**Part 1: Establish a Dashboard for Data Collection and Display**

1. **Define Data Structure:** Determine the structure of the data you want to collect. Ensure it's uniform across different models. Identify the global attributes you want to track.
2. **Integrate Models:** Implement mechanisms to collect data from different models. This may involve using APIs, database connections, or direct integration based on the model type.
3. **Dashboard Interface:** Design a user-friendly dashboard that visualizes the collected data. Use charts, tables, or graphs to represent the data. Consider using web technologies like HTML, CSS, and JavaScript for creating an interactive dashboard.
4. **Inconsistency Handling:** Implement functionality to detect inconsistencies in the global attributes across models. Provide options for users to either push data to synchronize models or accept the inconsistencies.
5. **User Actions:** Integrate user actions into the dashboard interface. Allow users to push data to other models or handle inconsistencies based on predefined workflows.

Technologies Part 1:

* Python for data analysis (NumPy or Pandas)
* Web technology? Could use HTML/CSS and JavaScript
* Database: SQL for storing/managing
* How are we connecting to the database?
* Deployment? Could go cloud like AWS or Docker for container.
* Dashboard design (link to web tech)

For Part 1, we’ll need to establish a global database in SQL to pull in data from. Then process the data using either NumPy or Pandas for analysis. We’re looking for global attributes, so we’ll have to sort the data till we get matching or just run through looking for the name to sort. We can create a universal dashboard as a website using HTML/CSS and JS. If given the databases where we get the data from, any discrepancies in the data or outliers will alert and ask for further actions: to push or accept inconsistencies. The outliers will show up in a different color and will have a link to ask what you might want to do.

??Pushing data to other models: GitHub? Link to the database?

Accepting the data: Just do nothing and leave it as is. Button to accept will remove color/modifications to the value.

**Part 2: Automate Analysis of Textual Communication and Model Usage**

1. **Textual Communication Analysis:** a. Use Natural Language Processing (NLP) techniques to process textual communication from engineers (e.g., Slack messages). Extract relevant information, topics, and keywords. b. Classify or categorize messages to determine if they pertain to model operations or discussions about global attributes.
2. **Model Usage Analysis:** a. Track changes and usage of the computer model, capturing frequency and patterns of attribute value changes, cursor movements, etc. b. Link these operations to the relevant model elements and attributes.
3. **Relate Discussions to Model Operations:** Develop algorithms to match relevant discussions (from Part 2.1) with model operations (from Part 2.2). Use timestamps, keywords, or context to establish relationships.
4. **Infer Important Elements:** Analyze the data to identify patterns where certain model elements or attributes are frequently discussed or undergo frequent changes. Apply machine learning or statistical techniques to infer the importance of elements based on communication and model usage.
5. **Integrate with Dashboard:** Display the inferred important elements in the dashboard, providing insights to users about what aspects of the models are receiving more attention or changes.

Part 2 Tech:

* NLP (Natural Language Processing) (Natural Language Toolkit or spaCy)
* Text mining and analysis (Textblob)
* Topic modeling (Latent Dirichlet Allocation or Gensim)
* Scikit-learn (ML for classification) maybe TensorFlow
* Event tracking (custom logging)
* Usage analytics (google analytics or mixpanel)
* Timestamp matching
* Keyword and context analysis
* Could store data in SQL
* Deploy through AWS
* Might need access to slack API to be able to monitor channels and messages

Part 2 is a little more complicated but I’ve listed the above technologies I might need to use. I’ve got little to no experience in most of them, but if we can make Part 1 work, I think I’ll have a good starting point for Part 2. We’re going to be looking for keywords to associate with the desired stuff (computer models). Keep track of word instances and relate/link them to a summary of the model. Based on frequency of the keywords, structure them in a list of importance to be referred to. Perhaps a dashboard to view all this data.