Operations the configuration system needs to support efficiently

and how to do it

Model for the changes

- How are the shifters viewed in this model?
    - Do they make edit configurations? (enable/disable things) If not, we need to support multiple configurations on our side so they can switch between them. If they do, how do we understand what is running from the configuration name only.
- How do the Detector experts fit in this model?
    - Are they expected to interact with DBE? Are they supposed to give us “files” that we digest for them?
- Defaults
    - Having defaults is easy because it allows creation of new configuration that are reasonable to start with
    - But they are also used to construct different global configurations
        - In general the current configurations are in the form of “this default + this particular specification”
        - Is this model still valid?
    - How easy would it be to change one of the parameters in these defaults? And what is the impact of these changes on the rest of the configurations?
- What happens when we have new versions of the C++/DAQ? How the configurations need to be recreated?
    - Is the model in which the configurations are stored in gitlab still ok? Or do we need to store the configuration in github to be tighter to the release?
    - Since the configuration consists of many data.xml files which are part of the release. What is the working model for those changes? Do we have enough flexibility to generate all the necessary configurations without changing the release related files? Do we have a clear picture of what files are a part of the release of what are not?
- What is the model to decide what goes into an xml file or another?
- What is the role of scripts and/or python interface to edit the configuration? Are we expected to use only DBE to change/generate the configurations? WHat other tools do we have? Are they all integrated or do we need to work on them to make them operation ready?

Operations

- Support for grouping different detector components
    - The idea is to be able to remove them, add them back in a way that is consistent: all the links between detector readout map, controllers, readout units are all set correctly
    - Subcomponent in the detector might be enabled/disabled as well
- Support for including different DAQ components
    - How to set a specific number of DF applications?
    - How much flexibility do we have to play with the number of writers in each DF application?
        - How do we set writing to particular locations?
- For the same detector component, how do we support different configurations?
- How do we enable/disable trigger primitive generations?
    - How do we support multiple threshold schemes for TPG?
- Menu of trigger configurations
    - What is the best way to maintain this?
    - How do we support different trigger rates? (shifters/pre prepared)? Different prescales
- Batch operations: how do we set thresholds, gains, offsets, etc for every channel in the system?
    - How do we expose this to detector experts?

Concept we had that probably are lost for good

- We had override because fddaqconf was not able to configure everything from the command we used to generate full configurations
    - We believe this is not an issue in this model, we think this is worth mentioning to be sure