

Ten Workspace Design Ideas

by Mark Stefanchuk, cadgurus.com

Recently I had an opportunity to help a design team at a large process facility set up a “packaged” workspace which means that all of the components are deliverable in a single file.

The team already had an old MicroStation workspace configured to handle cell libraries, seed files, font resources, macros, and other design tools. And the workspace operated correctly in their office, but extracting the tools from the workspace for distribution was a difficult process. It was not clear which tools in the workspace were being used and which ones were not. As a result, replicating the workspace for a new environment seemed impossible.

This scenario is not uncommon. Many design teams focus on design and not on the automation tools and that is the way it should be. But, keeping track of automation resources is a necessary activity. So we had to simplify the procedure. We had to find some ways to consolidate and clean up the 75 megabytes of work-space and make it manageable.

After some preliminary study, I discovered several areas where the workspace could be improved. I will share these with you in the following paragraphs by describing ten ideas for workspace design.

Document It!

Documentation is important. The next time you have to update a cell library, you need to know which one and where to find it. Comments in the configuration files are good, but non-technical notes in a word document make workspace management much easier.

The WORD doc needs to be one that describes which, directory paths or files, contain cell libraries, font resources, user commands, seed files, macros, and other applications. A simple two or three page document that explicitly describes the location of these workspace components will make understanding your workspace very easy.

Keep it simple and just say something like,

“All macros are contained in the following workspace location. K:\cadd\ws\wsmod\macros\. Place all files with a .bas extension in this directory.”

Or,

“All standard cell libraries are contained in the following workspace location. K:\cadd\ws\wsmod\macros\.”

Notice that I used explicit definitions. I say, k:\di-rec-tory... and not \$(variable)/directory. Or Variable= direc-tory. This word document needs to be a road-map for the design team that will add cells to cell li-braries and make changes to seed files.

Use Lots of Variables

Limit the number of places that you make explicit assignments in your configuration files. In all cases you can eliminate configuration variable assignments that include the entire file specification. In other words, avoid using MS_CELL>p:/workspacedirectory /wsmod/default/cell/

Instead use a variable definition like,

```
MS_CELL>$( _USTN_WORKSPACEROOT )  
wsmod/default/cell/
```

This will allow consultants to place the workspace anywhere on their system with minimal changes. That translates to fewer support requests.

Consolidate

Keep all workspace components in the same direc-tory. This makes it easier to deploy the workspace to other parties and makes it possible to create a zip file that extracts to a single root directory.



I can think of only one exception here and that might be the location of user directories. It may be necessary to point to `k:/$ (USERNAME) /` for some variables, for example `_USTN_USERNAME`, while the rest of your workspace points to

```
_USTN_WORKSPACEROOT=h:/workspace  
directory/.
```

In this case I would define a dual configuration. Since I don't know how the consultant might have their system set up I would use the local workspace directory and copy a default.ucf file from my workspace to the local `Bentley/workspace/user/` directory.

Use an Icon

This is a simple method for specifying the location of your `mslocal.cfg` file or as we will see in the next section, provides a means for executing a batch script. And, you can use the target and start in fields to define some preliminary configuration variables.

If you define the start in field of the icon to be `C:\Bentley\Program\MicroStation` then the following variable is predefined and locked by MicroStation.

```
_WORKDIR=C:\Bentley\Program\MicroStation\
```

Consider setting the "Start In" field of the icon to your workspace root directory. You can then define the following,

```
_USTN_WORKSPACEROOT=$( _WORKDIR)
```

By doing this, there is no need to edit any of the configuration files. They can expand all other variables based on `_WORKDIR`, a few Bentley predefined variables, and system variables like `USERNAME`.

Batch Commands and Scripts

Either of these are excellent automation tools. Check for the existence of the appropriate user configuration files, or use a script to rename files.

Visual Basic scripts have security issues, but they run silent - great for reducing support questions.

Here is an example of a batch command (See code below) that is launched from a desktop icon.

The command checks to see if `c:\win32app\ustation\config\mslocal.cfg` exists, checks for the existence of a `.ucf` file, and then launches MicroStation.

The batch command reads command line arguments into `%1`, and `%2`. In this case, the variables are the workspace location, and the MicroStation directory respectively. The command also makes use of an NT system variable called `USERNAME`, defined by the users NT login name to build the user configuration file.

You can do the same thing with a visual basic script. Some network administrators, as a security measure like to change the application associated with `.vbs` files to notepad. If this is the case in your office then use a batch command.

If you have the option to use visual basic scripts, use them. The application will run silent. By silent I mean you do not see a DOS window. When a batch command is launched a DOS command window is displayed and the actions are displayed to this window.

Even with the "@echo off" option many actions are still displayed. But using a visual basic script the commands execute without any visual queues. Not that it is a good thing for a user to not know what is happening, but the absence of the command window will reduce support calls.

Here is a similar command using visual basic script. A few more lines of code, but I think it is easier to read, debug and maintain. See code on page 17.

Custom mslocal.cfg

Use the `-wc` option in the start up string to specify a configuration file. This is a great way to tell MicroStation where everything is located. Remember that the batch file uses the start up line,

```
% 2\ustation.exe-wc% 1\cfgvardir\config  
\mslocal.cfg
```

```
@echo off  
if not exist %2\config\mslocal.cfg goto CHKTWO  
if exist %2\config\mslocal.cfg goto MOVELOCAL  
  
:MOVELOCAL  
move c:\win32app\ustation\config\mslocal.cfg c:\win32app\ustation\config\mslocal_cfg  
goto CHKTWO  
  
:CHKTWO  
if not exist %1\config\user\%USERNAME%.ucf goto MKDFLT  
goto BEGINCOPY  
  
:MKDFLT  
copy %2\config\user\default.ucf %1\config\user\%USERNAME%.ucf  
goto BEGINCOPY  
  
:BEGINCOPY  
%2\ustation.exe -wc% 1\cfgvardir\config\mslocal.cfg
```

```

' check for ucf file, msconfig and launch microstation
i =1
for each arg in wscript.arguments
  if i = 1 then
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set fldr = fso.GetFolder(arg & "\\plntware\pwpid\projects\mo")

    workdir=arg
  end if

  if i = 2 then
    Set fso2 = CreateObject("Scripting.FileSystemObject")
    if not fso2.FolderExists (arg & "\\pwpid\projects\mo") then
      fso2.createFolder (arg & "\\pwpid\projects\mo")
    end if
    fldr.copy (arg & "\\pwpid\projects\mo")

    ' pwlocal=arg
  end if
  if i = 3 then

    Set shell = CreateObject("Wscript.Shell")

    shell.run(arg & "ustation.exe -wc" & workdir & "\\cfgvardir\config\mslocal-ctr.cfg")
  end if
  i =i +1
next

```

Site Configurations

Site or standard configurations are variables pointing to resources that everyone needs. Put these in a `usersite.cfg` file and include this file in the `mslocal.cfg` file. Examples of the files contents include variables defining common cell libraries, mdl applications, macros, and font resource files.

Avoid creating user configuration (*.ucf) files that contain variable definitions that everyone needs. Especially try to avoid multiple ucf files. Additional files leads to less control of standards.

If you find yourself copying the `default.ucf` file to everybody's user login directory, stop! The user configuration file will be different for everyone and will be modified by the user. Take all of the variables defined in `default.ucf` and put them in `usersite.cfg`.

Minimize

Clean up doesn't happen as often as it should. A package consisting of fonts, cell libraries, applications, line style resources, and databases can grow quite large. So it is very important to do some house keeping.

Often when administering a workspace, zip archives, and backup directories are created. The first step then is to remove all of these back up and temporary files. Next step, compress all cell libraries and seed files.

It happens all of the time. An old design file has all of the dimension settings, text settings, and working units defined the way we like them to be defined. So, instead of starting with a new file we open it up and delete all of the elements from this file and viola! We have a seed file. Hmm, did I miss something? That's right, we forgot to compress.

By the way, that 75-megabyte workspace compressed down to 30 megabytes by the time we were finished with it.

Take Your Time

It is easy to slap together a workspace that can serve up your libraries and resources, but creating a lean manageable workspace requires considerable thought. I don't mean to suggest that you should wait until you think a workspace is complete to deploy it.

In fact, a small workspace to distribute cells, fonts, and seed files is better than working without a workspace. Over time you may find a better way to define a series of variables, or you may find that smaller discipline centric cell libraries are better than a couple of large cell libraries. The evolution of your workspace will happen, but not overnight.

Use a Self-Extracting Zip

Here's one more idea. The easiest way to deliver your workspace is with a self extracting zip file. No need to create an elaborate install script, Winzip will do just fine.

Set the extraction directory to `c:\temp\`. It is an easy directory to find, most people have one and your consultants, I mean consultants don't have to think very hard when installing the workspace into a different directory.

Tell your consultants how large the file is zipped and unzipped. And tell them which variables will need to be changed if they decide not to use the default `c:\temp\` installation directory.

Honest, we created a packaged workspace and the only definitions we had to make ourselves were the target and 'start in' fields on the desktop icon. And, if the consultants decide to take the c:\temp\ default, which of course they never will, we provided an icon definition that they could copy to their desktop.

Workspaces can get seriously complicated, but they should never be difficult to manage. So, take your time, minimize and before you know it you will have a clean, mean workspace machine.

About The Author

Mark Stefanchuk is a partner with Ramsey Systems, Inc., the developers of cadgurus.com. Mark can be contacted by email on mark@cadgurus.com.

Please email Mark with any feedback or suggestions for future articles.

CAD