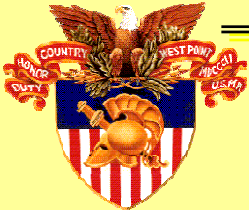
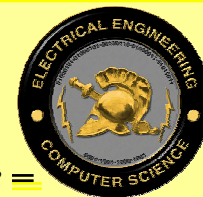


Intelligent Control Systems: Integration of Process Control and Predictive Models in a Combined GUI-Based Application

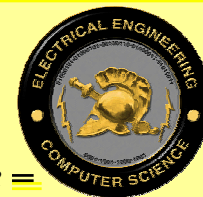
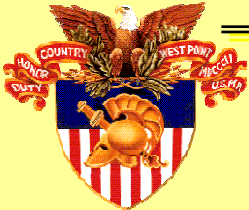
John M. D. Hill, Ph.D.
United States Military Academy
West Point, NY



Context



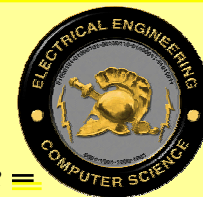
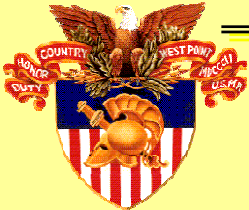
- Process Control
 - PLCs – Programmable Logic Controllers
 - SCADA – Supervisory Control and Data Acquisition
 - “Tag Points” – value reading and control setting
- Specific example:
 - West Point co-generation power plant
 - Intelligent Power Plant project
- Some software mechanisms
 - OPC Server – OLE for Process Control
 - OLE – Object Linking and Embedding
 - COM – Component Object Model
 - DCOM – Distributed COM
 - XML – eXtensible Markup Language



Some Issues and Opportunities

Electrical Engineering and Computer Science

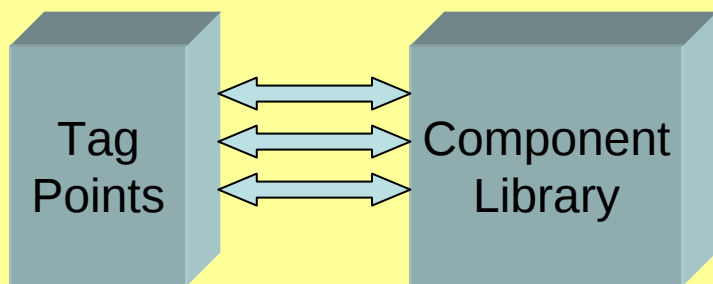
- Existing Interfaces often:
 - are locked-in and inflexible for the operators
 - don't provide dynamic facilities to support modeling, optimization, and control
- Sensor suites – thinking ahead
 - Beyond monitoring - need the right data points for modeling, prediction, optimization, and control
 - May require installation of additional sensors
- Opportunity to create a useful and flexible GUI
 - Provide visualization of what the operators want to see
 - Provide simple mechanisms for what the operators want to do
 - Make it easier for the modelers to build predictors and optimizers
 - Determine how to make it extensible to any process

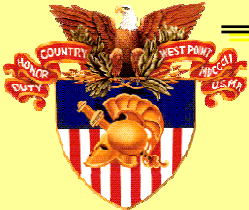


Discovery and Organization

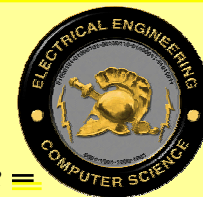
Electrical Engineering and Computer Science

- **Discovery**
 - Need a mechanism for exploring the exposed data points
 - Need input from the process engineer
- **Organization**
 - Re-name the data points without losing access to them
 - Hierarchical organization in agreement with intuition
 - Create read and write ability for each tag point
 - Consolidate all of this in a component library

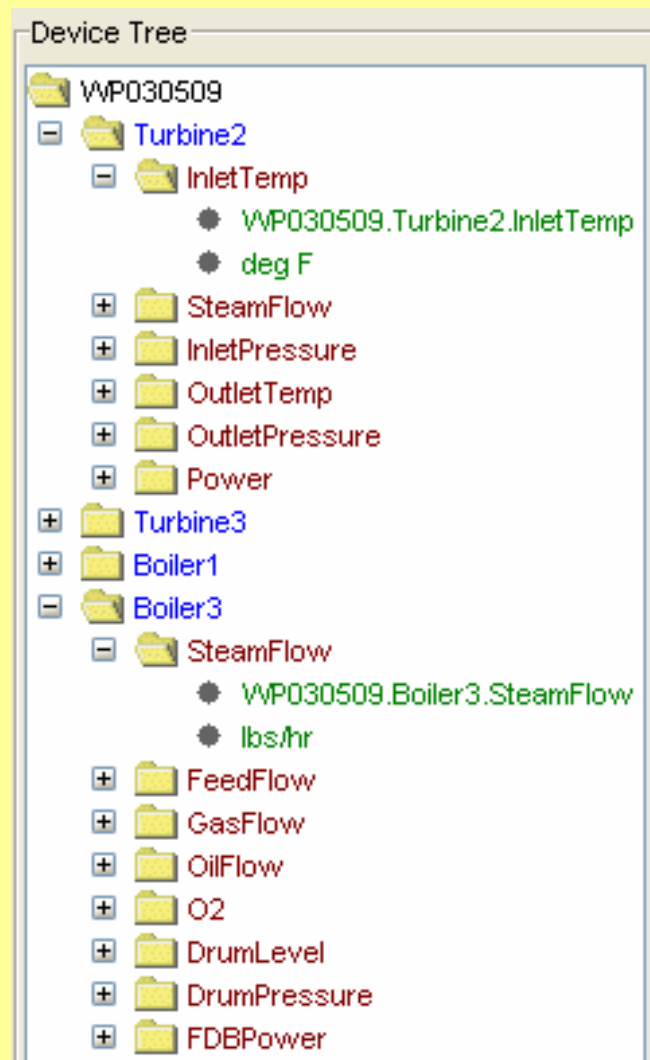


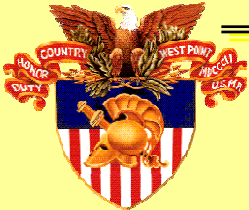


Discovery and Organization



- Model-Based Controller provides this mechanism
 - Component library available for access by other programs
 - Component library also saved in xml
 - *More on this from Nat Frampton*
- What can we do with it?
 - Use it as the sole mechanism for interactions with the process
 - Create a visualization of the component library, such as the hierarchical tree shown here.

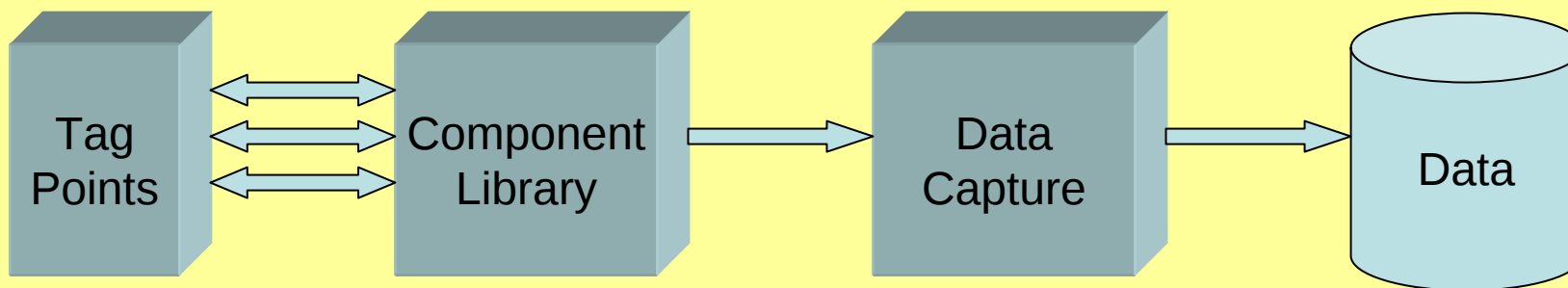


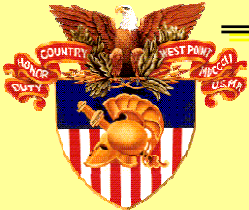


Data Capture

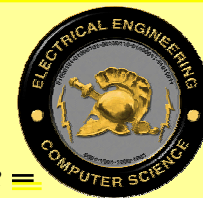
- Data Capture Module

- Select single tag points or entire hierarchies to capture
 - The operator knows the data that is desired
 - Provide a standard mechanisms that computer users understand
 - Double-click, appear in list
 - Drag from tree, drop into list
- Select time interval, and record timestamps
- Save recorded data to the database
- Can use *any* component library – don't have to create a new data capture mechanism for each process

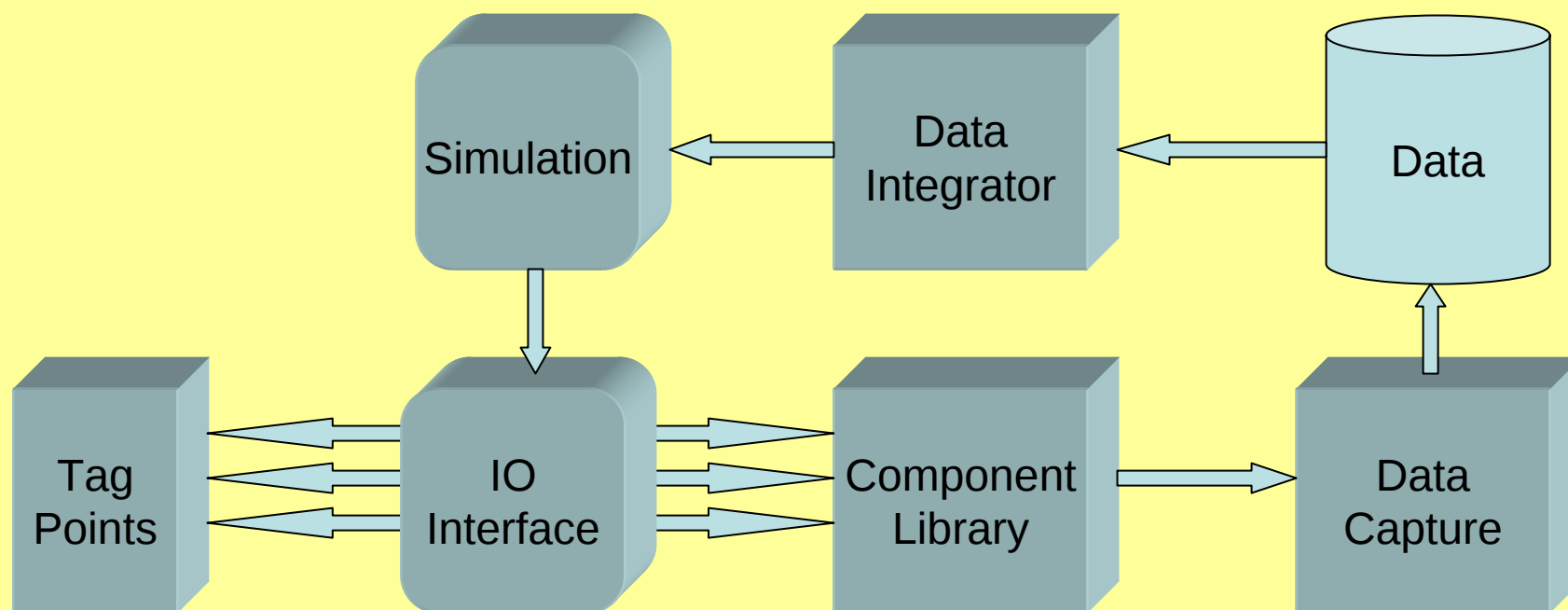


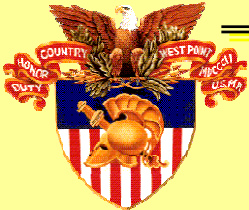


Simulation

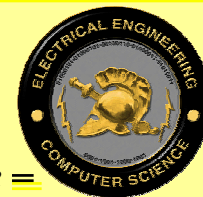


- Simple mechanism for incorporating simulation
 - Insert an IO Interface module before the Component library
 - Switch it to read data directly, or from the simulation
 - Simulation pulls the data through a Data Integrator
 - Data source is transparent to users of the Component Library





Simulation - Image



- Select and load the data files of interest, then run the simulation

MBCDCIOServer - Control Panel

Data Collection Files

Name	Rows
wp030509_030724_134110.csv	1217
wp030509_030724_144111.csv	1202

Data Collection Items

Name	Value
WP030509.AuxHPSteam.Pressure	156.9517669677
WP030509.AuxHPSteam.SteamFlow	3735.391357421
WP030509.AuxHPSteam.Temperature	370.7193298335
WP030509.Boiler1.DrumLevel	0.1632251739501
WP030509.Boiler1.DrumPressure	73.06980895996
WP030509.Boiler1.FDBPower	15.70931339263
WP030509.Boiler1.FeedFlow	3.05209890939295E
WP030509.Boiler1.NaturalGasFlow	14.74485397336
WP030509.Boiler1.O2	10.77109909057
WP030509.Boiler1.OilFlow	1.35135650634766E
WP030509.Boiler1.SteamFlow	0.0204842351377
WP030509.Boiler2.DrumLevel	-14.9630441665E
WP030509.Boiler2.DrumPressure	-74.3741378784
WP030509.Boiler2.FDBPower	
WP030509.Boiler2.FeedFlow	

Current Time 2/7/2005 7:22:40 PM

Now Playing

Filename: wp030509_030724_134110.csv Row: 2 of 1217 Current Timestamp (TS): 7/24/2003 1:41:17 PM

Begin Timestamp	End Timestamp	TS Span	Remaining TS	Next TS
7/24/2003 1:41:14 PM	7/24/2003 2:42:05 PM	00:01:00:51	00:01:00:48	00:00:02

Est. Time Remaining: 00:01:00:47

Buttons: Load Unload Run Step Stop Reset

Entire Run

File: 1 of 2 Time Started: 2/7/2005 7:22:36 PM

Total Rows: 2 of 2419 Est. Ending Time: 2/7/2005 7:22:36 PM

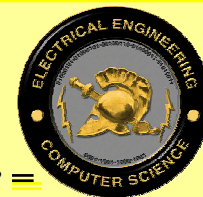
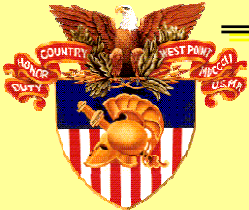
Elapsed Time: 00:00:04 Est. Time Remaining: 01:23:53:56

Server Info

State	Mode
Running	Auto

Interval (secs)	Source	Rate
3	DCFile	1:1

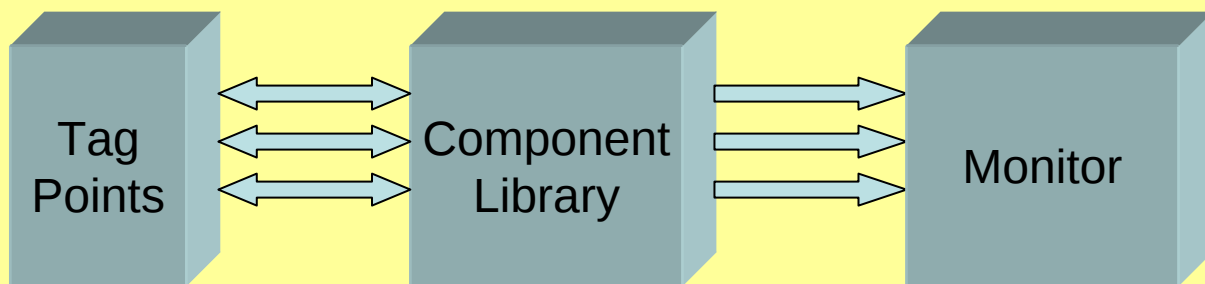
Buttons: Configure Update

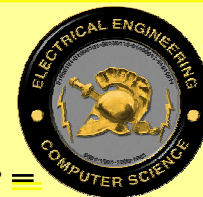
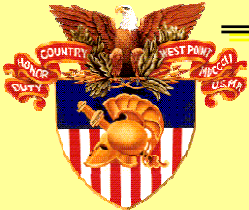


Process Monitoring

Electrical Engineering and Computer Science

- Tag Points of Interest
 - We have a visualization of the (re-named) tag points
 - Behind that, we have the means of reading the tag points
- Methods for Presenting the Readings
 - As text
 - As a graph
- Comparison of Different Tag Points
 - Trend graphs are synchronized in time
 - Snapshots of text and graph displays can be captured

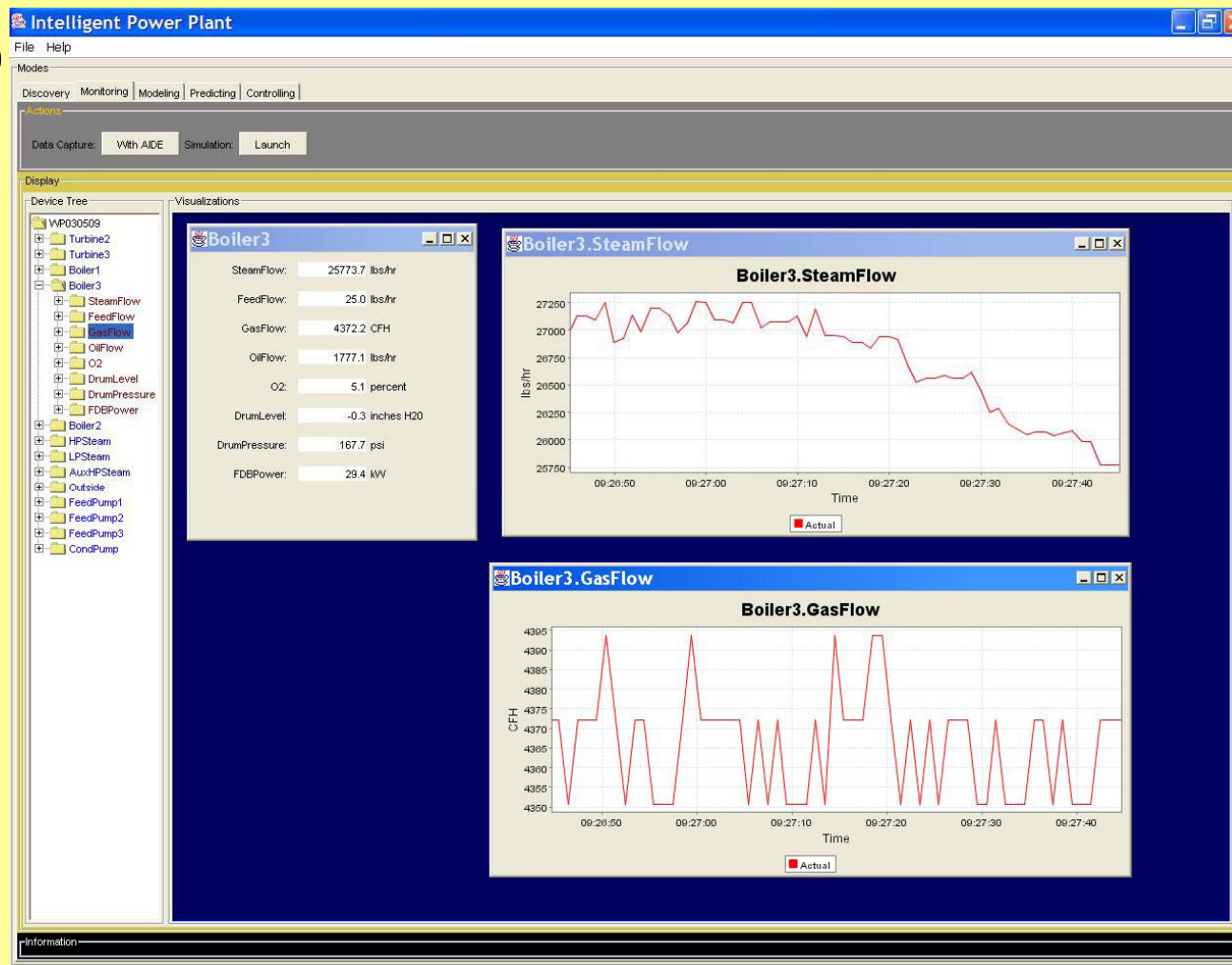


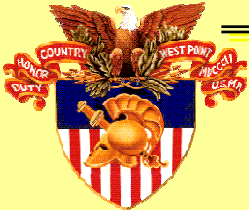


Process Monitoring - Image

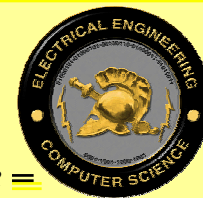
Electrical Engineering and Computer Science

- Example
 - Drag and drop from the hierarchical tree
- Features
 - Source independence
 - Flexibility
 - Live or replayed data

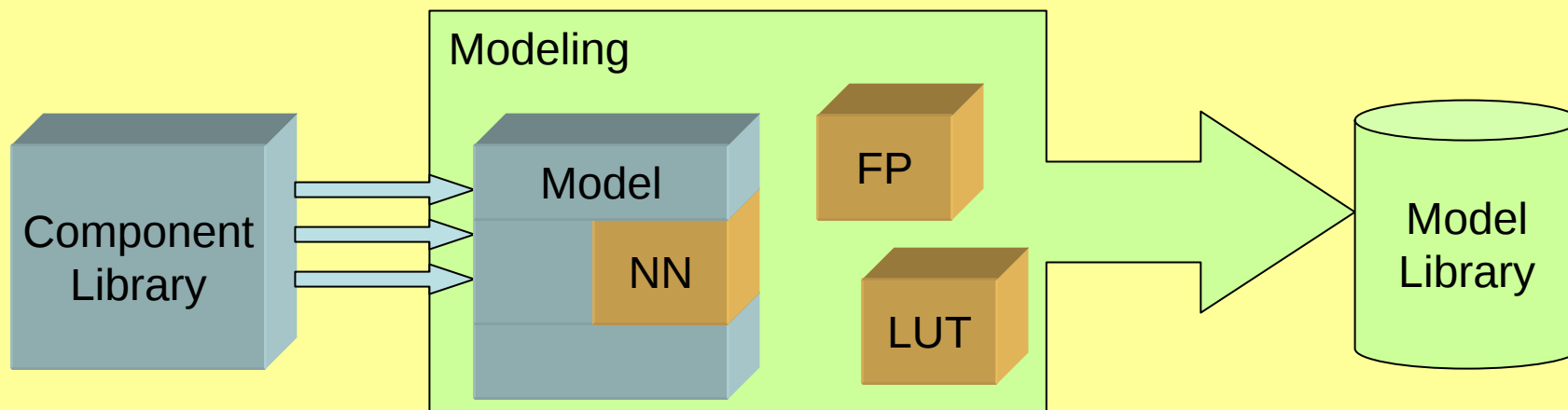


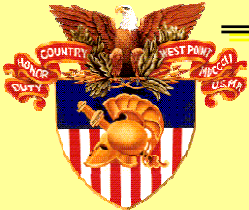


Modeling

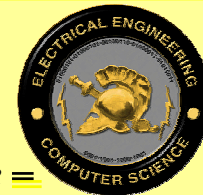


- Models based on tag points of interest
 - Select tag points from the component library
 - Remember that the data might be live or simulated
 - Modeling may be counter-intuitive (what follows what?)
- Different model types to “plug in”
 - First principles, look up tables, neural networks, etc.
 - More on model creation and neural networks from Peter Curtiss

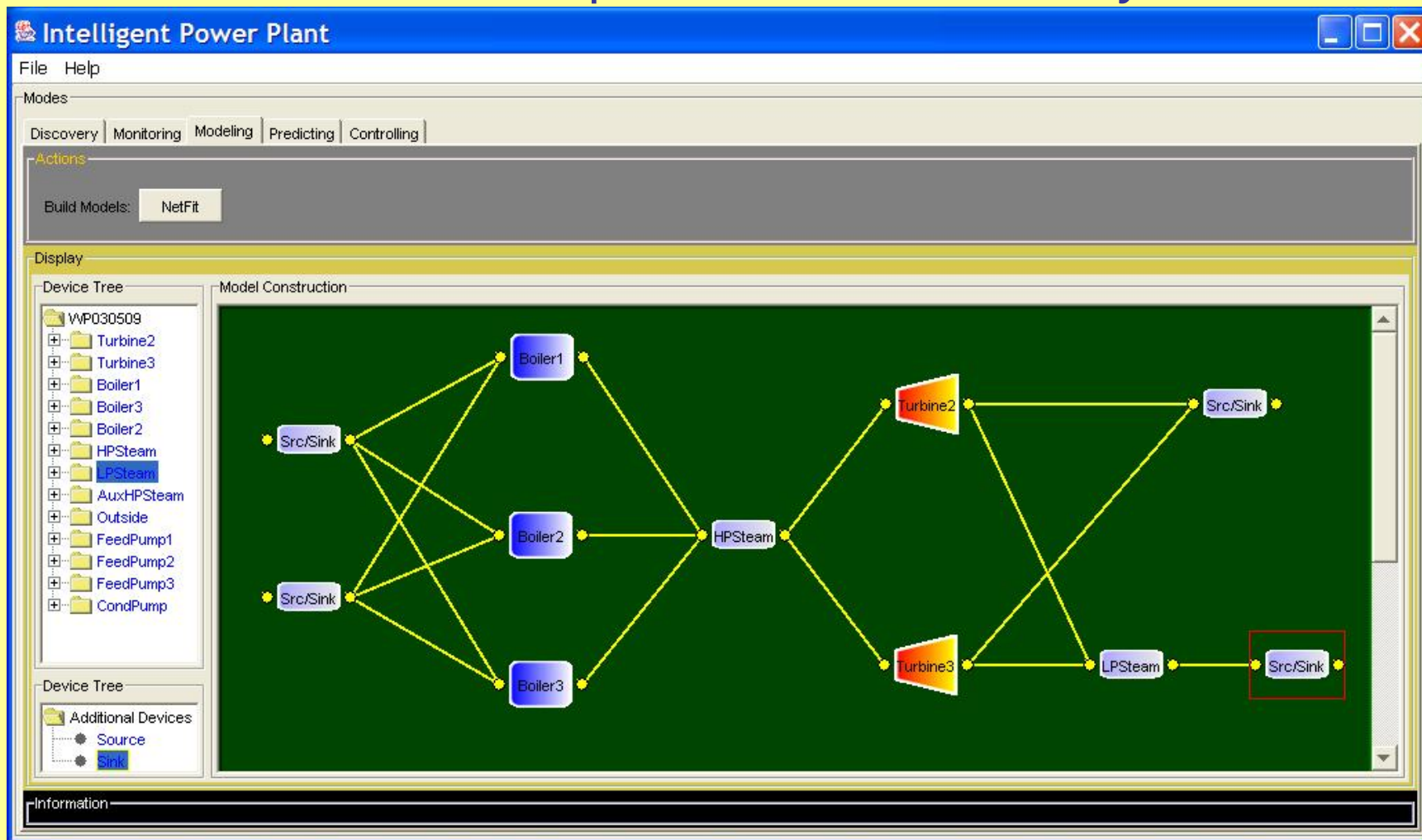


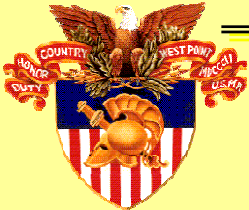


Component Modeling

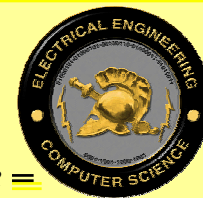


- Model tied to the components from the library

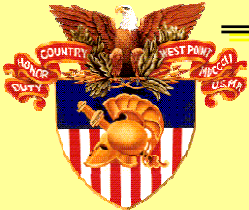




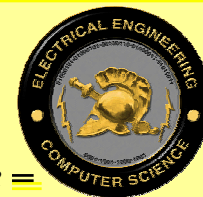
Process Modeling



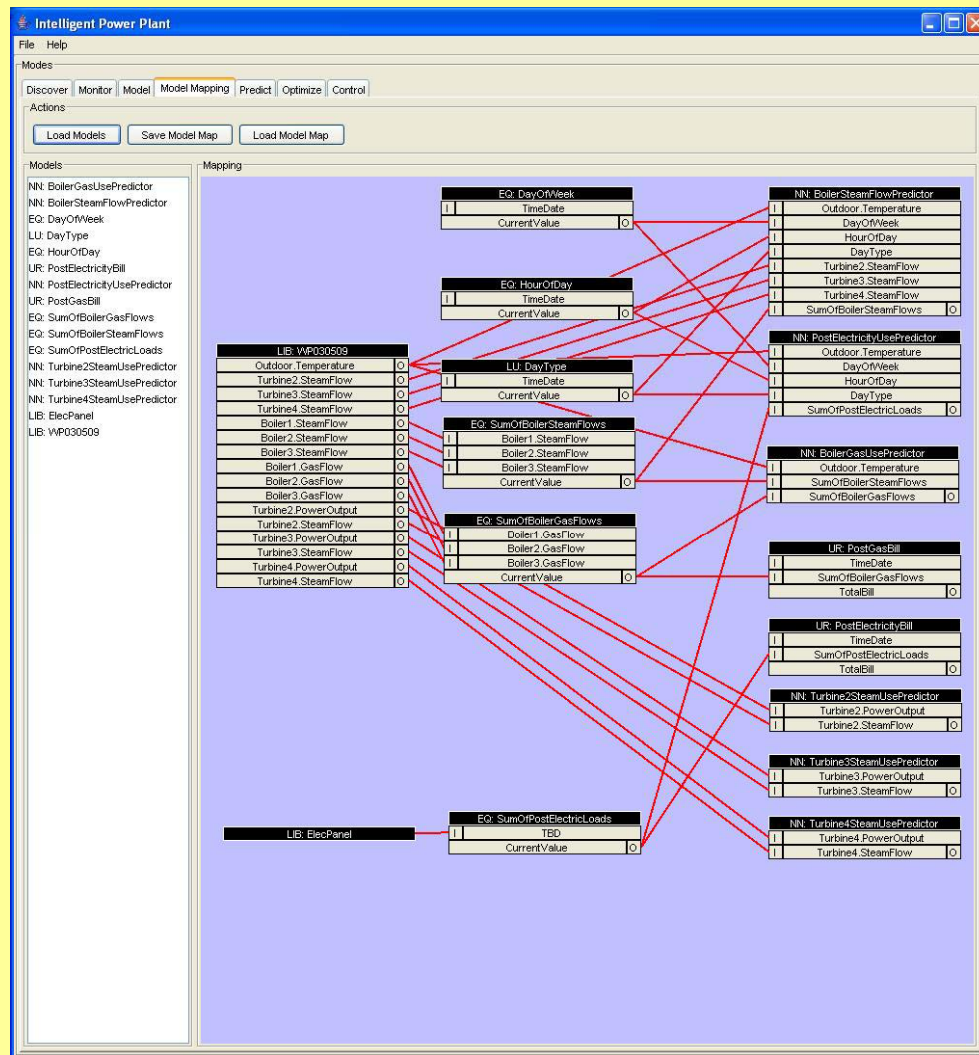
- Extended Modeling
 - Lots of things that affect the model (and predictions based on it) may not be included in the tag-points.
 - Examples from the USMA Power Plant:
 - Student attendance
 - Weather (trend, not instantaneous)
 - Others already mentioned by Darrell Massie
 - Might need to get these from external sources
 - Security issues! More on this from Scott Lathrop
- More sophisticated modeling
 - Decouple the models from the components
 - The output of some models may be the inputs of other models
 - Need mechanism for composition of the meta-models

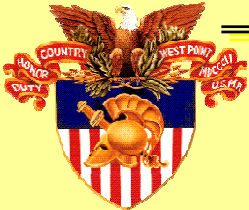


Model Visualization and Mapping



- Visualizing the models
 - Name, Inputs, Outputs
- Making the connection
 - List the models from the model library
 - Drag them onto the mapping field
 - Drag the output from one model to the input of another
 - Collapse/expand models for clarity
- Works for any model

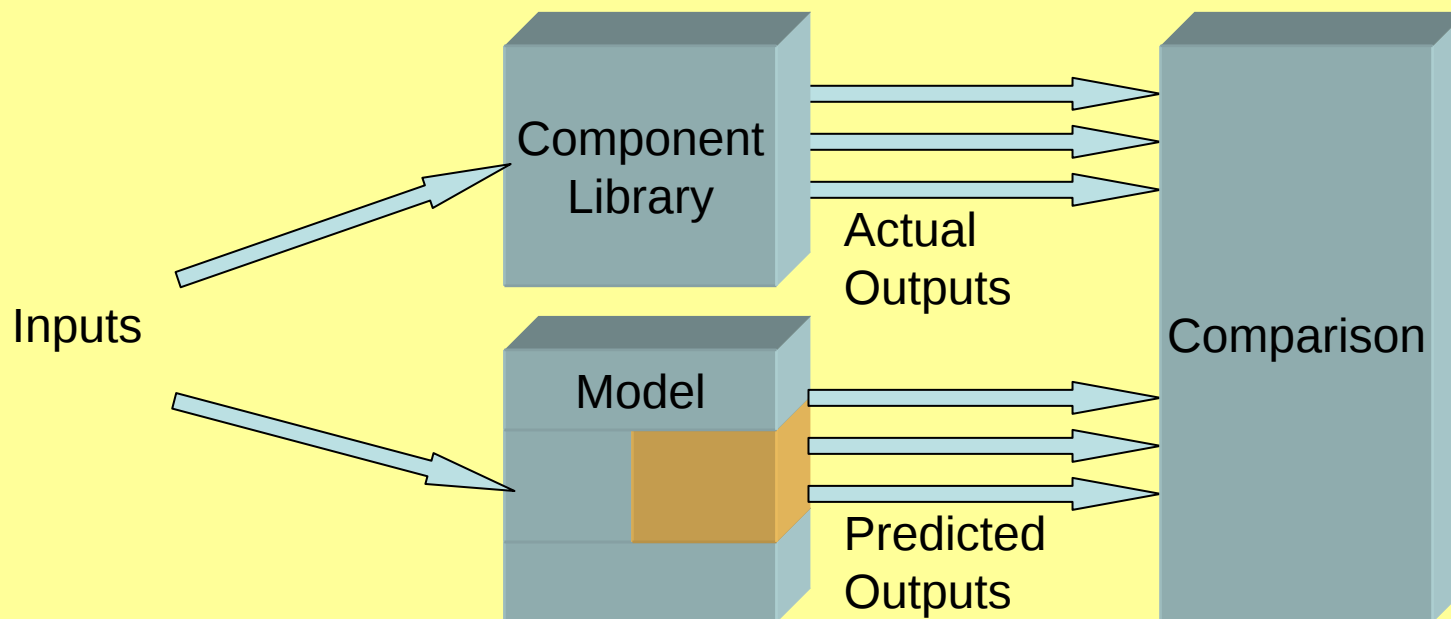


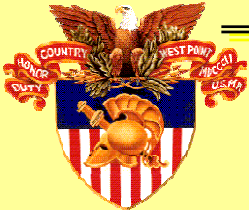


Modeling Validation and Verification

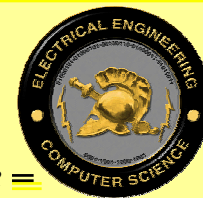
Electrical Engineering and Computer Science

- Validation
 - modeling the correct objects (input from the process engineer)
- Verification
 - modeling the objects correctly
- Statistical measures of accuracy



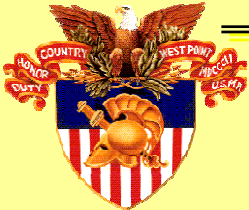


Prediction - Image



- Correct models
 - can be used for making predictions
- Visualization
 - of predictions by themselves
 - of predictions versus actual data
 - helps people see the accuracy (or inaccuracy)

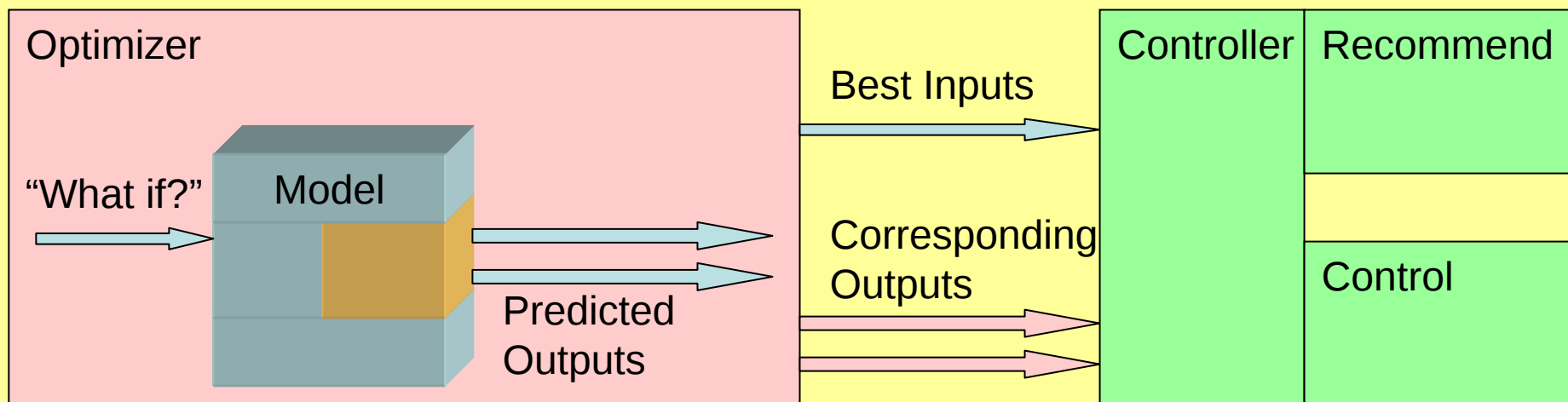


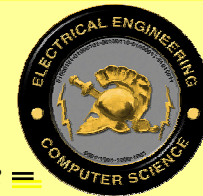
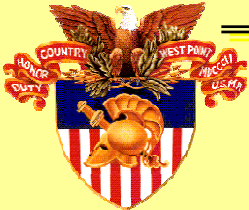


Optimization and Control

Electrical Engineering and Computer Science

- **Optimizer**
 - Feeds “what if?” inputs to the model(s)
 - Records the best combination
 - Sends the best combination to the controller
- **Controller**
 - Can recommend control settings to the operator
 - Can control the process directly (through the component library)

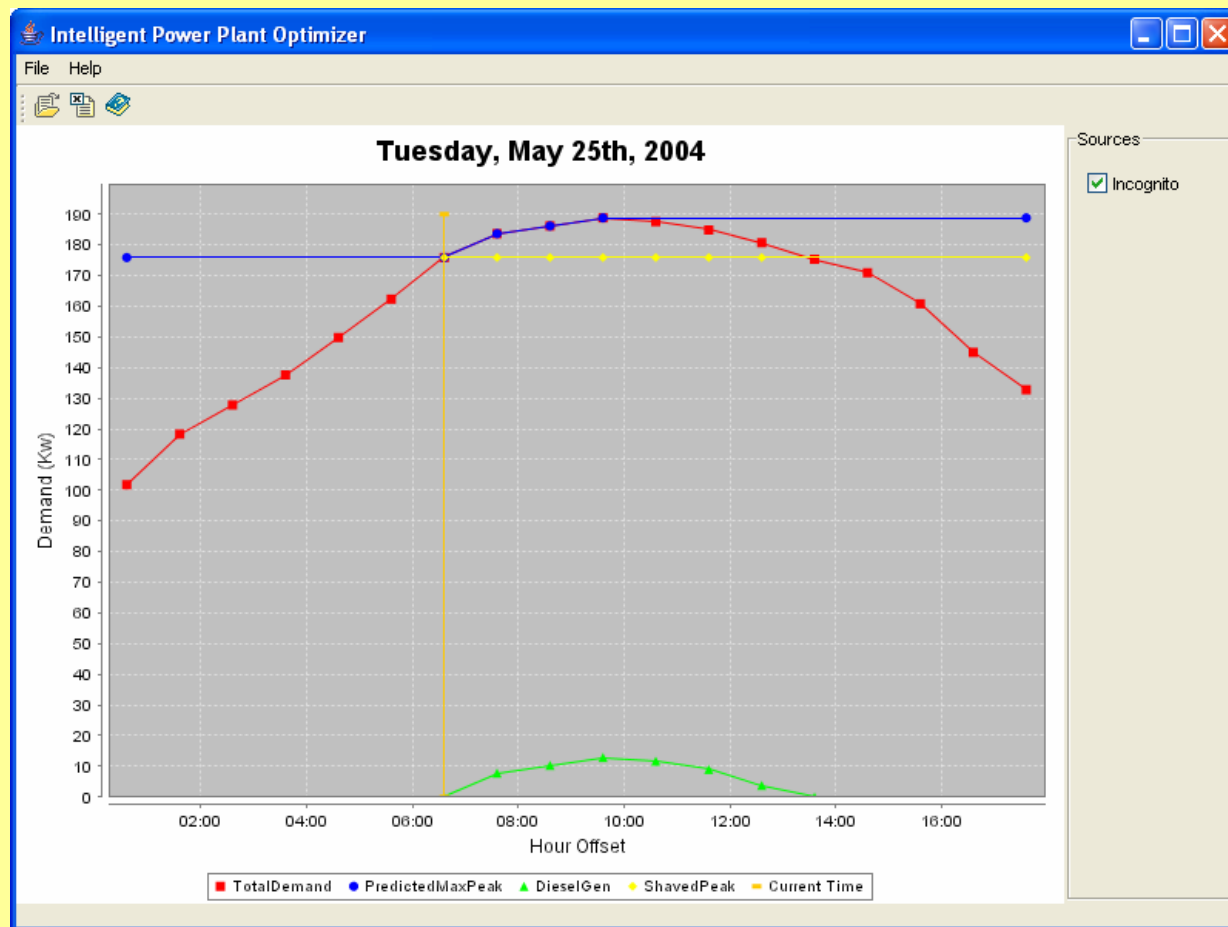


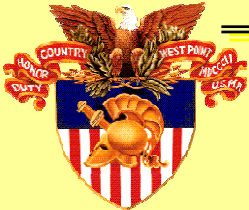


Optimization and Control - Image

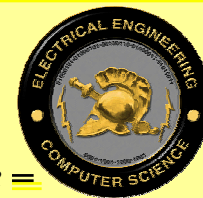
Electrical Engineering and Computer Science

- Visualization
 - helps the operator believe the recommendation





Conclusions



- The ability to discover, re-name, and organize tag points into an easily understood component library serves as the foundation for the other operations described here.
- Basing the other operations on the component library significantly reduces the amount of custom programming required.
- Graphical interfaces and visualizations:
 - make data capture, monitoring, etc., accessible to the operator.
 - make model creation and mapping easier for the modeler.