Understanding Complex Manufacturing Equipment Structure and Behavior through Advanced Visualization Technologies

Process Book

Alan Weber Faris Khan Filemon Mateus

Major Section 1 – Project Proposal

1 Basic Information

The project title, your names, e-mail addresses, UIDs, a link to the project repository.

Title: Understanding Complex Manufacturing Equipment Structure and Behavior through Advanced Visualization Technologies

Project team: Alan Weber, u1267778@utah.edu, u1267778
Faris Khan, u1164527@utah.edu, u1164527
Filemon Mateus, u1419667@utah.edu, u1419667

Repository link: https://github.com/faris-k/dataviscourse-pr-manufacturing-vis

Note to TAs: please accept the invitation issued to you for the github repository for this project, as it will remain private for the time being as we determine the best way to protect any IP that may exist in the repository as we get started. It will be made public before completion. We've invited "datavis-ta" as a collaborator to the repo.

2 Background and Motivation

Discuss your motivations and reasons for choosing this project, especially any background or research interests that may have influenced your decision.

The manufacturing equipment in semiconductor factories—both frontend wafer fabs and backend assembly, packaging, and test facilities—are arguably the richest sources of data about how well the production processes are being run. This data is used in the calculation of every KPI (Key Performance Indicator) used by a wide variety of stakeholders to manage day-to-day operations, from yield to cycle time to equipment and factory productivity to… the list goes on.

However, the lack of detailed visibility into equipment and process behavior is still cited as a limiting factor for effective smart factory management, despite the progress that has

been made over the past 3 decades in defining industry standards for communications and control of this equipment. As process and product complexity continue to increase, the need for more and better information in the industry's factory information and control systems is apparent.

The solution to this problem is not simply additional standards, but also includes deeper understanding of the data conveyed by the current implementations of available standards. Advanced visualization technologies can provide fresh insights into complex equipment structure by rendering their standard representations in different ways depending on a particular stakeholder's background and needs. These insights can in turn lead to more robust data collection schemes that enhance the equipment characterization and "discovery" processes. These in turn lead to improved performance of every manufacturing job function, application, and metric that depends on good equipment data.

3 Project Objectives

Provide the primary questions you are trying to answer with your visualization. What would you like to learn and accomplish? List the benefits.

Primary questions include:

- What physical and logical nodes comprise this equipment?
- How do they relate to one another?
- What process variables, control mechanisms, equipment constants, and other parameters are associated with each node? What are the signal characteristics of these parameters, and how must they be collected as a result?
- What events and alarms are available that chronicle its behavior—normal and otherwise?
- How is the performance of this equipment measured?
- How can all the above be visualized in ways that are meaningful and intuitive to the various stakeholder groups (process engineers, equipment engineers, maintenance engineers, field service technicians, industrial engineers, production planners, automation engineers, manufacturing application software developers, etc.)?

Benefits of answering these questions include:

- Faster equipment acceptance ($$)
- Steeper production yield ramp ($$$)

- Factory software engineering productivity ($)
- More accurate and timely calculation of manufacturing KPIs (Key Performance Indicators) ($$)
- Process engineering productivity—less time to determine what data needs to be collected to support mission-critical manufacturing applications; example includes MVA-based Fault Detection and Classification (FDC) systems ($)
- Improved manufacturing application performance ($$)

4 Data and Sources

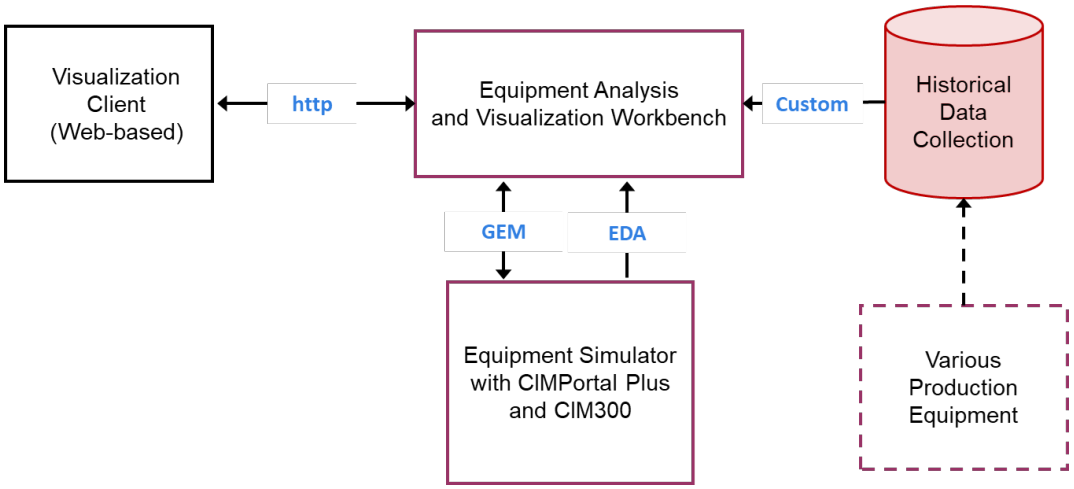
From where and how are you collecting your data? If appropriate, provide a link to your data sources.

Alan Weber has access to numerous equipment metadata models in XML format governed by two broadly adopted communications standards in the industry. These models can be “anonymized” to protect the IP of their respective sources.

Manufacturing data corresponding to these models comes from two sources:

- Actual manufacturing data, which again, can be anonymized by mapping equipment and parameter names to unrecognizable equivalents.
- Equipment simulators which can be driven by these models to generate pseudo-real-time manufacturing data with sufficient fidelity to support the characterization process.

A high-level block diagram of the system configuration used for this project appears in the figure below. See the References section for a partial list of actual software modules and data sources summarized in the figure.



5 Data Processing

Do you expect to do substantial data cleanup? What quantities do you plan to derive from your data? How will data processing be implemented?

We aim to utilize production-grade data sources which require little, if any, traditional “data cleanup.” However, we will need to anonymize some of the data, and convert their XML and JSON representations into formats that can easily be processed by our chosen visualization technologies.

As we determine what aspects of the data can be best explored with visualization technologies, we will filter out extraneous information to reduce the size and complexity of the datasets involved. For example, the standard XML format for the complete representation of the equipment structure exposed by a communications interface includes all sorts of references to supplier-specific implementation artifacts which are not relevant to the objectives of this project.

When dealing with real-time data collected from the equipment, we will need to “resample” much of the data on a consistent interval to facilitate statistical analysis of aggregated signals since the raw data may have been generated at a variety of frequencies. We will also need to adjust timestamps of the data to put them in a common “window” for ease of use since the datasets will come from multiple sources and timeframes.

Examples of signal analysis to be done in this project include

- Basic statistics for individual signals (mean, range, standard deviation, rate of change, moving average) within meaningful time windows.
- Correlation calculations for multiple signals (scatterplots, multivariate/principal component analysis for related parameters).
- Timing characteristics for individual events/alarms.
- Sequence analysis for related events (e.g., steps in equipment recipe execution, steps in automation material handling, etc.
- Identification of patterns of signals and deviations from calculated averages.

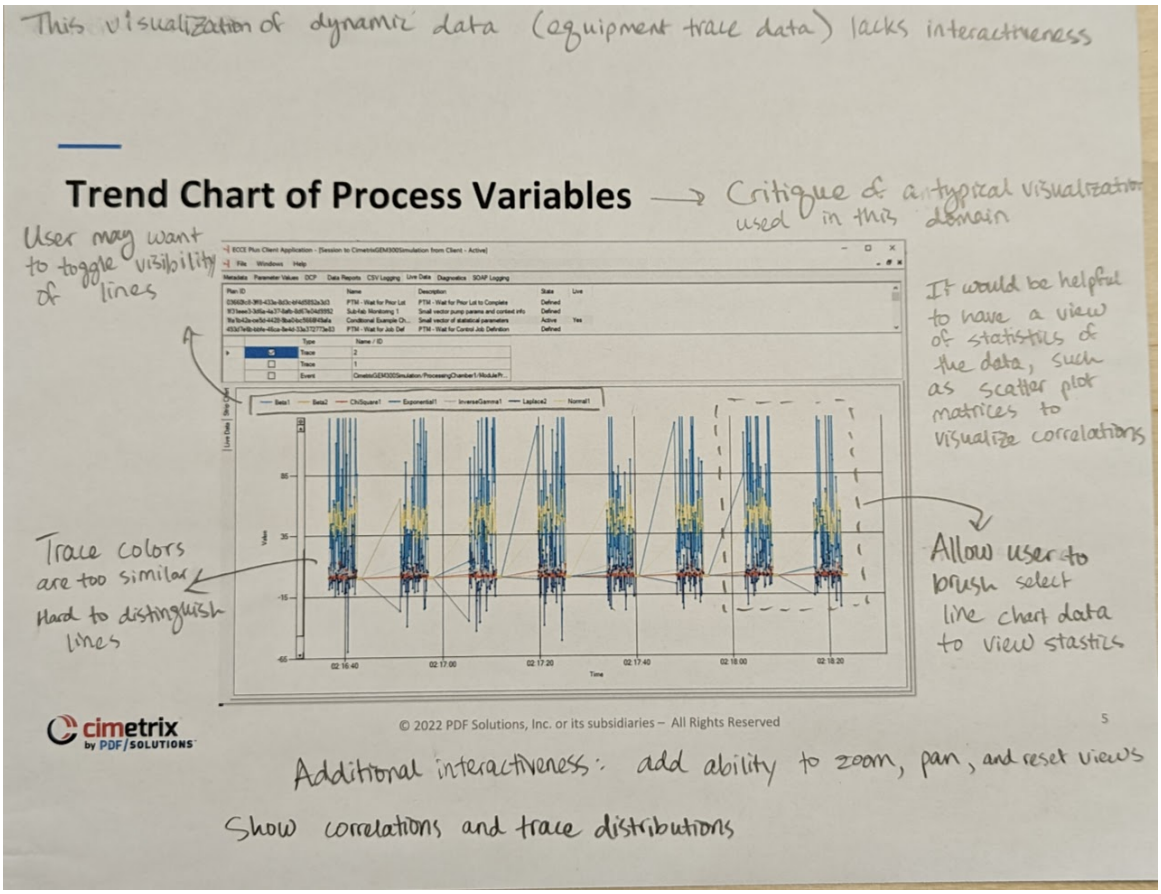
Proper categorization of signals (type, criticality, expected behavior) may require some manual annotation of the equipment model using domain experience about the specific process(es) supported by the equipment.

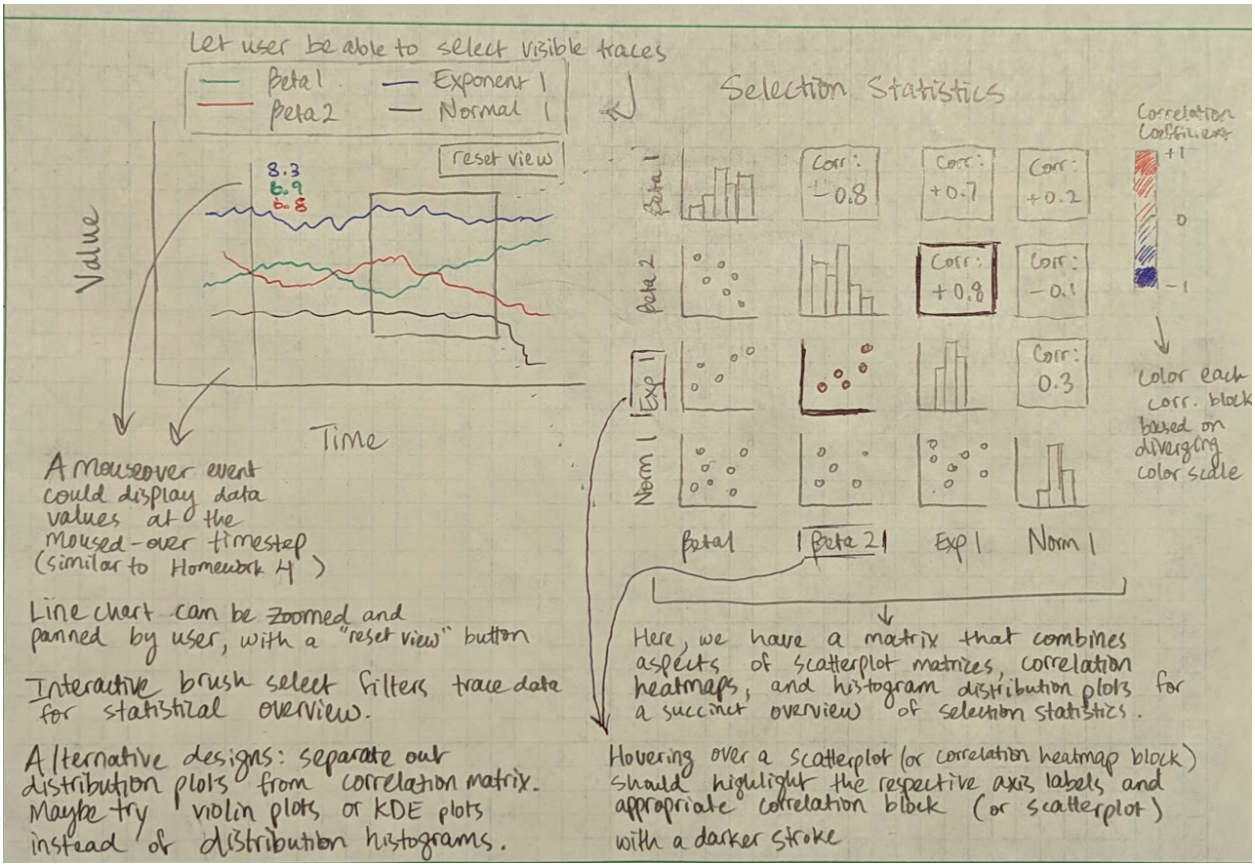
6 Visualization Design

How will you display your data? Provide some general ideas that you have for the visualization design. Develop three alternative prototype designs for your visualization. Create one final design that incorporates the best of your three designs. Describe your designs and justify your choices of visual encodings. We recommend you use the Five Design Sheet Methodology.

Initial Design of Dynamic Data

This portion of the visualization design focuses on dynamic data coming from process equipment. Time-dependent equipment data can be visualized by a line chart with multiple traces corresponding to multiple process variables. The user can interact with the line chart to brush-select data and visualize key summary statistics for the selection. Below is a critique of a typical visualization of dynamic equipment data used in industry, followed by a prototype of our visualization design for this dynamic data. Our statistics summary for the brush selection incorporates scatter plot matrices and correlation heatmaps to show correlations between process variables, and it also has histogram distribution plots along the main diagonal to visualize distributions of process variables.

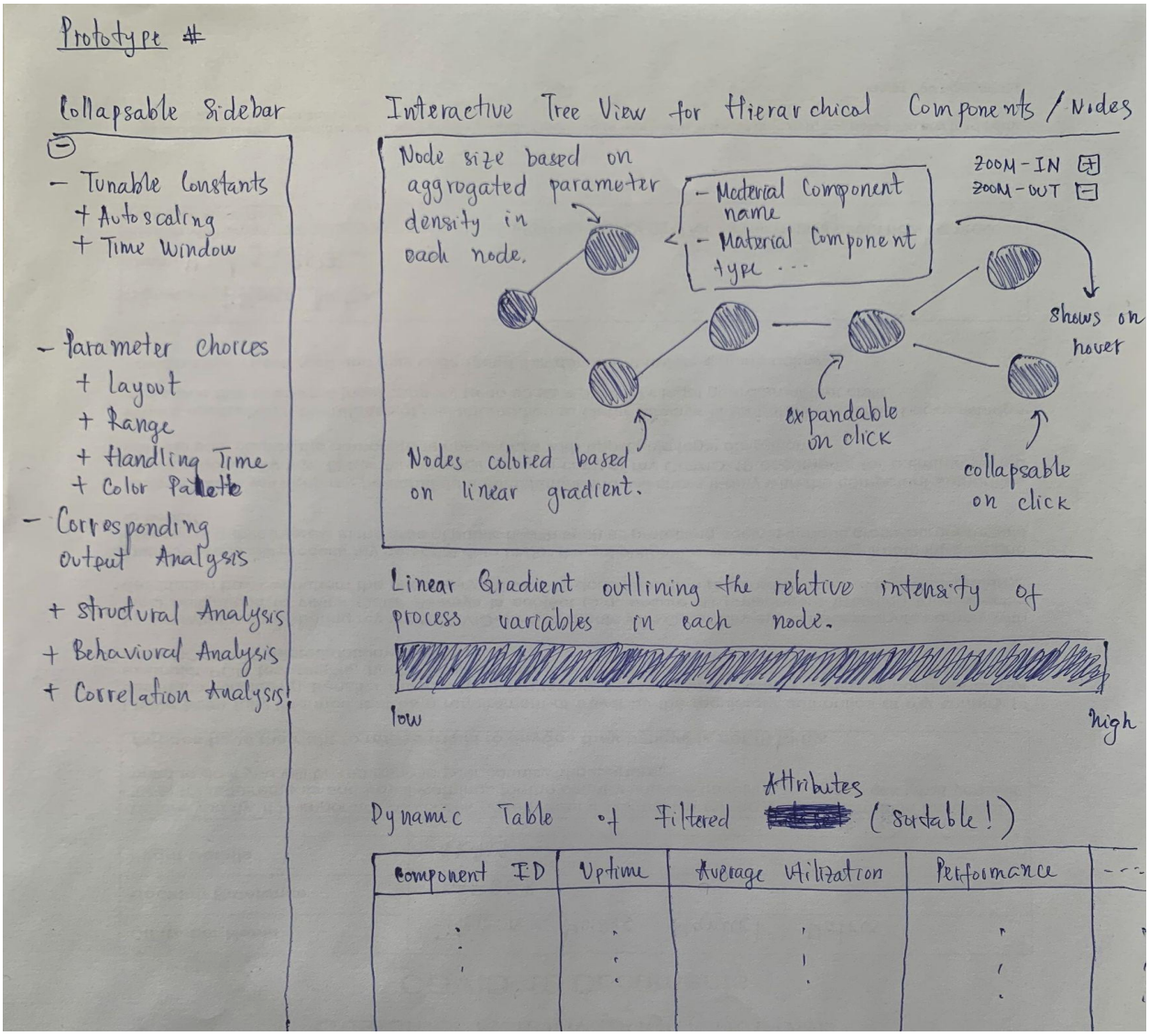




Initial Design of Static Equipment Structural Data (Tree View)

In this prototype design, the user interface is composed of four distinct, interrelated components—providing a diverse analysis and visual interactivity to better understand the dynamics between different system states and parts. Due to the complex nature of manufacturing equipment structure and behavior, we have a collapsible sidebar that allows the user to select different linear combinations of tunable constants, parameter choices, and the type of analysis to run. The intention here is to have a non-invasive component on the page with the flexibility to be hidden when inactive or unutilized by the user. Coupled with this, it is an interactive tree view that dynamically explores the hierarchical associations among component parts: modules, subsystems, sensors, actuators, logical elements, etc. Each node of the tree is colored by a linear gradient representing the relative intensity of process variables within that node, and we hope that by rendering nodes this way, we will achieve better, visually noticeable differentiations among component parts—i.e., similar component parts will map to similar intensity values in the linear gradient. Finally, we have a sortable, dynamic table of filtered attributes that displays detailed and pertinent information about every material component/node of the tree. Selecting a row of the table dynamically reconfigures the tree in such a way that the

region of interest displayed is centered around the selected node, and a tooltip displaying the material component name, type, and icon is shown.

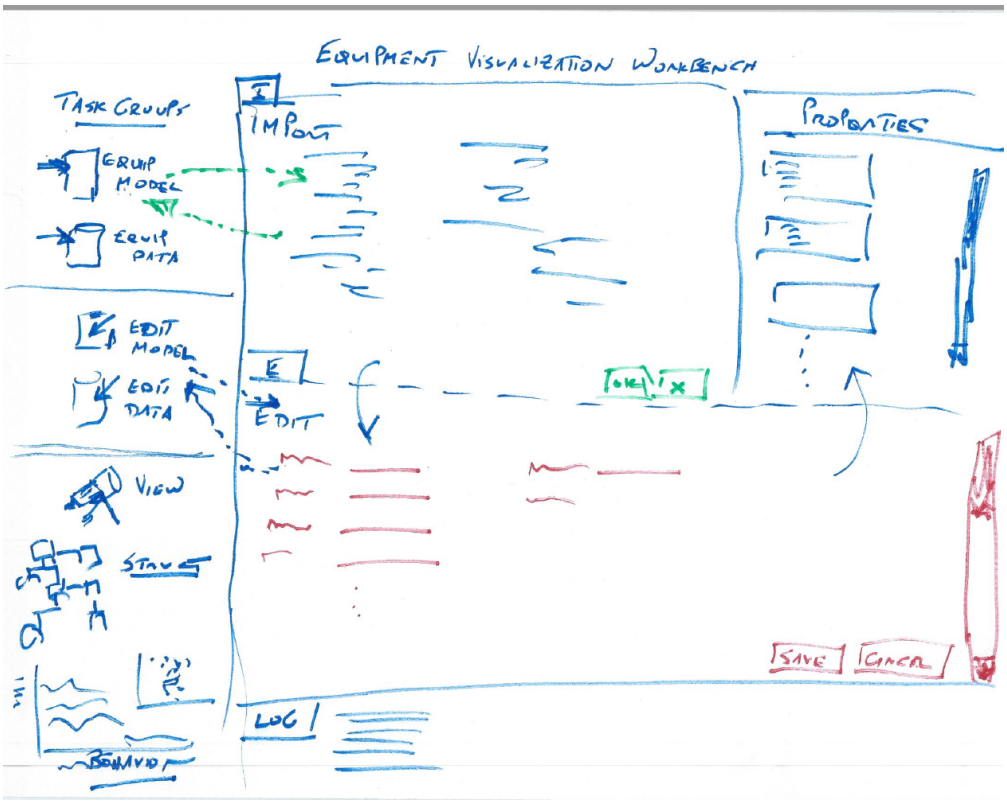


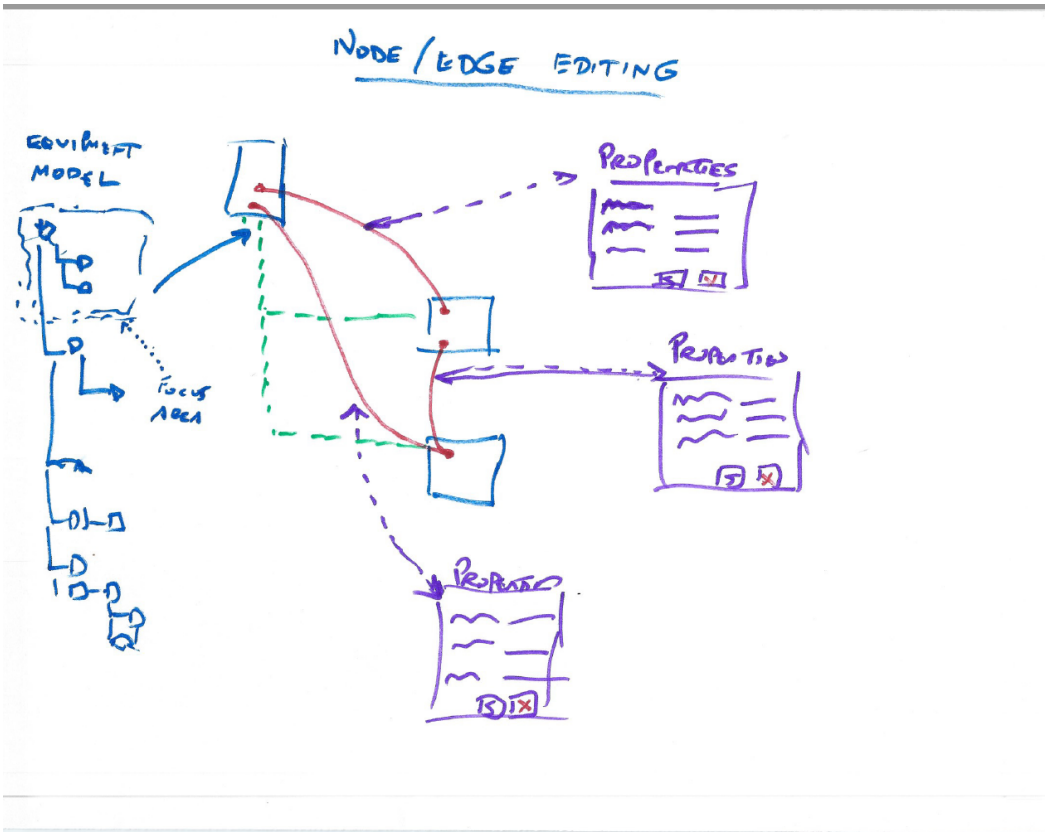
Final Design Sketches

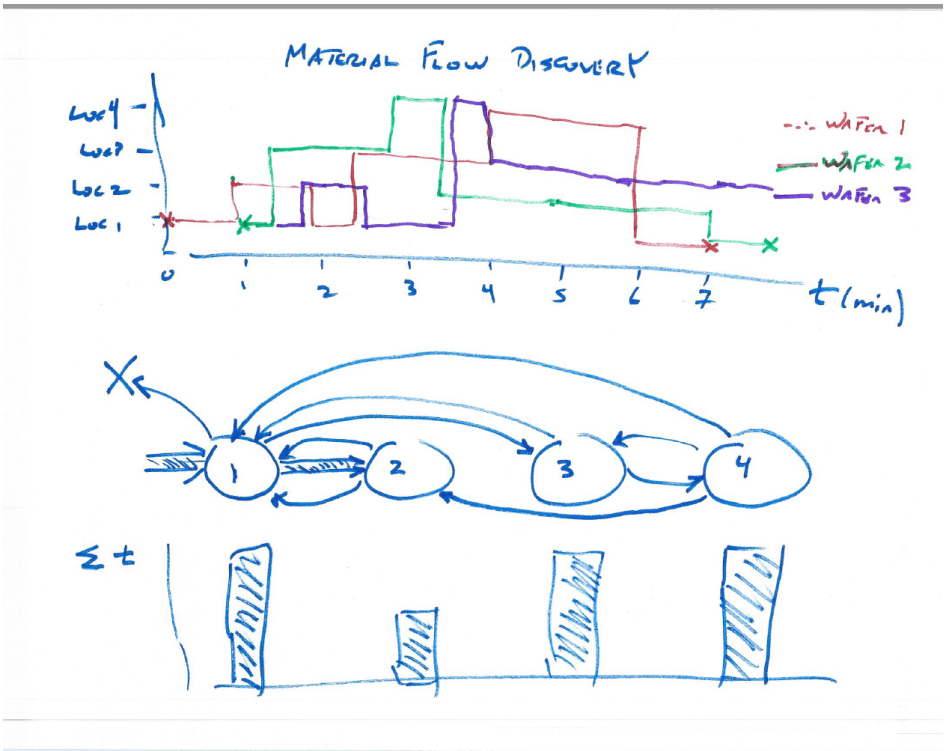
The drawings below show the home screen for the workbench from which the specifics tasks are selected. These fall into 3-4 major categories:

- Importing and filtering static equipment structural data
- Importing and cleaning collected equipment parameter and event data
- Exploring equipment structure using various visualization techniques
- Observing and analyzing equipment behavior by visualizing collected data
- Editing the equipment model to capture insights from static and dynamic exploration to further enhance the visualization.

- Examples include
    - Converting the equipment model tree to a graph with node-to-node relationship information captured on the edges
    - Building sequence diagram and timing charts of material flow through the equipment based on events monitored at the substrate location nodes




NODE / EDGE EDITING
EQUIPMENT MODEL
FOCUS AREA
PROPERTIES
PROPERTIES
PROPERTIES


MATERIAL FLOW DISCOVERY
Loc4
Loc3
Loc2
Loc1
0
1
2
3
4
5
6
7
t (min)
WAFER 1
WAFER 2
WAFER 3
1
2
3
4
Σ t

7 Must-Have Features

List the features without which you would consider your project to be a failure.

For equipment structural data, minimum requirements include:

- Node browser for equipment components with iconography that indicates the component type (module, subsystem, sensor, actuator, logical element) and function (material handling/storage, processing, measurement, inspection, control mechanism).
    - Must allow expand/collapse node, pan/zoom region of tree, filtering of attributes displayed.
- Creation of edges between related nodes and addition/population of attributes that affect visualization behavior.
    - For example, a graph would reconfigure itself based on node selected and “strength” of relationship with other nodes as expressed in shared edges.
- Tree map showing relative “size” of nodes.
    - Size determined by number of parameters, nested subsystems

For display of dynamic data (i.e., parameter values and events collected during equipment operation):

- User selection of parameters to include in a variety of chart types (think Microsoft “Excel”), autoscaling, time window selection, “grouping” of parameters with on/off selection by group.
- Generation of standard statistics vector for parameters over a user-selected timeframe (scatterplot and correlation matrix with histogram distribution plots).
- User selection of color palette for parameter set and auto-assignment of colors to signals.
- Correlation analysis of potentially large number of parameters.

8 Optional Features

List the features which you consider to be nice to have, but not critical.

“Nice to have” features might include:

- Filtering display content based on user (stakeholder) type
    - For example, an industrial engineer would care more about production event timing than PCA plots of process variables.

- Highlighting explicit relationships between dynamic behavior and related structural information.
    - For example, a “heat map” overlay on a node tree map showing where the most active signals are.
- Automatic identification of event sequence patterns, potentially guided by manual grouping of known-related signals.
- “Replay” function with a sliding time window to simulate real-time arrival of signals in the charts.
- “Fingerprinting” of various subsystem behaviors over a meaningful production period to establish a baseline against which future runs could be compared.
- Pareto charts showing relative contribution parameters in a set to a selected phenomenon.
- Parameter”synthesis” and display alongside raw parameters that were included in the calculation
    - Generation and display of a “moving average” signal for a given parameter
    - Combination and display of parameters (Use Power (P) = Voltage (V) * Current (I) to calculate and display P alongside collected values for V and I
- Saving sequences of configuration and analysis tasks as “scripts” or templates that can be applied to other equipment instances.

9 Project Schedule

Oct 24: Deep dive into domain-specific information relevant to project to get everyone up to speed on technical terminology
Oct 28: Acquire and preprocess all data and create cleaned data structures
Nov 4: Get separate working prototype of structural, dynamic, and linked views
Nov 11: Submit project milestone
Nov 25: Prototype of full visualization (“landing”/initial view plus structural and dynamic views; finish up linking any synced visualizations)
Dec 2: Final submission

10 References

- The simulator shown in the figure in Section 4 is a combination of commercial software products from Cimetrix by PDF Solutions (Alan Weber’s employer), including:
    - CIMPortal Plus – equipment-side implementation of the SEMI (Semiconductor Equipment and Materials International trade organization) EDA (Equipment Data Acquisition) standards suite

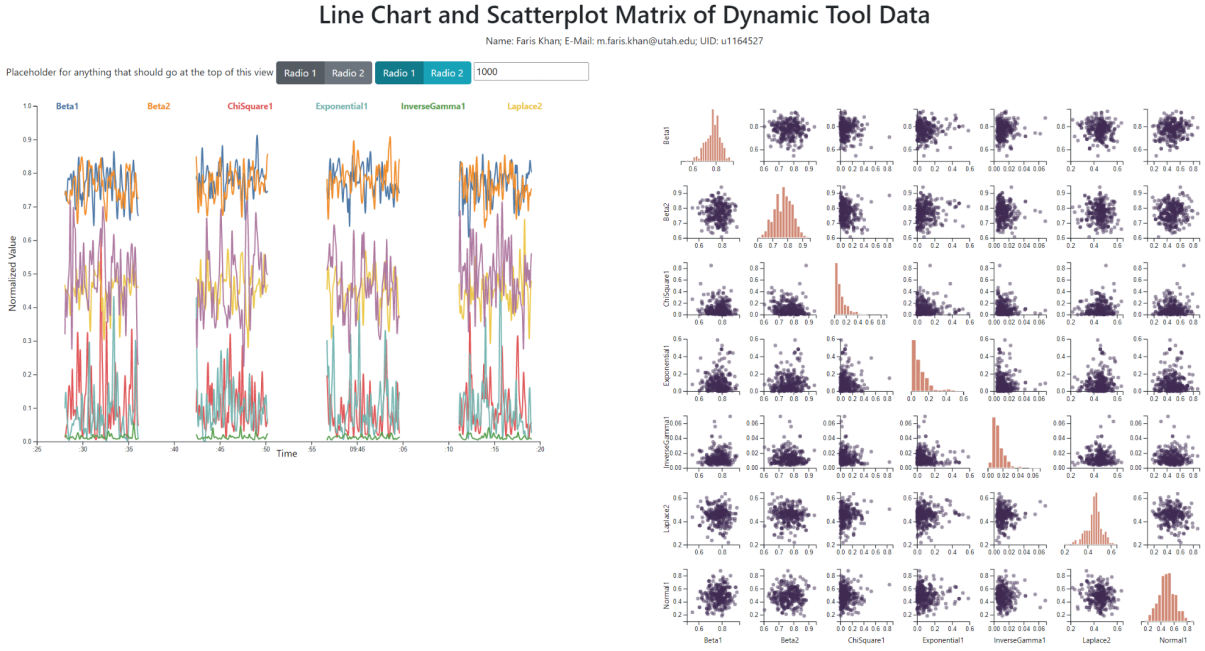
    - CIM300 – equipment-side implementation of SEMI's 300mm automation standards
    - CIMConnect – equipment-side implementation of SEMI GEM (Generic Equipment Model) communication standards suite
    - Sample equipment control application – simulates 2-chamber 300mm wafer processing equipment by running material continuously and generating parameters values and events that would correspond in an automated production setting

- Part of the Analysis and Visualization Workbench is another Cimetrix product–ECCE Plus–that implements the client side (factory) of the SEMI EDA interface. It is an example of a basic model browser, data collection platform, and data visualization utility suitable for testing the interface of a single unit of equipment. ECCE stands for Equipment Client Connection Emulator.
- More information about Cimetrix products appears at www.cimetrix.com.
- The SEMI Standards referenced in this proposal and that will be used in this workbench are part of the Equipment Automation volume of standards. Visit https://www.semi.org/en/products-services/standards for more information. Detailed information about these standards is also explained on the Cimetrix web site.
- Other data sources include log files from data collection servers at a variety of sites and timeframes. These include, but are not limited to
    - Tokyo ElectronLithography track system
    - ASML lithography scanner (exposure equipment)
    - Varian implanter (now part of Applied Materials)
    - Applied Materials plasma etcher
    - Applied Material CVD equipment (Chemical Vapor Deposition)
    - DNS (Dai Nippon Screen) wet bench / cleaning equipment
    - Lam Research plasma etcher

Major Section 2 – Steps to first Project Milestone

1 Scope of prototype

For the milestone, we've implemented basic skeleton code for each of our views. Below is a screenshot of the current state of our view of the dynamic tool data.



Our goal for this view is to have two views of the data synced together. First, a linechart will show the time-dependent nature of each of the traces coming from the tool. Second, a scatterplot matrix will show correlations between the traces. We have the basic implementation done for the milestone, but the only interaction that happens right now is for selecting and de-selecting traces to be visible on the line chart. We still need to implement brush interactions for the linechart to select portions of the data to display on the scatterplot matrices; currently, the two views aren't synced to one another. Additionally, we still need to implement a way for the user to select the bounds for the time axis and filter the data that is shown on the line chart.

In general, we've noticed that the dynamic data has a lot of data points, which makes this view of the data a bit laggy to initially render and update. The scatterplot matrix in particular takes a while to render. In the screenshot above, we've only included 400 of nearly 70,000 data points that are in the full dataset (in real life, an engineer will always have trace data such as this filtered to a narrow time window, otherwise it would always be unreasonable to visualize).

The trace data is quite noisy. We've thought about adding a moving average to the line charts to more effectively show the overall trends of the data. The histograms on the scatterplot matrix might be better replaced with kernel density estimates for a cleaner look. We plan to replace the top right of the scatterplot matrix with correlation coefficients.

2 Data collected

Trace data for the dynamic visualizations was parsed in Python. A jupyter notebook is provided that details data cleanup. For the most part, tool data needed little cleanup. We simply concatenated data files together into a large JSON and CSV. Worth noting is that the tools output values of zero whenever there is no measurement. We changed these periods of all-zero values to NaN's (or nulls) to use d3's line.defined(). This allows the line chart to clearly show where data isn't being collected by having gaps in the line chart instead of connections or all-zeros.

We chose to display all the traces for the line chart on a single graph, so we had a single shared vertical axis going from 0 to 1. We used scikit-learn's MinMaxScaler to scale each trace to the bounds [0, 1]. When we implement some of the interactive features for the plot, we plan to include a mouse hover-over effect similar to the one in Homework 4, where the tooltip displays the true value of the trace as opposed to the scaled value.

3 Data still needed

The data that is being used for the dynamic data views is generated by a simulator. Because we're having some issues in rendering times due to the sheer amount of data points generated by the simulator, we may decide to re-run the simulator and create new trace data that is less noisy and recorded at longer intervals. This is not a lengthy process at all; we have access to the software to simulate trace data, and it would be in the exact same format as the pre-parsed data files used for this milestone. All we would need to do would be to re-run the Jupyter Notebook to parse any new files created this way (if need be!).

4 Important data structures

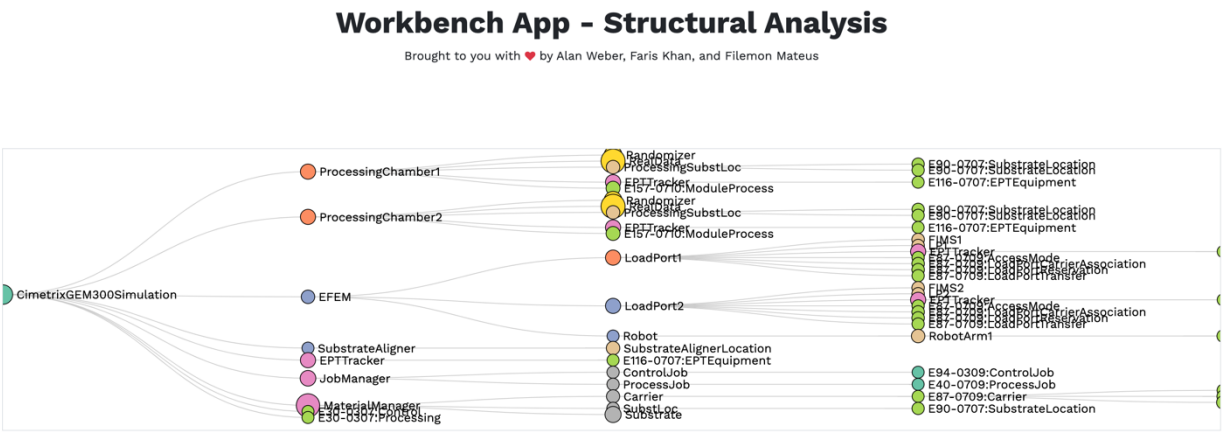
Dynamic trace and event data was parsed to simple tabular files in the form of CSVs and JSONs (only the JSONs are used in the code for the dynamic data views, but we included the CSVs in case there was any trouble).

5 Implementation code fragments

See the GitHub repository https://github.com/faris-k/dataviscourse-pr-manufacturing-vis.

This has all the code corresponding to the different views of the data. We've organized the repository into separate folders for each of the three group members to separately create visualizations for different aspects of the data (workbench home screen, event data, trace/dynamic data, and equipment model visualizations). For the project milestone, we've decided to keep the HTML files for each of these views separate, and we will combine our views as a group into a single file for our final submission.

6 Equipment Structural Data -- Treemap View Prototype 1



name	type	nStates	nEvents	nAttributes	nParameters	nExceptions	nStateMachines
CimetrixGEM300Simulation	Equipment	0	8	0	31	0	2
ProcessingChamber1	Module	0	0	0	13	0	1
ProcessingChamber2	Module	0	0	0	13	0	1
EFEM	Subsystem	0	0	0	6	8	0
SubstrateAligner	Subsystem	0	2	0	0	0	0
EPTTracker	LogicalElement	0	0	0	13	0	1
JobManager	LogicalElement	0	0	0	15	0	2

Dynamic treemap view capturing the hierarchical relations among different equipment components with visual interactivity and exploration.

Our planar treemap view captures the hierarchical relations among different equipment components with visual interactivity and exploration. At the top, we have a hierarchical view of nodes and links/edges that outline the parent-child relationship between different component types. For better differentiation among equipment types, we strategically decided to color them based on their type and encode the intensity of processes in them with their radius--meaning the larger an equipment node size the larger the number of child processes running on them.

We opted to augment the hierarchical view with a dynamic table that offers detailed, structural information about different node substrates of the tree view. Our hope is for the end user to be capable to refer to this table in the event he/she wants to know the

exact processing intensity in a given node; a task that would not otherwise be possible to accomplish with an inferential guess on the node size on the tree view. Future work includes augmenting the tree view with a linear gradient to better capture equipment processing intensity and adding more seamless interactions between the tree view and the dynamic table.

7 Workbench App Overview and Use Cases

Use Case -- Material Tracking

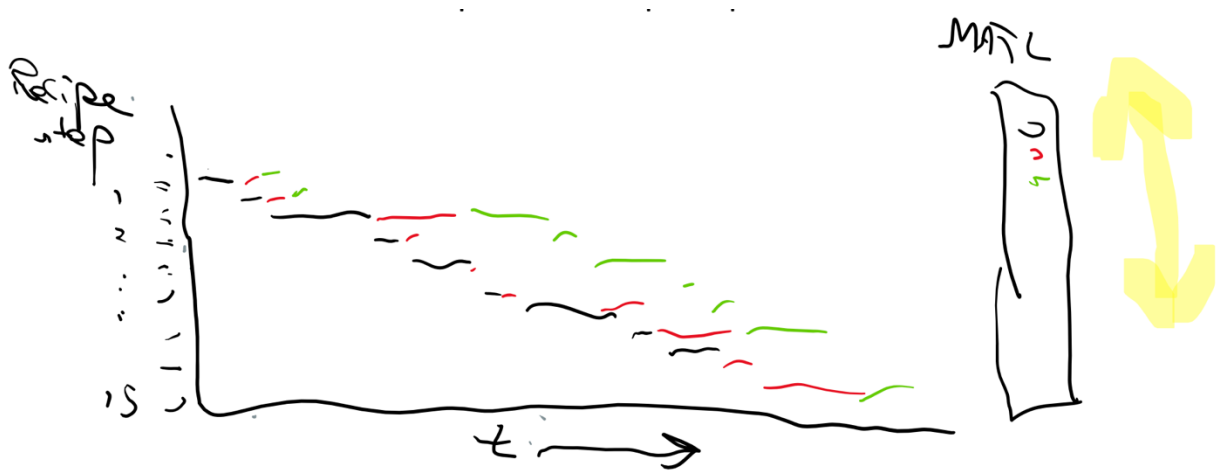
- Data collection required
  - Log substrate location status change events (unoccupied <-> occupied).
  - Collect material ID and timestamp at each status change.

- Visualization
  - Brush material list and location sequence appear on hover.

Use Case – Recipe Execution Tracking

- Data collection required
  - Log recipe execution status changes events (start, stop, step change).
  - Collect module (loc) id, material ID, recipe ID, recipe step #, and timestamp at each change.

- Visualization
  - Brush the material list and show the sequence of recipe steps.



Input Data Formats

- *Substrate tracking reports in .csv format*

eventData.csv

```
Time,Id,CimetrixGEM300Simulation/MaterialManager/SubstLocID,CimetrixGEM300Simulation/MaterialManager/SubstLocSubstID
2022-11-10T00:11:26.944-07:00,SubstrateLocation:1:Unoccupied-Occupied,"Robot","CARRIER1.16"
2022-11-10T00:11:27.516-07:00,SubstrateLocation:1:Unoccupied-Occupied,"ProcessChamber2","CARRIER1.16"
2022-11-10T00:11:28.094-07:00,SubstrateLocation:1:Unoccupied-Occupied,"Robot","CARRIER1.17"
```

- *Recipe execution tracking reports in .csv format*

eventData.csv | VIZ Recipe Execution Tracking_ER_ModuleProcess_5_GeneralExecution-StepActive_00.csv

```
Time,Id,CimetrixGEM300Simulation/ModuleID,CimetrixGEM300Simulation/ProcessJobID,CimetrixGEM300Simulation/ProcessJobIDList,CimetrixGEM300Simulation/RCID,CimetrixGEM300Simulation/RecID,CimetrixGEM300Simulation
2022-11-11T14:26:18.069-07:00,ModuleProcess:5:GeneralExecution-StepActive,"ProcessChamber1","LocalProcessJob0003","Arr[0]","RecipeCARRIER1","RecipeCARRIER1","Arr[0]",1,"Step-0","CARRIER1.13","Arr[0]"
2022-11-11T14:26:19.131-07:00,ModuleProcess:5:GeneralExecution-StepActive,"ProcessChamber1","LocalProcessJob0003","Arr[0]","RecipeCARRIER1","RecipeCARRIER1","Arr[0]",2,"Step-1","CARRIER1.13","Arr[0]"
2022-11-11T14:26:20.176-07:00,ModuleProcess:5:GeneralExecution-StepActive,"ProcessChamber1","LocalProcessJob0003","Arr[0]","RecipeCARRIER1","RecipeCARRIER1","Arr[0]",3,"Step-2","CARRIER1.13","Arr[0]"
2022-11-11T14:26:21.270-07:00,ModuleProcess:5:GeneralExecution-StepActive,"ProcessChamber1","LocalProcessJob0003","Arr[0]","RecipeCARRIER1","RecipeCARRIER1","Arr[0]",4,"Step-3","CARRIER1.13","Arr[0]"
```

Data Structures

- Represent one hour of production at one second resolution
  - locStatus: -1 if no material present, materialID if present
  - materialStatus: -1 if unaccounted for, locationID if at location
  - recipeStatus: -1 if recipe active, {recipeName, stepNumber if active}
- Brush targets
  - Timeline – shows recipe steps, material IDs, and locations active
  - Material list – shows recipe steps and material locations active
  - Location list – shows material sequence in each selected location