# Use Case Action Sequence in Detail for New UI 5/30/2013

Black indicates stuff already implemented or user actions  
Red indicates stuff to change/add.  
Green indicates optional changes.   
Blue indicates email quote  
Purple indicates important concepts  
~~Crossed Out~~ indicates ideas I've rejected/ tasks completed

1. Start Program
2. File > New Project…

Projects hold information between sessions so you don’t have to input it every time. Saves to a file.

File > Open Project… Open file dialog

File > Save Project… Save file dialog. Ask to save before exit.

1. *Wizard Opens*
2. Wizard page 1: Setup the data source. Radio button for CSV. If CSV selected, show a text box and a browse button that opens an Open file dialog. Radio button for DB. If DB selected, show text boxes for all the DB info.
3. Press the next button to go to the next page of the wizard.
4. Wizard page 2: Setup the ORules (Organization Rules) for the database. These rules determine the structure of the data. Emerson’s battery data must group rows into batteries, and batteries into cabinets.
5. Click on New O-Rule. O-Rule 1 appears with a blank condition.
6. Select the ‘Group’ condition. Select the ‘Rows’ group. Select the appropriate keys. Emerson’s battery data groups batteries by String Tag, String Number, Jar Number.
7. Possible conditions:
   1. Group by
   2. Column >/</=/!= value. This could be used to ignore ‘marked’ rows from a previous rule.
   3. Column between values
   4. Label
8. Label the rule as “Battery”. The rule changes names. The battery group is added to the Groups list.
9. Click on New O-Rule. O-Rule 2 appears.
10. Select the ‘Group’ condition. Select the ‘Battery’ group. Label this group as ‘Cabinet’ and select the key StringTag. This group will help us manipulate the data of an entire string.
11. Press Next for the next step or Previous to go back.
12. Wizard page 3: Specify Ruleset. This is one place we can edit the rules. These Rules will directly identify the easier problems in the data and label them.
13. Press New Rule. Choose one of the rules. For now, these rules are hard-coded. Just select one.
14. Possible conditions:
    1. Look for groups of \_\_ every \_\_
    2. Column >/</=/!= value. This could be used to ignore ‘marked’ rows from a previous rule.
    3. Column between values
    4. Mark row as \_\_\_\_\_ in column \_\_\_\_\_ . Creates a new column or marks the selected ‘error column’
15. Click Done.
16. The Data Viewer opens.
17. Choose the X,Y axis. Choose the data display for the X axis, Data/Error/Step2
18. The Data is shown in a table in the Data Viewer.
19. Click ‘Graph This’ to view the data in the main window.

# Temporary Instrument faults implementation

# Ideas:

* Projects
* A more robust options interface.
  + Tree View on the left. When a node is selected, change the Frame on the right side.
* Right Click graph to do things. Context Menu.
  + Show DataViewer
  + Change options
* DataViewer
  + Keys can be removed (Is now in Wizard)
  + Ind Axis can be one line with DDM
  + Dep Axis can be one line with DDM
  + Remove ‘Graph Step 2’ button, use ‘Graph This’ button for all tabs, Focused tab
  + Remove ‘Set Thresholds Using’, setup as a Rule
  + Remove ‘Remove Values’, setup as Rules
  + Remove ‘Options’
  + Remove Buttons and etc.
  + Remove everything from the left window entirely.
  + Add a tree view to the left window.
  + Data Tables
    - Remove tabs
    - Add Option above table to select Data/Error Type A,B,C,…/Step 2 next to the X,Y DDMs.
    - If room is cramped horizontally, try listing options vertically on left, since Data Table will only have 2 columns anyway.
* Tree View
  + General, Open Wizard
  + Rules
    - Rules Profiles/save/load
  + Graph Options
* Wizard
  + Page 3: Rules: Same as above

# To Do:

1. Data Viewer filters show filters that are not specific to other selected filters. Ex: Select StringTag: 1098909, and there are StringNumber and CellNumber options available in the drop down menu that don’t exist on that StringTag.
2. StringTag 1098909 has 61 cells. This means not all cabinets have 40 batteries. Wondering if “groups” on this string are still “4”. Or 10% of 61 = 6 batteries to a group for this string. Last group has 7? Relevant for Type 4/5 instrument faults.
3. Data Viewer becomes frozen with no feedback as to what is happening after refresh button pressed.
4. Somewhere in the Emerson database there was a table that had entire datasets within each row as XML. How can we extract this data? Is this data different than the data in the BatteryReadings table?
5. Wizard tabs can’t be hidden. Try something else.1

# Details

* Hello, Liz, Randall, Taghi,  
  I am putting a lot of thought into the Emerson project and I need your help. When we met with a few of the Emerson guys in person, they listed off some 'expectations' of the project. I believe they wanted some kind of data graph that plots data from a database, quickly. Beyond that, I don't know what they really want. As it stands now, I feel like I am just adding features that may or may not be used.

I can't envision what the end result of this product will be or what Emerson is really looking for. It would help me greatly if I had 'vision' for what the product will become. If we really have no idea what the product will look like, I have some ideas:

I imagine what Emerson really wants is a way to quickly identify the exact location and time of a problem that has occurred in their data. This could be a graphed plot of "error types" vs. "Reading Date" where the Y axis is the probability. Say the graph is plotting the nine BDS-40 faults. It would plot 9 different probability lines. At any point along the X axis "Reading Date", the highest line will be the most likely explanation for that particular reading.

Maybe we could switch between graphs of probability / resistance / voltage / resistance of entire string / voltage of entire string / probability of entire string / etc.

I would like to hear your thoughts about the 'vision' of the project.

* The primary goal of Phase 1a is a tool which will look at previous  
  readings and say "based on the data, jar N of string M at site K has  
  just experienced an error of type T." I don't know what interface this  
  tool should have, but I do know that the key concern is being able to  
  react to past data (so no regressions that include future points) and  
  know what the fault is and where it is located, so Emerson can send an  
  engineer with appropriate training and tools to fix the problem. Most  
  important is recognizing whether a fault comes from the sensor itself,  
  the battery being monitored, or any other independent source, such that  
  repair jobs can know in advance where the problem is.  
    
  As for the user interface itself, in the absence of other data you can  
  basically go with whatever's easiest and simplest for you. However, they  
  did seem to like your graphs from before, and database connection is  
  probably important. So basically, take your existing interface and add  
  the new features (fault localization) into that. If you want to get  
  fancy by overlaying probability graphs for different fault types or  
  giving an ordered list which ranks faults based on probability and  
  severity (where severity is given by the user elsewhere), that's fine too.  
    
    
  Sincerely,  
    
  Randall