Styles

Introduction

Often I come across models that have 100s even 1000s of styles, a clear indicator that they have been created in an ad hoc fashion, i.e. "made up as you go along". To do this is not a show-stopper, but in the age of Shape Regions, which now require a much more wide-ranging set of styles (especially small text and transparent backgrounds as well as various "heatmaps"), it increasingly makes the Styles library unmanageable, which actually leads to the creation of an ever-expanding set of Styles so compounding the problem.

Typical Naming Conventions

For example, someone wants to add a shape region style that has say a transparent background and 7pt Arial text and no border. They look at the list of styles that might have started with a "semantic" naming convention, e.g.

- Organization (swim lane)
- Process (main)
- Process (activity)
- Technology (logical)
- Technology (physical)

Interspersed with this maybe attempts to get more precision:

- Organization Name
- Organization big picture
- Technology SR type 1
- Technology SR type 2
- Technology disk used

In a multi-user situation where individuals are allowed to create their own Templates and they only half understand styles, we might also have things like:

- Process (default yellow)
- Shape region grey small text

But notice one thing: not a single style is reusable. So, they create another one, even though the existing style called "Technology disk used" is exactly what they want (but as they are applying this style to an Entity they may not want t use it anyway).

This is how you end up with loads of duplicated styles making it harder to maintain (what if you want to change all instances of any style with a 7pt Arial font to 6pt Verdana)? After a while, someone comes along and says actually, the colours you are using are not allowed anyway and there is a need to rebrand all diagrams with corporate colours.

Alternative Naming Convention

A solution to this is to make the Styles purely independent of the objects to which they are applied and using a naming convention that clearly states what is in the style; e.g. something like this:

- Colour [Transparency and percentage] Font-size [Font-colour] [Font-face] [No Border]

The square brackets are optional and used only if they deviate from a common standard. So, suppose the colour palette is controlled (so there are clear names for the allowed colours), the font standard is Verdana in black, the shape has a border, and there is no transparency, the we might see a set of styles as follows:

- Sandstone 08pt
- Sandstone 09pt no border
- Sandstone\Transparent (50%) 10pt no border
- Sandstone\Transparent (75%) 11pt blue-text no border
- Sandstone 12pt blue-text Arial no border

If colours do not have specific names, then it is better perhaps to add an RGB value, so you do not end up with subjective descriptions, e.g.

- Green (0-128-0) 10pt no border

This can, of course be an extremely time-consuming job. At one client, e.g. they had a corporate colour palette of 12 colours. So, I created styles as follows.

- For each color, create a separate style that uses font sizes from 7pt-12pt
- For each Font size, create a style with and without a border

That equals 12 styles per colour. In addition, for each color, create the same as above using 50% Transparency a further 12 styles. This resulted in 288 base styles. In addition, there were specialist styles that might have larger fonts or styles set up specifically for lines and heat-maps and styles that have the same colour for fonts and shape (so making the text invisible).

This was necessary for this specific client as only those 12 corporate colours were allowed to be used and using this naming convention allowed for easy re-use.

This was a time-consuming job; it took one person two days to create these. Then once I had my library of 300 or so base styles, I could recreate the Diagram Templates and reapply them which was a further time-consuming job as there were 50+ Diagram templates.

In short, paying attention to Style creation as part of designing modeling standards is far better than trying to deal with it later.

Managing Styles

As described above, unless Styles have been activity managed, there is a dynamic that results in more and more redundant styles being created. This can sometimes lead people to try to clean up the Styles library and delete unused Styles.

The main problem here is that Styles do not act like normal objects. You cannot download them with Auto Modeler or report on them with Reporter, which means you cannot see where they are being used. And if you try to delete them, they will be deleted regardless of whether they are being used or not.

Therefore, if you want to try to rationalize the Styles library, a useful tool is an un-supported utility, ModelpropertiesViewer.exe, which identifies where a style is being used.