# Current Design Process

The current TP design process consists of multiple steps where some are automated and some require manual intervention. The steps currently required to design a TP include:

1. Parse the node model using MOM Tool, creating the delta doc
2. Update delta doc with additional information
3. Merge delta doc with Model-T doc
4. Use Model-T to generate TP in the DB
5. Analysis and review of manual modifications
6. Apply manual modifications and generate tpi

Each of the steps outlined above fulfils a specific task within the process and each task in turn results in information that is ultimately required to design a TP. However, the disjointed nature of the tooling currently in place and the multiple transitions from automation to manual and back to automation introduces several potential points where errors can be added to the process.

The node model provides the base information to define a schema for the tables within the TP and the TP designer then provides what can be described as the “value add” of the TP. It is important to capture this information correctly and provide the designer with all the information required to make this job easier. The current process does provide the required information but in different formats across multiple stages.

## Potential problem areas

The current design process has several areas that cause potential problems. In some cases, the chances of a problem arising may be low but the possibility of introducing a problem exists and should be identified and managed.

### Delta Doc Generation

The current operation of the MOM Tool is to take in 2 versions of a node model and produce the delta between them. This delta is then used as an input to update an existing TP. This approach is predicated on the assumption that every delta from the node models are implemented into the TP. It also heavily relies on the consistency of the node models over multiple revisions.

As the delta doc generated goes through manual intervention and merging into the TP, it is difficult to estimate the volume of errors introduced at this stage but the possibility exists. Where possible, we should ideally avoid being dependent on the quality of items we do not control.

### Updating the Model-T

This task can be broken into two parts. The first part is the inclusion of the information from the delta doc which can be described as a “copy & paste” activity. The second part requires the designer to populate the information required that is not provided from the node model e.g. transformations and Busy Hours. The vast majority of issues in TPs are introduced at this stage. The issues introduced can be described as human error but most, if not all, of these issues are preventable.

### Manual Modifications

Manual modifications are the single greatest challenge in TP design today. Their very nature is fully dependent on the designer input. The modifications made to the TP define the operation of the TP so the SQL provided must be correct and even their order of execution must be considered and defined.

The need for these modifications is driven primarily by the complex and inflexible code that generates the part of a TP known as the sets. Another driving factor is the use of modifications to manipulate the sets to facilitate legacy implementation in some TPs.

In our current situation, manual modifications are very often overlooked, incorrect and even completely forgotten about in the design process. Adding to this, designers are unable to define some modifications until after the TP has been created and in some cases, not until the TP fails a test cycle.

It is important to address at this stage that no automation or tooling can address this problem alone. There is no doubt that the tooling has a part to play in the issues caused at this stage but the mindset and competence around these modifications is also a significant factor.

# Proposed Solution

## Aim of the Proposal

The overall aim of this proposal is to define an environment that facilitates the design of a TP and all the steps that it requires. This environment should accept all manually defined information at one stage and once the TP model is defined, full automation should be capable of completing the creation and testing tasks with zero manual intervention.

As it is not possible to design a TP without human intervention today, the competence factor must be considered within the proposal. Any tool to facilitate the design of a TP must also aim to facilitate competence building within the design teams. This aim makes the availability and presentation of information just as important as the capabilities of the tool.

## Overview of the Proposal

Using the conditions already described in the MOM Tool on how to parse the various node models allows us to build up a structure of the available information from the node. This can be described as data point 1 (DP1). This data point describes the structure of the MOs from the node and what counters are contained within each MO.

The next phase is to use DP1 and apply a defined set of rules. These rules should cover the scenarios we know how to handle in TP design and that we can automate. For example, if counter abc is a vector counter within the MO xyz then we know that the counter should be added to a vector table where the tag for that table would be xyz\_v to signify that it is a vector table. There are many rules similar to this that could be included to automate the design aspect of the TP. Using these rules and comparing the information available from the node against the existing TP will highlight what situations can be automated and what situations require a human decision. This structure after having applied the rules and highlighting the decision points can be described as data point 2 (DP2).

DP2 fulfils the “copy & paste” activity in our existing design process. The collection of changes to be made to the TP can be edited, where needed, and approved by the designer to be applied to the TP structure. The edit functionality allows the designer to address the situations that could not be handled by the predefined rules and also provides the interface to add the “value add” information. After the designer has completed this task, we now have an updated DP2 describing all the changes proposed for the TP and DP3 which describes the proposed TP model.

To address the manual modifications aspect, some scenarios may be suitable to be automated by the predefined rules and flag to the designer that a modification may be required. Ideally, all modifications would follow this approach but further analysis and competence building would be needed to fully understand this. Pending further analysis and POC, the feasibility of handling the modifications adequately may need to be considered and addressed accordingly.

A redesign of the sets generation code needs to be undertaken. The code is complex and offers no flexibility to the designer to alter their operation. Offering some flexibility within the sets generation code would allow the designer to describe an alteration much easier than today. Also, improving the sets generation code to enable generation for a specific table rather than the entire TP allows us to update and improve DP3. DP3 now describes the proposed TP model and the resulting sets, all of which can be generated on the fly at runtime. This ability allows the designer to see the impact the modifications they make will have on the sets at runtime and enable them to update DP2 and DP3 to describe the modifications as part of the sequence of changes to be made to the TP.

Once the designer is satisfied with the proposed TP model they can publish the model which will be added to version control and scheduled to run though the automated build, CI and delivery processes.

DP1, 2 and 3 make up the interface to the designer and these data points provide all the information for analysis and design of the TP. The creation of these data points allows us to break the design process in clear stages where each stage can be controlled in isolation. This approach allows designers to describe changes in DP2 and view how they would look in DP3 without having to worry about making a mistake. If a mistake is made, they can revert the change and DP3 will be updated accordingly enabling a safe learning environment that can be used to build competence in TP design.

## Design requirement

The parser logic from MOM Tool needs to be implemented into a generic structure to make it easier to add and edit parsing capabilities moving forward

The output of the parser needs to be changed from xlsx to a model maintaining it structure (DP1)

The automated rules to apply and a mechanism to apply them

The creation of DP2 that describes the structure after the rules have been applied and will track the changes the designer wants to make.

The sets generation code needs to be rewritten to allow inputs to alter the sets being generated.

A full GUI is needed to enable the designer to view and edit the TP structure.

## POC required

The existing sets generation code contains handling for specific TPs in specific ways making the code overly complex. Also, the feasibility of being able to manipulate the sets as easily as we would like is currently an unknown.

We need to understand if its feasible to redesign the sets code in a generic fashion that:

1. Support these specific handling already implemented
2. Supports our requirement to simply manual modifications
3. Enables single tracking and wont create issues should there be a change to the sets in a future ES release.

The old sets code will have to remain in the platform to support the TP IDE externally. This is a undesired setback as having to support and design two versions of the code is far from ideal.