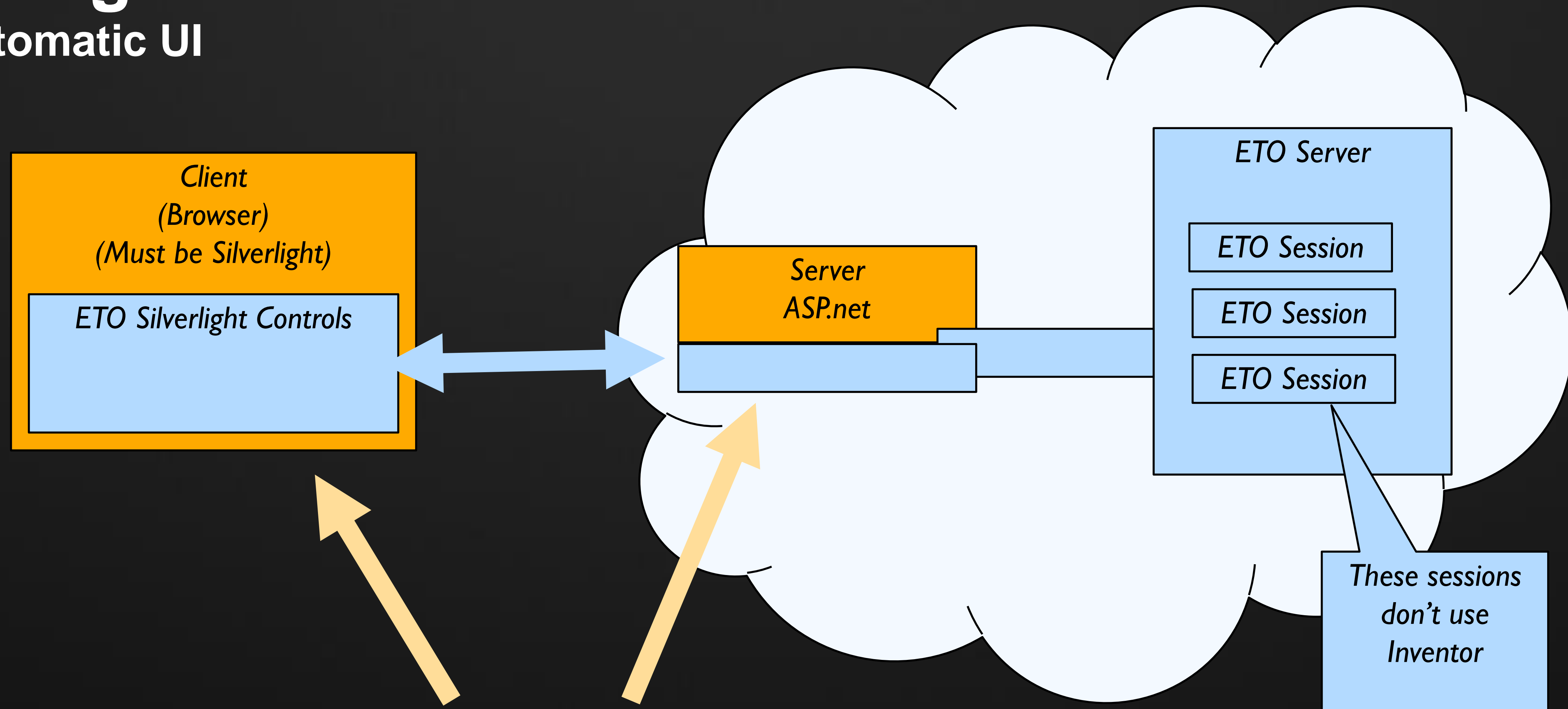
Table Top

The weight must be below 100



Using ETO Server Tools

Automatic UI



These are required, but not part of ETO
But ETO provides most of the
implementation in this case.

Summary

- Good combination of ease-of-implementation vs. functionality
- Advantages:
 - Easy to go from desktop to web
 - Easy to change end-user UI (within limits)
 - Relatively high performance
 - Partially due to FBP
- Disadvantages:
 - Must use pre-generated geometry (assets)
 - Can't use Inventor constraints (must use FBP)

Overall Summary

- ETO is for customers who have
 - Bid- and Order-Processes Challenges
 - Complex, configurable products
- ETO uses Rules to capture design knowledge
- ETO has inherent member file management capabilities and can be easily linked to Vault
- ETO “apps” deploy the knowledge where it is needed
- 3 modes of deploying an ETO app on the web:
 - Batch processor, custom interface
 - Interactive, custom interface
 - Interactive, automatic interface, with “fast” graphics

