

Autonomous Commands And Standards Control

Commands that think. That's what we need. A command like that could perform an action one way on one element and then differently on another. Areas of shapes on level "parcels" are exported to spread sheet column "A" while areas of shapes on level "remainders" are added to spread sheet column 2. Instead of two or more commands you only need one. And standards control would no longer require the CAD Police. North arrow cells could be forced to a level called "north arrow" not just during placement, but during copy or move. Then again maybe this type of command is closer to reality than we think.

In implementing some rules commands might act autonomously, on their own to determine how to process a certain type of cell, line, or shape. This hybrid command would save time, control design quality, and would be easier to learn.

Consider the delete command I created for my Right of Way Tract Management application. Our designers decided early on that tract bubbles (a circle with a number in it) were not to be removed from the right of way map but should have the word "DELETE" added to it to indicate that they are no longer included in the project. Later, our design team decided that this command needs to be a toggle. And finally they decided that with all other symbols the command should work just like MicroStation delete.

This one command has three distinct operations depending on the symbol (a MicroStation cell element).

1. If the user deletes a tract bubble and it does not have the "DELETE" text then add the text to the symbol but do not remove the symbol from the file.
2. If the user selects a tract bubble with the "DELETE" text then the command removes the text from the cell and the tract bubble remains.
3. Any other symbol is deleted from the file.

I decided that the best way to distinguish tract bubbles from other elements was by type (a cell) and by cell name. You may have chosen to add a tag to the cell at placement time, or to look at the component count in the cell. Any approach that isolates the tract bubble from the other symbols would have worked.

The production code has been implemented in MDL but to simplify this discussion I am presenting you with a VBA version for MicroStation V8. Upon accepting the element the following test is performed.

```
' test for cell else delete
If Element.Type = msdElementTypeCellHeader Then
    Set oCell = Element
    If StrComp(Left(oCell.Name, 3), "TRT") = 0 Then
```

```

toggleDelete oCell
Else
  ActiveModelReference.RemoveElement Element
End If
Else
  ActiveModelReference.RemoveElement Element
End If

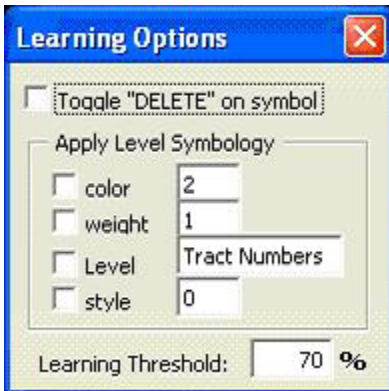
```

Now, what if the command could learn? In the previous example we explicitly set the rule so that the command applies the DELETE toggle to all cells whose name starts with TRT. And for most application designs this is good enough. But in setting a rule in the software code we have a couple of problems. First, the end user can't change the name of the cell, and second, the user can't apply the delete toggle to more than one cell.

If on the other hand we keep a weighted index of the cell names used we can apply the rules to the highest ranked item. Here's another way to look at this. Upon placing a cell the user indicates that the cell is to be modified by adding the word "DELETE". This may be done by turning on a toggle button (a check box).



And the toggle button when checked on opens a learning options dialog. That is, a form that allows you to tell the command what to learn. It might look something like this.



When the learning option "Toggle DELETE on symbol" is checked cells selected will have the text "DELETE" added or removed but the symbol will be preserved. When off the symbol is removed from the file.

Each time this operation is performed a support file is updated. This file will contain the name of the cell(s) and two corresponding indices. One index is the number of times the cell has been subjected to learning. More simply the number of times the delete command was used on a cells of this name. The second index is the number of times learning is

applied to the cell with the “Toggle DELETE on symbol” option checked. A listing of the file might look something like this.

```
Tract Number,10,10
Tract Symbol,10,4
Structure,10,0
North Arrow,10,0
```

We want the second index to increase each time learning is applied with the “DELETE” option checked, but we also want it to decrease (to a minimum of zero) when the “DELETE” option is not checked.

On the Learning Options form is a text control labeled “Learning Threshold”. In this example the threshold is set to 70 percent. When the ratio of DELETE to learning sessions is greater than 70 percent then the command will continue to use the toggle DELETE rule even when learning is turned off.

Instead of testing the cell name directly use a function that can indicate what to do with the cell. The accept function now looks like this.

```
If markAsDeleted(oCell) Then
  toggleDelete oCell
  If learning Then
    updateDeleteFile oCell, True
  End If
Else
  ActiveModelReference.RemoveElement Element
  If learning Then
    updateDeleteFile oCell, False
  End If
End If
```

The new test function markAdDeleted would have the following psuedocode.

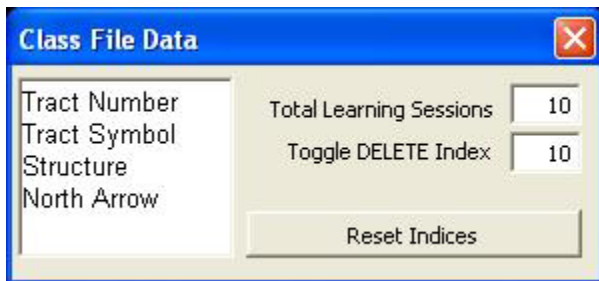
```
If Learning Then
  Use the DELETE toggle on the learning form to set the return status.
Else If not learning
  open delete class file and look for cell name
  If cell name found then get indices.
    If index ratio is >= learning threshold Then
      return true (means apply the rule)
    Else If index ratio is < learning threshold Then
      don't apply the rule.
    End If
  Else If cell name is not found then don't apply the rule.
  End If
```

Close the delete class file.
End If

The new update file function would have the following psuedocode.

```
Open the delete class file
If cell name found Then
  Add 1 to the learning index
  If increment Then (second parameter passed)
    Add 1 to the DELETE index
  Else
    Subtract 1 from the DELETE index
  End If
Else cell name not found Then
  Append name to the end of the file and set learning to 1
  If increment Then
    Set Delete index to 1.
  Else
    Set DELETE index to 0.
  End If
End If
```

A few things to note about this approach to standards control. First, in the interface that you design be sure to provide the user with controls to reset the rule for a particular cell. This will likely require building a form with a listbox (displays all of the cell names) and to the right of the list add controls for the indices. Don't make the user open the class file (the text file) directly. You may also want to limit the number cell names in the class file saving only those with say the twenty highest ratios.



Realize too that this form allows the CAD manager to edit the indices directly. In many cases it may be faster to force learning than to allow learning to occur over time. Next, if you could use the same file for all of your users in a distributed environment then new users would be able to adopt the standards set by experienced designers thus reducing time needed to learn standards. Setting a configuration variable at the standards level of your workspace could point to the location of your control file. All "experienced" users would be able to teach the command simultaneously thus reaching the learning threshold faster.

And finally, since this is a not an easy concept for many users to understand include a control to turn off or hide the learning tool. Passwords and/or and experienced users list may be appropriate in your organization.

Of course our delete tool is a very simple example and reason dictates that more complex systems could learn design standards as well as graphic standards. Think about placing details for architectural features by selecting just one command. Identifying the door, window or roof penetration places the correct detail. The software decides based on rules it has learned from experienced designers.

Don't take my word for it. Check out the latest versions of Microsoft Word, Excel or Outlook. Select any pull down menu and notice that the list is abbreviated to just the commands you have used most recently. Certainly we can expect technology such as this to be extended not just to office automation but to design automation too. It's just a matter of time.