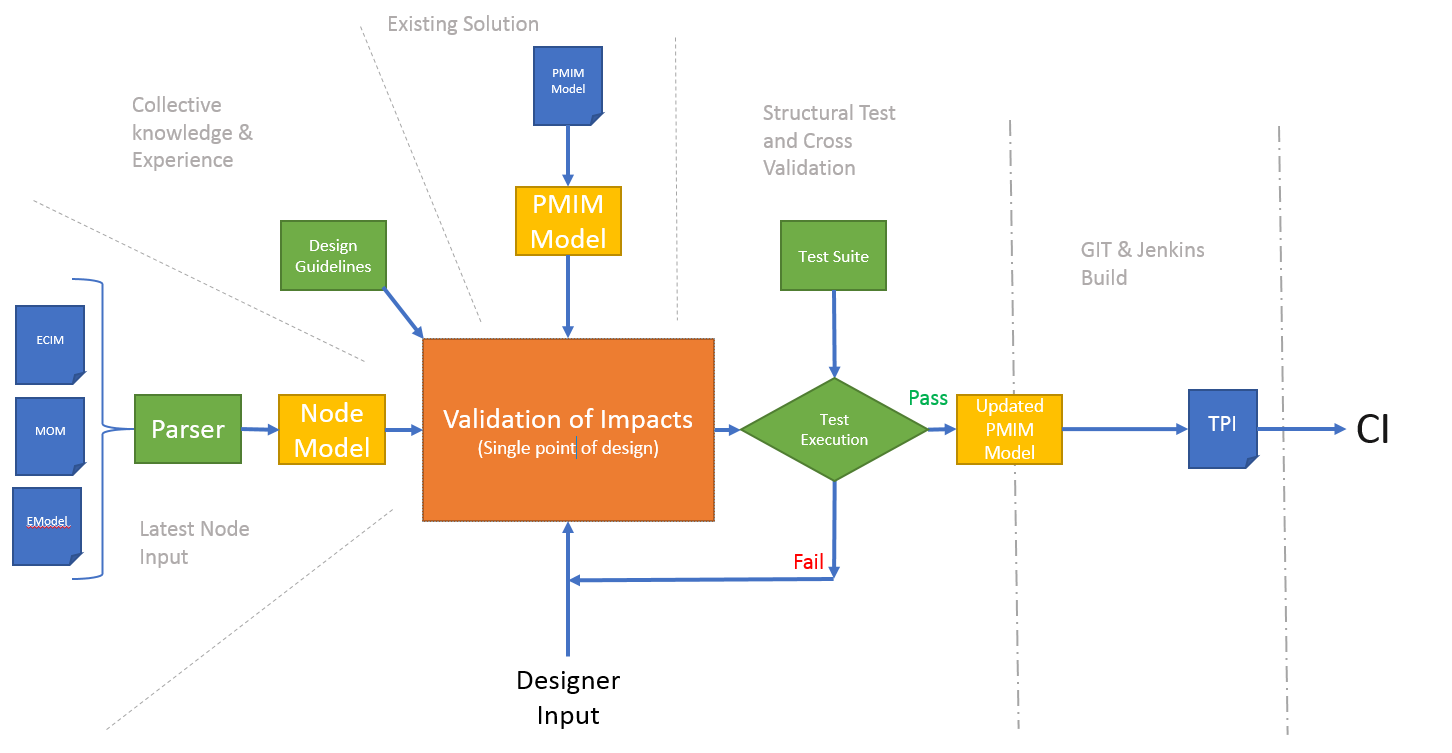
PMIM POC



# Overview

The overall aim of this POC is to demonstrate the feasibility of a TP design environment that maximizes the automation possibilities and provides a single interface where all manual interaction can be provided.

Utilizing the functionality already available in the MOM Tool and TPC, along with adding mechanisms to include our collective knowledge and experience into the design of every TP and a test suite of structural tests and cross validation will provide designer with a complete environment to design quality TPs.

## Concept Walkthrough

The diagram above breaks down the flow into 6 elements:

1. Latest Node Input
2. Collective knowledge & experience
3. Existing Solution
4. Designer Input
5. Structural test and cross validation
6. GIT & Jenkins build

Each of these elements plays a significant role in the design process and are all required to ensure consistent quality and traceability of our solutions.

### Latest Node Input

This element of the environment is currently covered by the MOM Tool. The MOM Tool provides the ability to parse various inputs from nodes and converts the input to an Excel output. The Excel then undergoes some manual changes and is provided to the designer.

Some of the functionality offered by the MOM Tool will be utilized here. The ability to parse the node inputs is required but instead of writing to an Excel file, the node information will be stored in a model that maintains its structure and relationships.

This element must be capable of loading any node model and describing it in common model structure. The architecture must support the ability to add additional parsers with ease and minimal impact to the overall system.

### Collective Knowledge & Experience

A number of default rules are applied to designing every TP. These rules include adding a regular counter to a regular table, a vector counter gets added to a \_V table and a pmFlex counter gets added to a \_Flex table. These guidelines can be used to predict conflicts or the need for a modification and describe a recommended solution to the designer. Rules or guidelines like this can be automated and applied to the solution removing the mundane tasks from the designer’s job.

These rules/ guidelines represent the collective TP design knowledge we possess and should be applied to every TP we design. Ideally, these rules should not be part of the deployable tool but be an add-in that is loaded from a central location (GIT repo for example). This will allow us to update the rules and guidelines independently without having to roll out a new version of the tool, impacting all designers.

### Existing Solution

In an upgrade scenario, the existing solution would be loaded from GIT to provide the base on which to upgrade. The existing solution file would be similar to the current TP model but with the addition of the modifications into the structure.

An existing solution is made up of a TP model, interface model/s, universe model/s and modifications to be made. One area this POC should consider is the implementation of these models and their storage/ control. The concept of a PMIM model is a model that describes everything required to create a full solution with no further manual intervention required. The implementation of PMIM does not, however require that the entire PMIM model be stored in a single file.

How this tool and the design process handles the flexibility within the PMIM concept will need to be addressed. For example, today we have an ERBS G1 and a G2 TP, each of which can have a universe. But, there may also be a universe that combines G1 and G2. In that scenario, the storage of the combined universe model needs to be considered. Should the combined universe be

1. Stored in the G1 PMIM
2. Stored in the G2 PMIM
3. Stored as its own PMIM that is dependent on the G1 and G2 PMIM

Option 3 dictates that a PMIM is merely a conceptual model loaded from multiple files which make up relative parts of that model. A solution like the combined universe is a secondary requirement but any change made to the G1 PMIM may have an impact on the combined universe so this POC must, at least, consider the possibility of solution requirements like this moving forward.

### Designer Input

Using the node details collected in the node model, the knowledge defined in the guidelines and the existing solution, it is then possible to bring all this information together and determine what impacts the new node model has on the solution.

It would be possible to highlight what new counters need to be added, if any existing counters have changed or if any tables have changed. In most cases the guidelines could define what action to take for most of the situations but there will be some scenarios that will require a designer to make a decision.

The designer should then be presented with all the actions that can be taken highlighting those that require a decision. It is important that the designer approves the changes that can be automatically added to ensure the impacts are considered from a solution level. The changes the designer makes must be run through a series of verification checks to highlight scenarios that would impact upgrade times of the TP. This is not a blocker test as the impact may be acceptable but the impact should be highlighted and approved by the designer.

The designers interface to the tool should also allow them to update all aspects of the TP so they can add in the additional information required like Busy Hours, Transformations, External Statements etc…

### Structural test and cross validation

When the designer has completed the design activity the tool should be able to apply a test suite over the TP to confirm its content. These test cases should cover the structural integrity of the solution but not the operation i.e. confirm every counter that can be loaded has a corresponding object defined in the universe but not that an aggregation set executes as expected.

The scope of these tests is still under analysis and discussion but the mechanism to apply them should be added to the tool. Similar to the rules and guidelines, the test cases to be executed should ideally be stored in a central location so that we can independently add additional tests or fixes without the need for a roll out of a new tool version.

### GIT & Jenkins Build

The design environment tool must not be capable of outputting a tpi file. When the designer has completed the design and the solution has passed all the tests in the test suite then the tool should publish the solution PMIM to GIT. The Jenkins build can then trigger a variation on the existing TPC tool today to load the PMIM and generate the required outputs.

The implementation of the GIT and build mechanism is dependent on the storage and implementation of the PMIM concept. A single file that describes the entire solution requires TPC to load that file and produce multiple outputs. A PMIM concept made up of multiple file inputs means TPC could load the contributing files independently and produce the output i.e. load an interface model to generate an interface tpi without needing to load the TP and universe also.

## Project Components

The functionality available in this tool is also required by TPC for the DM build environment as well so there is a significant overlap. To avoid the need to build a single large all-encompassing tool, it would be best to break TPC down into its component elements and use those elements as a base to build this tool.

A high-level overview of the element breakdown could be:

1. Modelling Structures
   1. TP Model
   2. Intf Model
   3. Universe Model
   4. Node Model
2. Utilities
   1. SFTP
   2. SSH
   3. DB Access
3. Node model parsers
   1. ECIM Parser
   2. MOM
   3. Etc..
4. Create abilities
   1. Create a TP to the DB
   2. Create a Intf to the DB
   3. Create a Universe
5. Environment
   1. Collect and package an ENIQ environment
   2. Deploy an environment and create the DB structure.
6. TPC CLI interface
7. PMIM GUI interface

With a breakdown of the elements similar to above, it allows us to maintain a single codebase while creating and deploying multiple tools. The PMIM tool would be made up of items 1 – 5 and 7 where as TPC only requires 1, 2, 3, 5 and 6.

## Scope of POC

The scope of this POC is to demonstrate the feasibility of the single design environment for TPs. The scope will be limited to loading the ERBS ECIM node model and identifying counters that need to be added to the TP.

The counters will be identified by name alone without any of the related data i.e description, vectors etc… and demonstrating how those counters would be applied to the existing solution. It is not necessary to create the resulting TP as it is already known that given an input model, TPC can create the TP in the local DB. The scope of this POC to demonstrate the feasibility to create the PMIM Model only.

As the competence around the scope of manual modifications is currently limited, it is not necessary to produce an automated solution. The POC must consider the application of modifications and gather understanding of how a redesign of the sets generation code would affect both the modelling of the modifications and the process in general.

## Acceptance Criteria

The POC must:

1. Accept the ERBS ECIM node model
2. Identify any new counters to be added to the solution
   1. By using design guidelines and the existing ERBs solution
   2. Highlight any conflicts or decision points that are required
3. Demonstrate how the conflicts can be resolved
4. Produce the resultant PMIM model including all new counters
5. Apply a test suite to verify all new counters have been included in the solution.