AUTODESK UNIVERSITY

Programming The Work Out of CAD Management

Chris Lindner

onebutton cad solutions

About me

Chris Lindner

From Central OH, married 34 years, two adult children.

AutoCAD user since 1985 (AutoCAD 2.1)

AUGI Board of Directors

Freelance consultant www.onebuttoncad.com

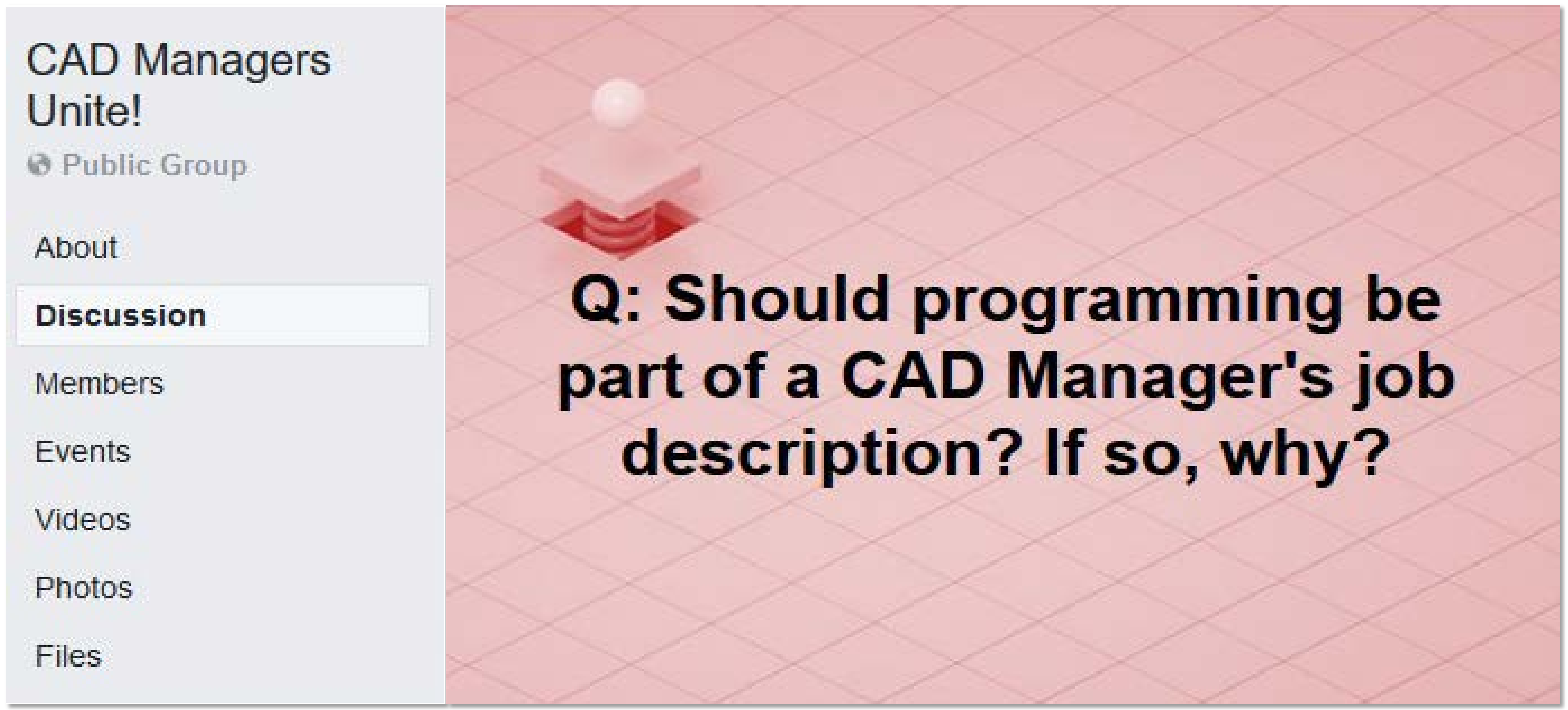
AutoLISP "programmer"

About you

CAD Manager Profiles

- **Roles**
  - Full-time
  - Part-time
  - Spare-time
- **Programming skills**
  - Basics helpful

Programming the Work out of CAD Management



Q: Should programming be part of a CAD Manager's job description? If so, why?

Programming the Work out of CAD Management

Four Main Categories for Programming

- Configuration
- Standardization
- Customization
- Automation

Programming the Work out of CAD Management

Four Main Categories for Programming

- Configuration
- Standardization
- Customization
- Automation

Why Programming is Important for CAD Managers

- Efficiency is essential; automation is the key.
- AutoCAD is generic; your client's/company's needs are not.
- AutoCAD is open architecture; leverage it.

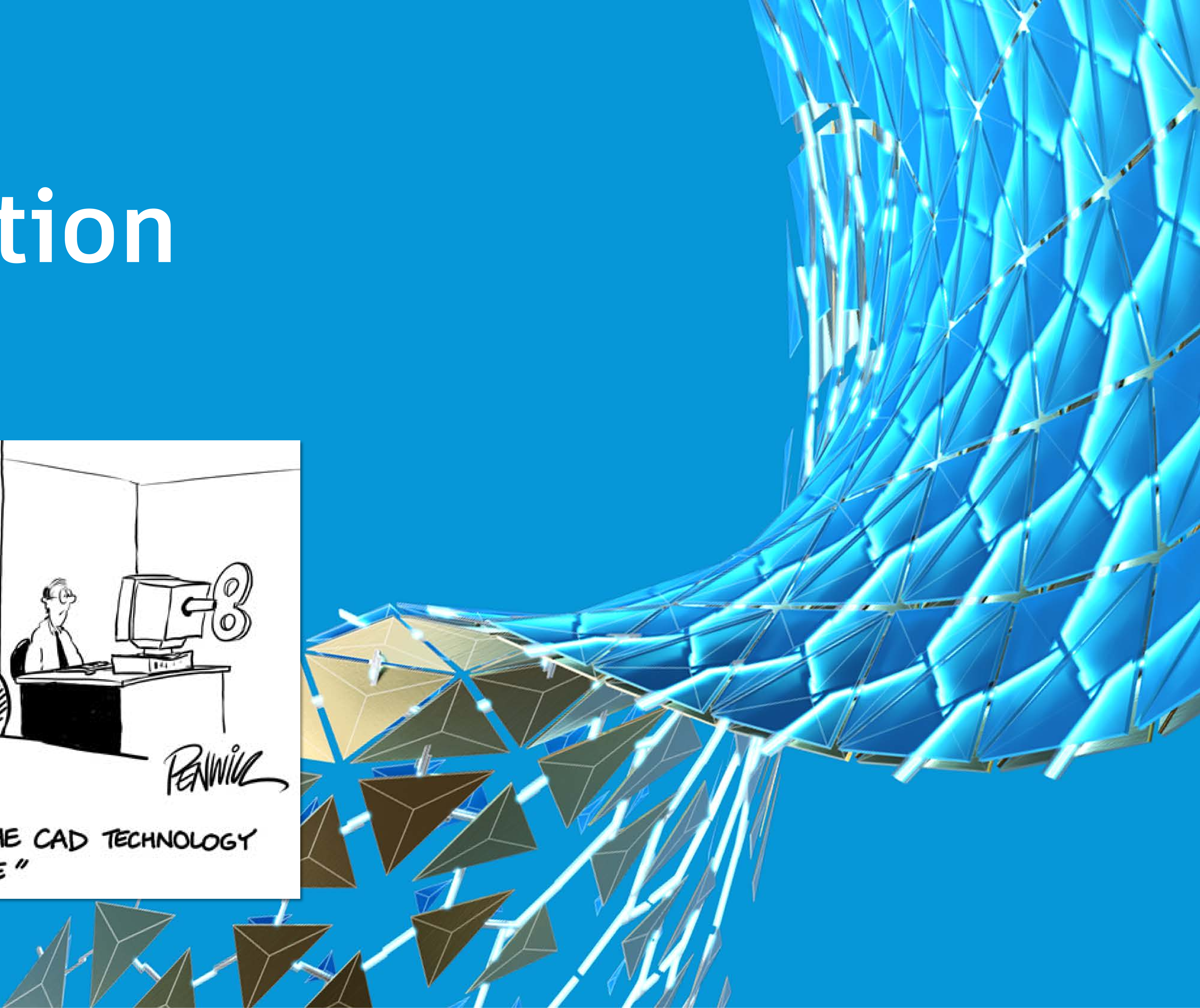
Programming the Work out of CAD Management

Learning Objectives

- **Overcome AutoCAD's configurations limitations** with some simple AutoLISP code.
- Leverage AutoLISP to **manage and maintain your company's standards**
- Explore the vital way that **standards enable automation** via AutoLISP
- **Create adaptable AutoLISP code** that reacts to project, client, or user preferences intelligently

Configuration

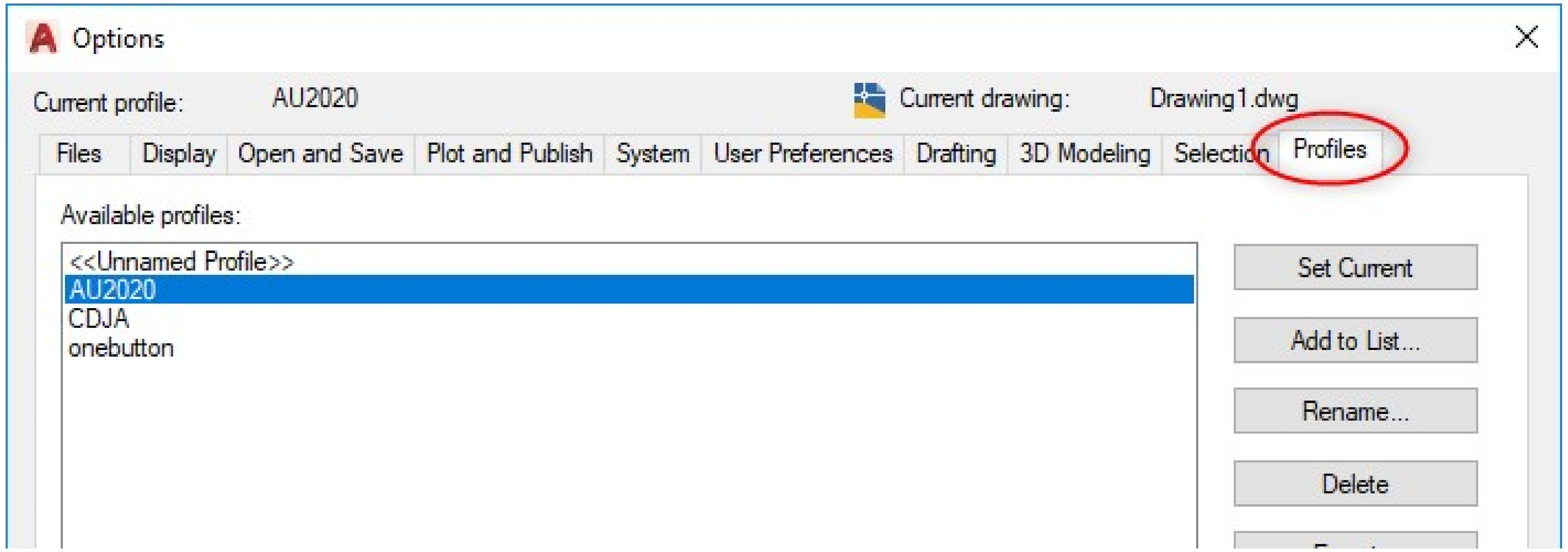
PENWILL
"WE MUST GET THE MOST FROM THE CAD TECHNOLOGY WE ALREADY HAVE"



Programming the Work out of: Configuration

AutoCAD Profiles

- A way to **"*store program settings*** *for different users or projects*"
- **Profiles** tab in the Options dialog



Available profiles:

<<Unnamed Profile>>
AU2020
CDJA
onebutton

Programming the Work out of: Configuration

AutoCAD Profiles

- A way to **"*store program settings*** *for different users or projects*"
- **Profiles** tab in the Options dialog

Profile Limitations

- Fragile – missing paths are removed
- Local – once loaded, all settings are stored locally
- Inflexible – don’t accommodate both company and personal settings
- Vulnerable – users can freely "tinker"

Programming the Work out of: Configuration

Key Files

ACAD.LSP	– automatically loads each time AutoCAD starts
ACADDOC.LSP	– automatically loads each time a drawing is opened
<cui name>.MNL	– automatically loads when <cui name>.CUI is loaded

For This Class

Only use the ACAD.LSP file (ACADLSPASDOC = 1).

Programming the Work out of: Configuration

Delayed Execution

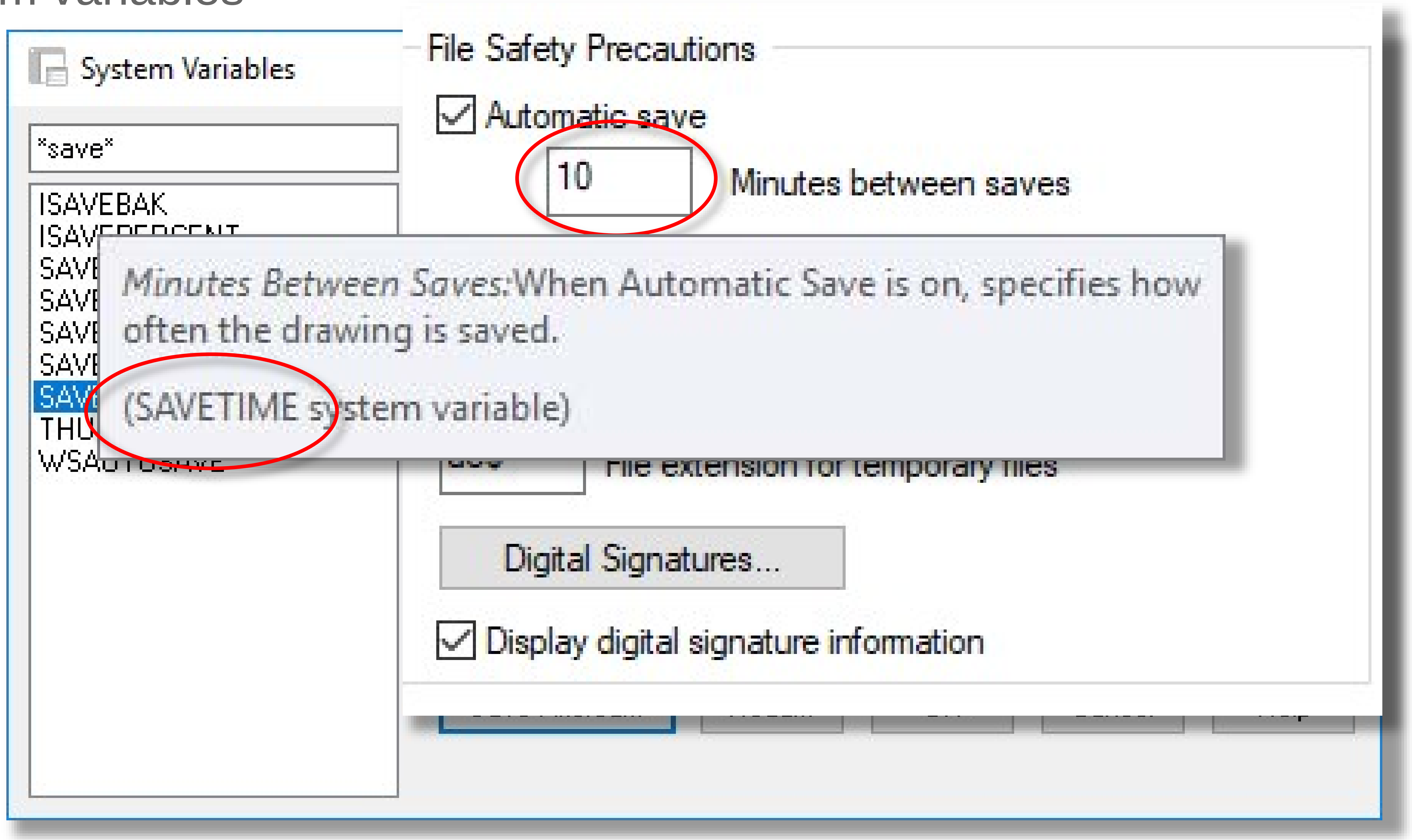
- Code loads before drawing fully open
- User-defined S::Startup function - delays code execution until after drawing is fully opened

```
(defun S::Startup ()
…<code to be executed after initialization>…
)
```

Programming the Work out of: Configuration

System Variables

- They store "*information about the current drawing or program configuration*"
- Many settings found in the Options dialog box can be controlled via AutoLISP
- Use Express Tool's SYSVDLG to explore all system variables


System Variables

save

ISAVEBAK
ISAVEPERCENT
SAVE
SAVE
SAVE
SAVE
SAV
THU
WSAUTOSAVE

File Safety Precautions

✓ Automatic save

10 Minutes between saves

*Minutes Between Saves:*When Automatic Save is on, specifies how often the drawing is saved.

(SAVETIME system variable)

File extension for temporary files

Digital Signatures...

✓ Display digital signature information

Programming the Work out of: Configuration

Programming SysVars

- (getvar) function

```
(getvar <variable name>)
```

- (setvar) function

```
(setvar <variable name> <value>)
```

Example:

```
(setvar "SAVETIME" 15)
```

Programming the Work out of: Configuration

SysVars Examples

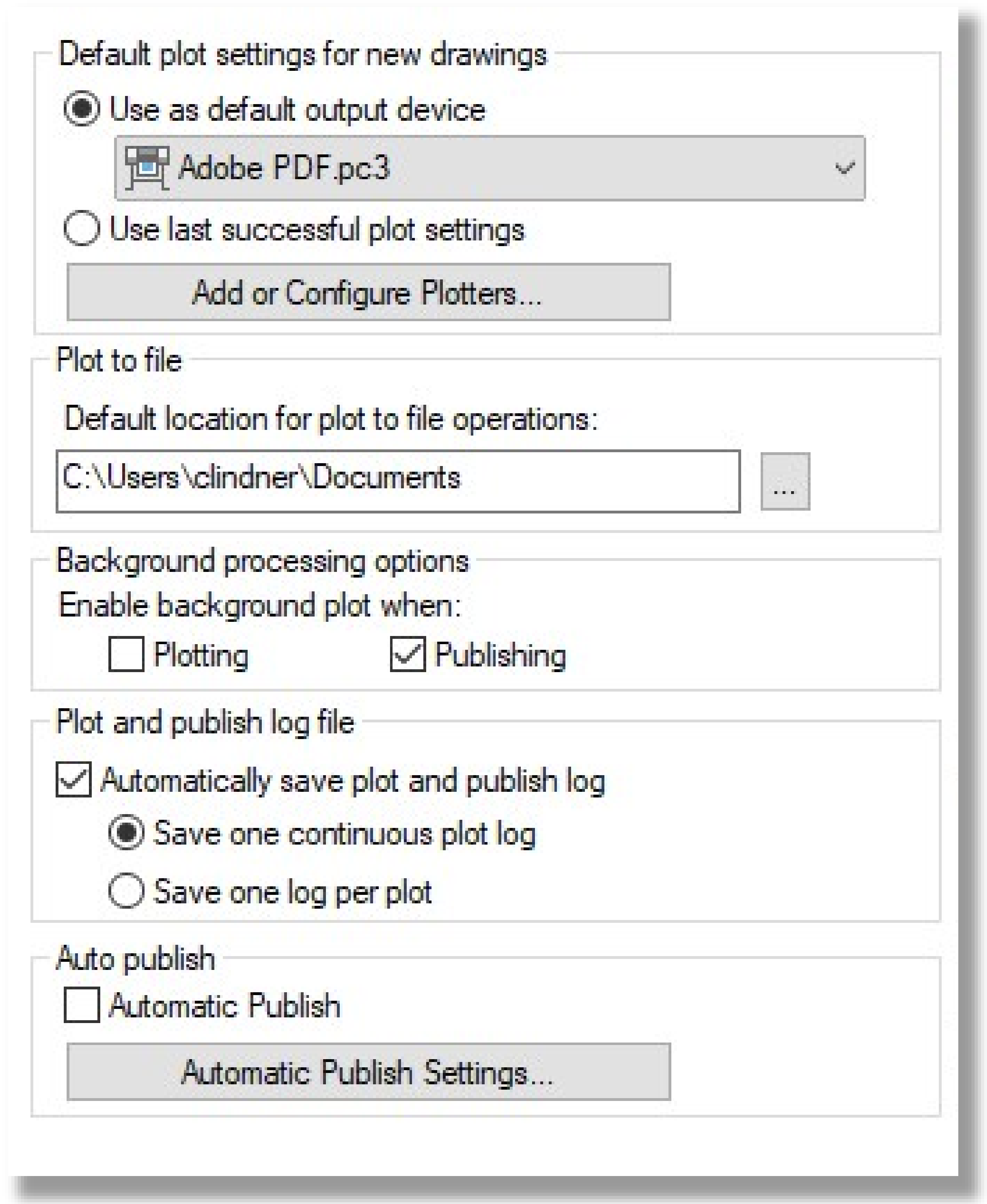
```
(setvar "FILEDIA" 1)                ; display dialog boxes
(setvar "CONSTRAINTINFER" 0)        ; turn off inferred constraints
(setvar "OBJECTISOLATIONMODE" 0); turn isolated objects back on
(setvar "XREFTYPE" 1)               ; default to "overlay" xrefs

(setvar "CECOLOR" "Bylayer")        ; ensure current color is ByLayer
(setvar "CELTYPE" "Bylayer")        ; ensure current linetype is ByLayer
(setvar "HPLAYER" ".")              ; force hatch to use current layer
(setvar "LUNITS" 4)                 ; default to Arch units
```

FIGURE 2 SYSTEM VARIABLE EXAMPLES

Programming the Work out of: Configuration

Environment Variables

- *"operating system environment variables*"
- Many of the settings found in the Options dialog box can be controlled via AutoLISP


Default plot settings for new drawings

● Use as default output device

Adobe PDF.pc3

○ Use last successful plot settings

Plot to file

Default location for plot to file operations:

C:\Users\clindner\Documents

Background processing options

Enable background plot when:

☐ Plotting ☑ Publishing

Plot and publish log file

☑ Automatically save plot and publish log

● Save one continuous plot log

○ Save one log per plot

Auto publish

☐ Automatic Publish

Programming the Work out of: Configuration

Programming EnvVars

- (getenv) function

```
(getenv <variable name>)
```

- (setenv) function

```
(setenv <variable name> <value>)
```

Example:

```
(setenv "AutomaticSaveMinutes" "15")
```

Programming the Work out of: Configuration

EnvVar Examples

```
(setenv "HideSystemPrinters" "1")      ; hide system printers
(setenv "DefaultFormatForSave" "60")   ; force saving to 2013 file format
(setenv "ShowTabs" "1")                ; show layout tabs
(setenv "ShowFullPathInTitle" "1")     ; show full drawing path in title bar
```

Figure 3 Environment Variable Samples

Programming the Work out of: Configuration

Environment Variable Differences

- Pertain to the system or the AutoCAD application (not to the drawing)

Programming the Work out of: Configuration

Environment Variable Differences

- Pertain to the system or the AutoCAD application (not to the drawing)
- Case sensitive

```
(setenv "AutomaticSaveMinutes" "15") ✓
(setenv "automaticsaveminutes" "15") ✗
```

Programming the Work out of: Configuration

Environment Variable Differences

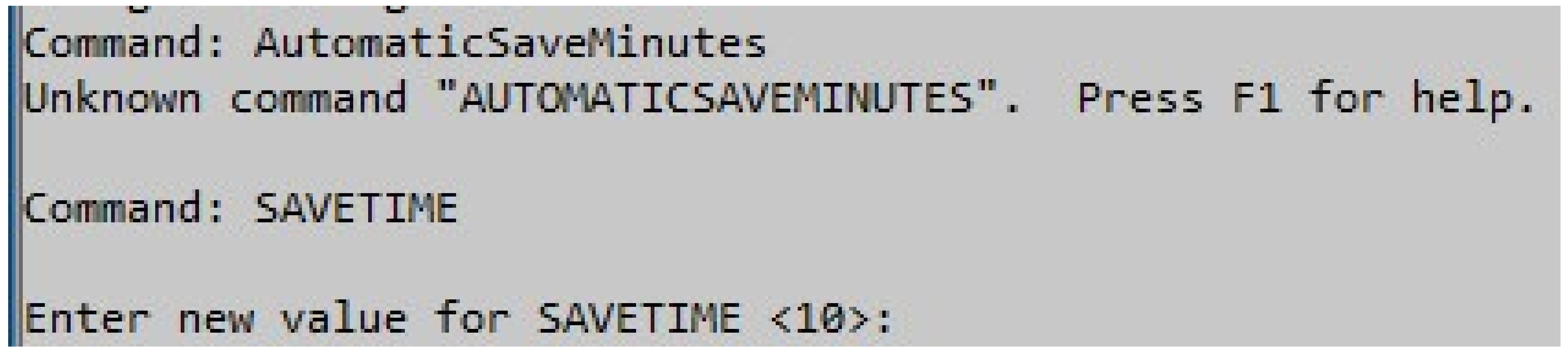
- Pertain to the system or the AutoCAD application (not to the drawing)
- Case sensitive
- Accepts string values only

```
(setenv "AutomaticSaveMinutes" "15") ✓
(setenv "AutomaticSaveMinutes" 15)   ✗
```

Programming the Work out of: Configuration

Environment Variable Differences

- Pertain to the system or the AutoCAD application (not to the drawing)
- Case sensitive
- Accepts string values only
- Can’t be typed on the command line



```
Command: AutomaticSaveMinutes
Unknown command "AUTOMATICSAVEMINUTES".  Press F1 for help.

Command: SAVETIME

Enter new value for SAVETIME <10>:
```

Programming the Work out of: Configuration

Environment Variable Differences

- Pertain to the system or the AutoCAD application (not to the drawing)
- Case sensitive
- Accepts string values only
- Can’t be typed on the command line
- Stored in the registry

Programming the Work out of: Configuration

The Same But Different

- Both of these examples work the same, even though the variable names are different:

System variable:
```
(setvar "savetime"    10)
```
Environment variable:
```
(setenv "AutomaticSaveMinutes" "10")
```

Programming the Work out of: Configuration

The Same But Different

- Both of these examples work the same, even though the variable names are different:

System variable:
```
(setvar "savetime"     10)
```
Environment variable:
```
(setenv "AutomaticSaveMinutes" "10")
```

- The system variable example below will return an error (it's read-only) …

```
(setvar "TempPrefix"     "C:\\Temp")
```
✗

…but its environment variable counterpart will work fine.

```
(setenv "TempDirectory" "C:\\Temp")
```
✓

Programming the Work out of: Configuration

Going Beyond Vanilla AutoLISP

- Visual LISP - beyond vanilla AutoLISP

Programming the Work out of: Configuration

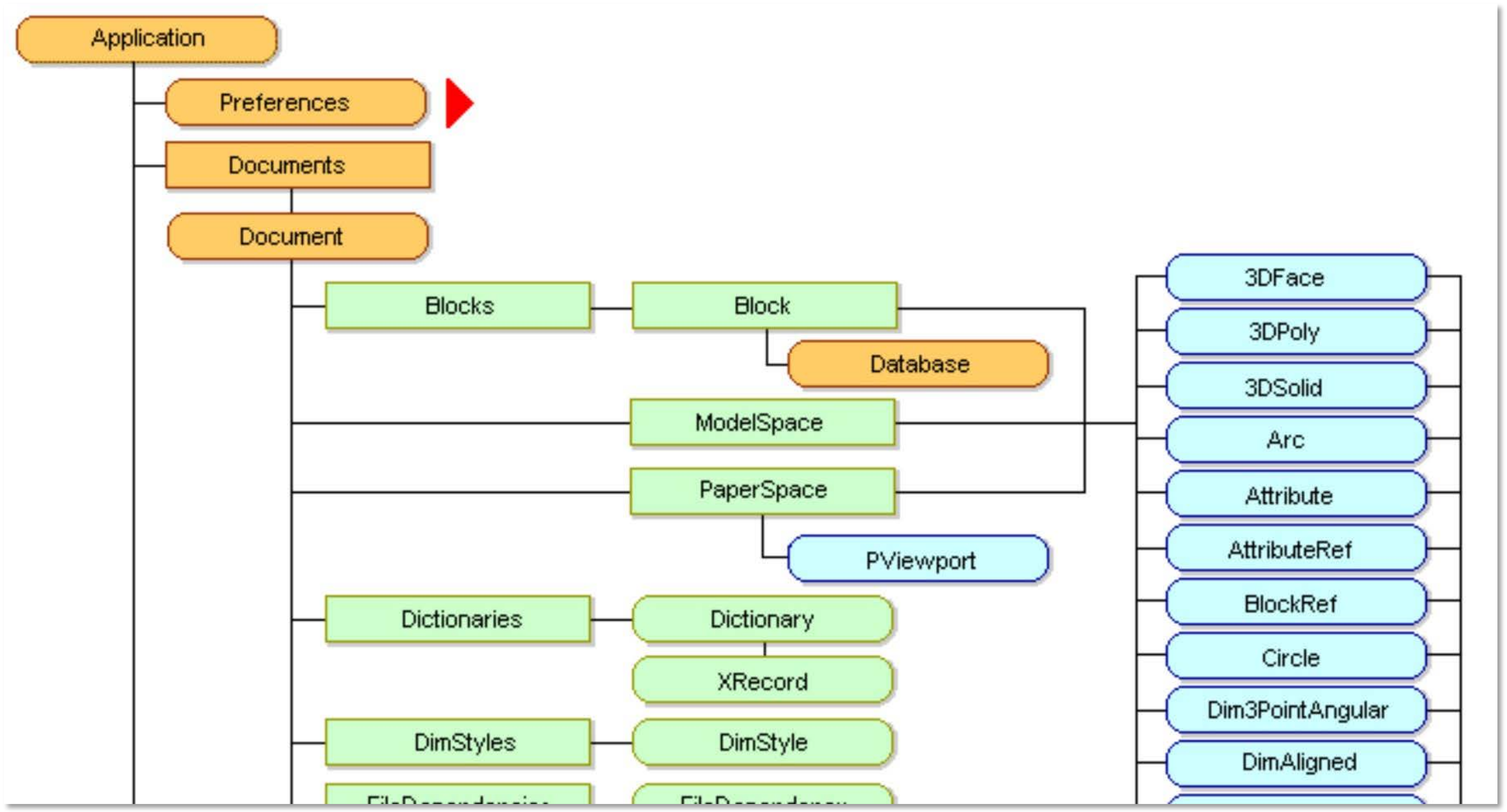
Going Beyond Vanilla AutoLISP

- Visual LISP - beyond vanilla AutoLISP
- *“an **extension** of the AutoLISP programming language”*

Programming the Work out of: Configuration

Going Beyond Vanilla AutoLISP

- Visual LISP - beyond vanilla AutoLISP
- *“an **extension** of the AutoLISP programming language”*
- Provides access to the "AutoCAD Object Model"



Programming the Work out of: Configuration

Going Beyond Vanilla AutoLISP

- Visual LISP - beyond vanilla AutoLISP
- *“an **extension** of the AutoLISP programming language”*
- Provides access to the "AutoCAD Object Model"
- Code to get it started:

```
(vl-load-com)
(setq *acad*  (vlax-get-acad-object))          ; ACAD application
(setq *opts*  (vla-get-preferences *acad*))    ; Options dialog
(setq *files* (vla-get-files *opts*))          ; Files tab
(setq *doc*   (vla-get-activedocument *acad*)); current drawing
```

Programming the Work out of: Configuration

Programming Paths via VLISP

- (vla-get-<path>) function

```
(vla-get-<variable name> <files object>)
```

- (vla-put-<path>) function

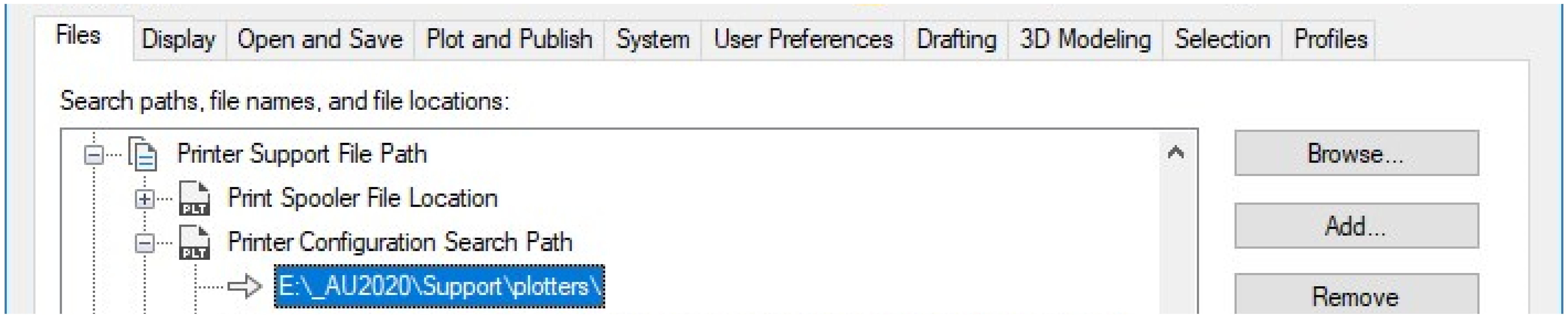
```
(vla-put-<variable name> <files object> <value>)
```

Programming the Work out of: Configuration

VLISP Path Examples

```
(vla-get-PrinterConfigPath *files*)
```

```
(vla-put-PrinterConfigPath *files* "E:\\_AU2020\\Support\\Plotters")
```

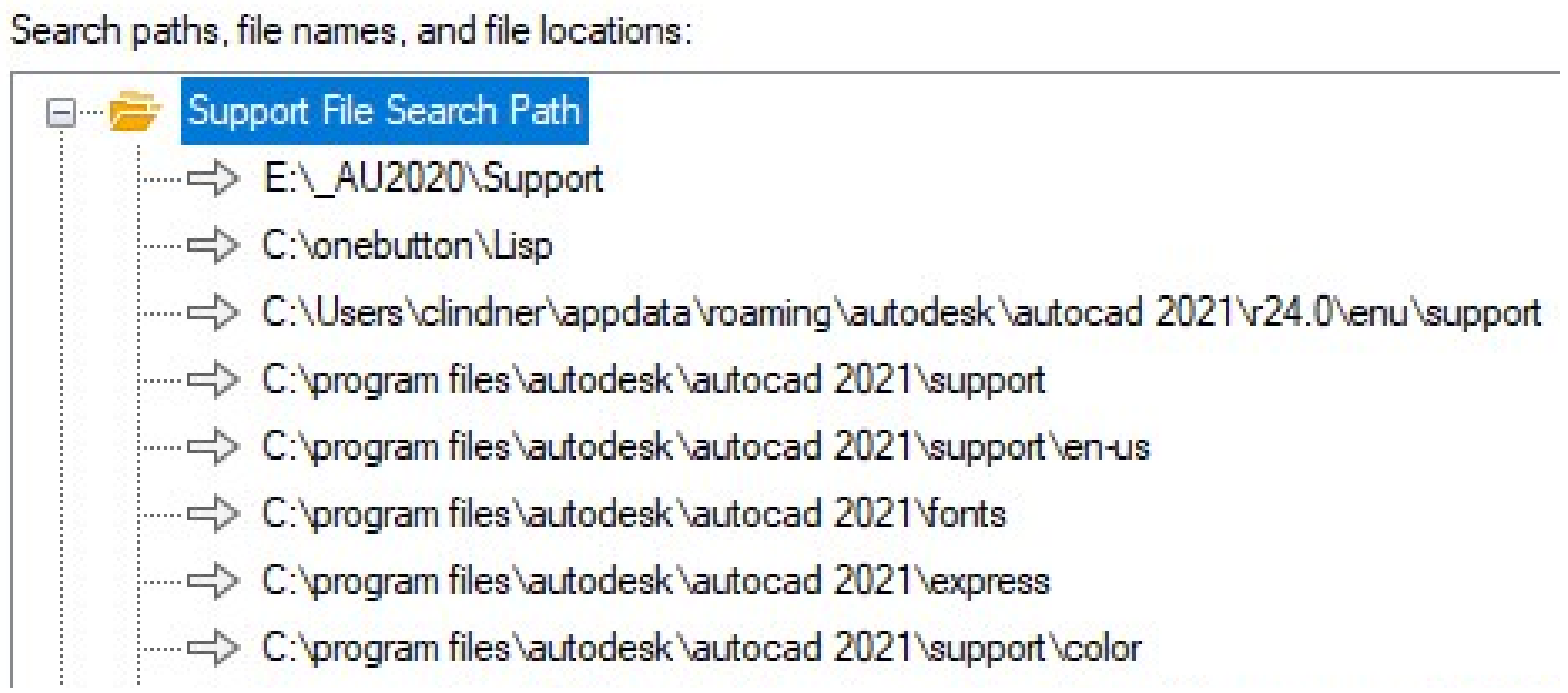


Search paths, file names, and file locations:

Printer Support File Path
Print Spooler File Location
Printer Configuration Search Path
E:_AU2020\Support\plotters\

Programming the Work out of: Configuration

```
(defun SearchPathFix ()
  ;; Set standard Support File Search Paths
  (vla-put-SupportPath
    *files*
    (strcat
      "E:\\_AU2020\\Support;"
      "C:\\onebutton\\Lisp;"
      "C:\\users\\clindner\\appdata\\roaming\\autodesk\\autocad 2021\\r24.0\\enu\\support;"
      “C:\\program files\\autodesk\\autocad 2021\\support;"
      “C:\\program files\\autodesk\\autocad 2021\\support\\en-us;"
      “C:\\program files\\autodesk\\autocad 2021\\fonts;"
      “C:\\program files\\autodesk\\autocad 2021\\help;"
      “C:\\program files\\autodesk\\autocad 2021\\express;"
      “C:\\program files\\autodesk\\autocad 2021\\support\\color;"
     ) ;_ end of strcat
  ) ;_ end of vla-put-SupportPath
  ;; Set trusted paths
  (setvar "trustedpaths"
           (strcat
             "E:\\_AU2020\\Support;"
             "C:\\onebutton\\Lisp;"
           ) ;_ end of strcat
  ) ;_ end of setvar
) ;_ end of defun
```


Search paths, file names, and file locations:

Support File Search Path
- E:_AU2020\Support
- C:\onebutton\Lisp
- C:\Users\clindner\appdata\roaming\autodesk\autocad 2021\r24.0\enu\support
- C:\program files\autodesk\autocad 2021\support
- C:\program files\autodesk\autocad 2021\support\en-us
- C:\program files\autodesk\autocad 2021\fonts
- C:\program files\autodesk\autocad 2021\express
- C:\program files\autodesk\autocad 2021\support\color

Standards

Programming the Work out of: Standards

Getting a Handle on Standards

- Dynamic not static
- Winning the implementation battle
  - "Pushed" standards
  - "Pulled" standards
  - Smart defaults
  - Adaptable code

IF YOU CAN MAKE IT EASIER FOR THEM TO DO IT RIGHT THAN IT IS TO DO IT WRONG, THEY'LL USUALLY DO IT RIGHT.

Programming the Work out of: Standards

Pushing Standards

```
(defun LockColmLayers (/ layer)
  (foreach layer '("S-COLM" "S-COLM-IDEN" "S-COLM-GRID" "S-COLM-DIMS")
    (if (tblobjname "layer" layer)
      (vla-put-lock
        (vla-item (vla-get-layers *doc*) layer)
        :vlax-true
      ) ;_ end of vla-put-lock
    ) ;_ end of if
  ) ;_ end of foreach
) ;_ end of defun
```

FIGURE 7 AUTO-LOCKED LAYERS

Programming the Work out of: Standards

Pushing Standards

```
(defun LockVPorts (/ vplay layout ent)
  (setq vplay (vla-Add (vla-get-Layers *doc*) "0-VPRT")); create 0-VPRT layer
  (vla-put-plottable vplay :vlax-false)                 ; and set to no-plot
  ;; loop thru all layouts
  (vlax-for layout (vla-get-Layouts *doc*)
    (if (eq :vlax-false (vla-get-ModelType layout))     ; skip model space layout
      (vlax-for ent (vla-get-Block layout)              ; for each ent in layout
        (if (eq (vla-get-ObjectName ent) "AcDbViewport"); if entity is a viewport
         (progn
           (vla-put-DisplayLocked ent :vlax-true)       ; lock the viewport
           (vla-put-Layer ent "0-VPRT")                 ; assign vp to "0-VPRT"
         ))))) ;_ end of vlax-for
) ;_ end of defun
```

FIGURE 8 AUTO-LOCKED VIEWPORTS

Programming the Work out of: Standards

Pushing Standards

- Example: standards applied after drawing opened

```
(if (= 1 (getvar "DWGTITLED")) ; skip on unnamed drawings
  (progn
    (princ "\nLocking viewports...")
    (LockVPorts)
    (princ "\nLocking column layers...")
    (LockColmLayers)
  )
)
```

FIGURE 9 PUSHING STANDARDS

Programming the Work out of: Standards

Pulling Standards

- Example: custom commands for “pulling” standards

```
(defun C:LockColumns ()
  (LockColmLayers)
  (princ)
  )
```

FIGURE 10

```
(defun C:LockVPorts ()
  (LockVPorts)
  (princ)
  )
```

FIGURE 11

Programming the Work out of: Standards

Standards via Suggestion

- Example: current dimstyle “nudge”

```
(vla-put-activedimstyle
  *doc*
  (vla-item (vla-get-dimstyles *doc*)
            "Arch_Tick-Anno"
  ) ;_ end of vla-item
) ;_ end of vla-put-activedimstyle
```

FIGURE 12 SUGGESTED DEFAULTS

Programming the Work out of: Standards

Adaptive Code

```
(vla-put-activedimstyle
  *doc*
  (vla-item (vla-get-dimstyles *doc*)
            "Arch_Tick-Anno"
  ) ;_ end of vla-item
) ;_ end of vla-put-activedimstyle
```

FIGURE 12 INFLEXIBLE CODE

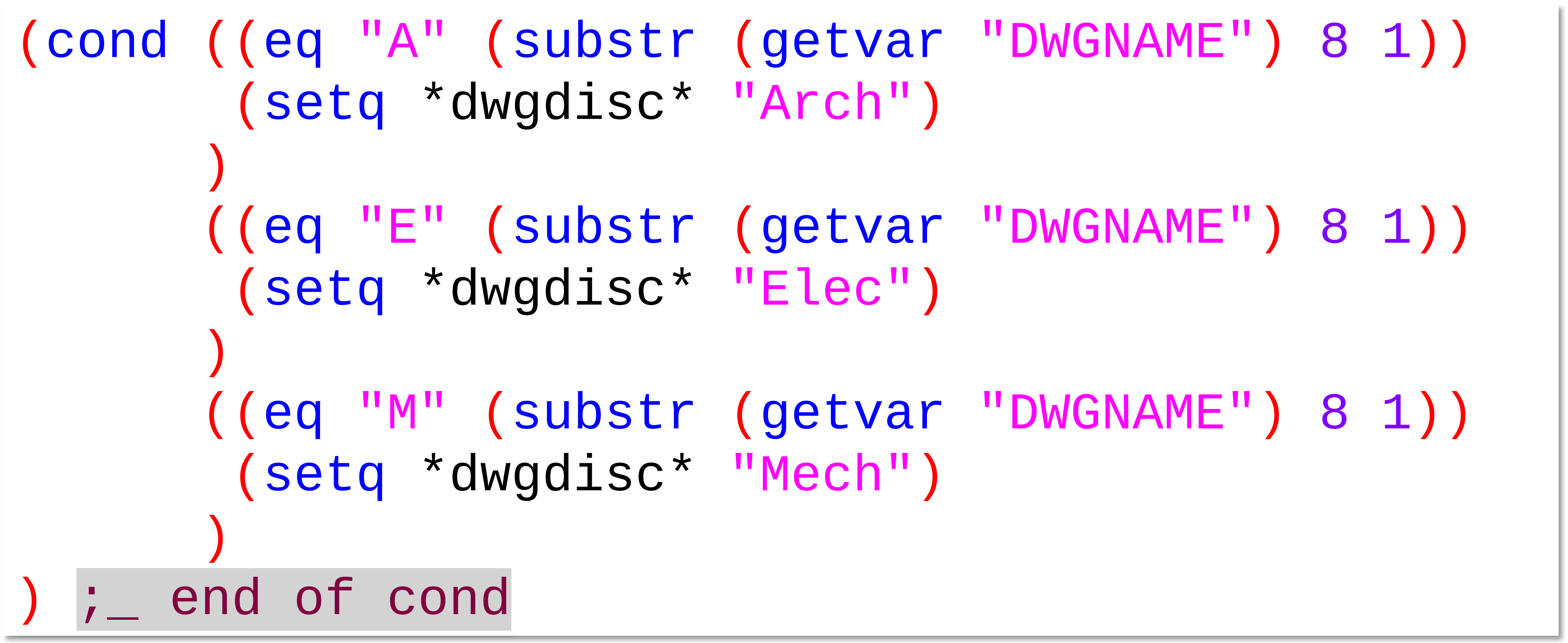
```
(if (tblsearch "DIMSTYLE" "Arch_Tick_Anno")
  (vla-put-activedimstyle
    *doc*
    (vla-item (vla-get-dimstyles *doc*)
              "Arch_Tick-Anno"
    ) ;_ end of vla-item
  ) ;_ end of vla-put-activedimstyle
) ;_ end of if
```

FIGURE 13 FLEXIBLE CODE

Programming the Work out of: Standards

Adaptive Standards

- Standard filename format: <project no>_<disc><sht #>.dwg
- Sample filename: 201905_A0201.dwg



```
(cond ((eq "A" (substr (getvar "DWGNAME") 8 1))
       (setq *dwgdisc* "Arch")
      )
      ((eq "E" (substr (getvar "DWGNAME") 8 1))
       (setq *dwgdisc* "Elec")
      )
      ((eq "M" (substr (getvar "DWGNAME") 8 1))
       (setq *dwgdisc* "Mech")
      )
) ;_ end of cond
```

FIGURE 14 FILENAME-BASED DISCIPLINES

#programmingCM

Programming the Work out of: Standards

Adaptive Standards

```
(setvar "*_TOOLPALETTEPATH"
        (cond ((eq "Mech" *dwgdisc*) "N:\\Palettes\\Mech\\")
              ((eq "Elec" *dwgdisc*) "N:\\Palettes\\Elec\\")
              (t
               "N:\\Palettes\\Arch\\"
              )
        ) ;_ end of cond
) ;_ end of setvar
```

FIGURE 15 LEVERAGING FILENAME-BASED DISCIPLINES

"THE FOUNDATION FOR AUTOMATION IS STANDARDIZATION"

Programming the Work out of: Standards

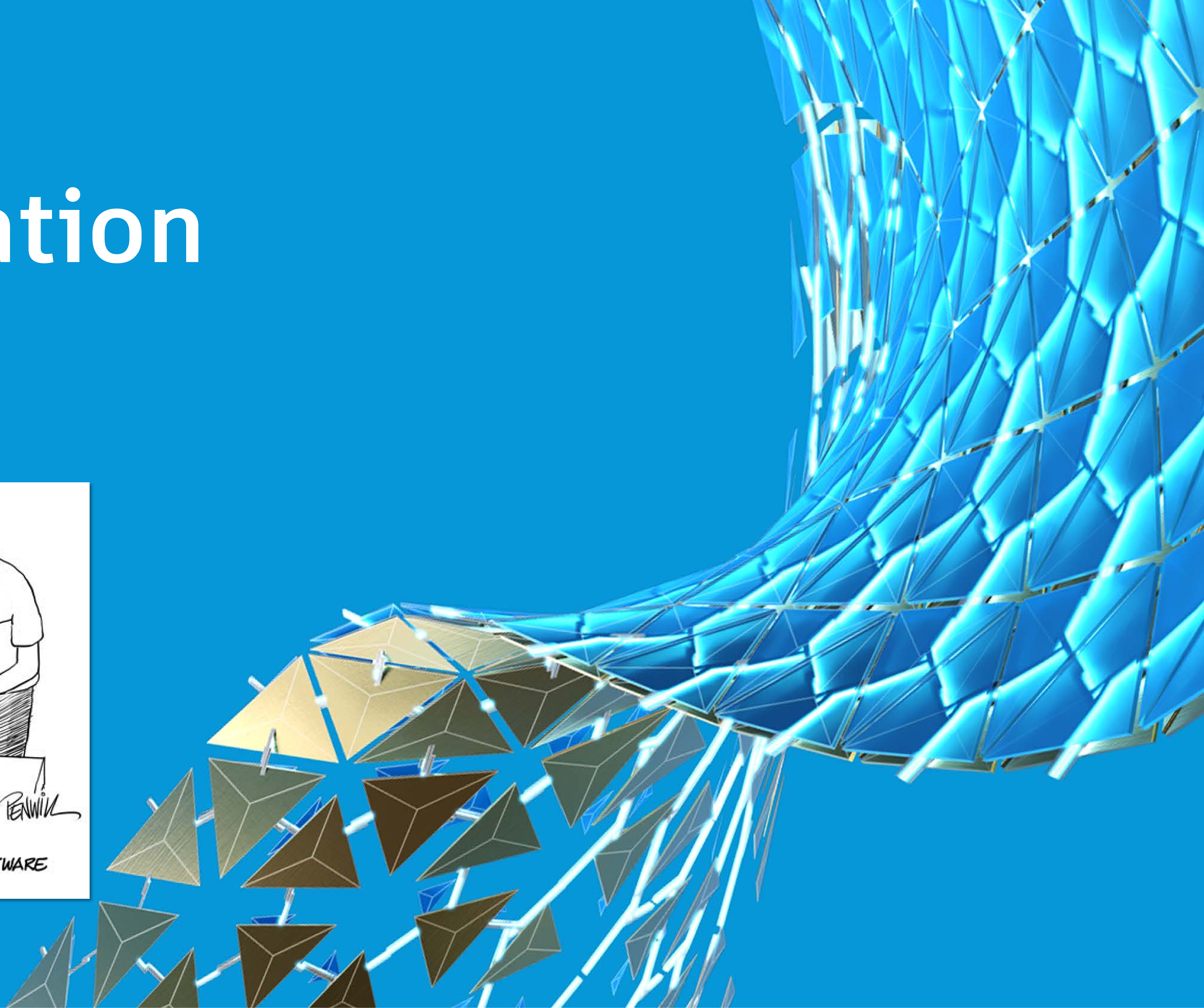
Flexibility within Constraints

```
(if (setq usr (findfile "C:\\ACAD\\My ACAD 2021\\Personal.lsp"))
  (progn
    (princ (strcat "\nLoading personal code for \""
                   (getvar "LOGINNAME")
                   "\"..."
           ) ;_ end of strcat
    ) ;_ end of princ
    (load usr)
  ) ;_ end of progn
) ;_ end of if
```

FIGURE 16 LOADING USER AUTOLISP CODE

Customization

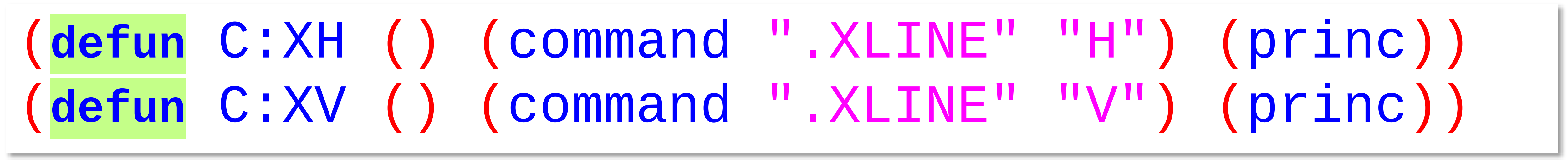
ZAPCAD
PENWILL
"SURELY THERE'S MORE TO CUSTOMISING THE SOFTWARE THAN JUST DECORATING THE BOX"



Programming the Work out of: Customization

Simple New Commands

- Beyond the ACAD.PGP
- Example: Shortcuts for creating vertical or horizontal xlines



```
(defun C:XH () (command ".XLINE" "H") (princ))
(defun C:XV () (command ".XLINE" "V") (princ))
```

FIGURE 18 SIMPLE CUSTOM COMMANDS

Programming the Work out of: Customization

Expanding New Commands

- Example: shortcut to bring objects to the front

```
(defun C:BF (/ SS)
  (if (not (setq SS (ssget "_i")))
    (progn
      (princ "\nSelect object(s) to bring to front: ")
      (setq SS (ssget))
      (if SS
        (command "DRAWORDER" SS "" "F")
        (command)
      ) ;_ end of if
    ) ;_ end of progn
    (command "DRAWORDER" "F")
  ) ;_ end of if
  (princ)
) ;_ end of defun
```

FIGURE 19 ENHANCING EXISTING COMMANDS

Programming the Work out of: Customization

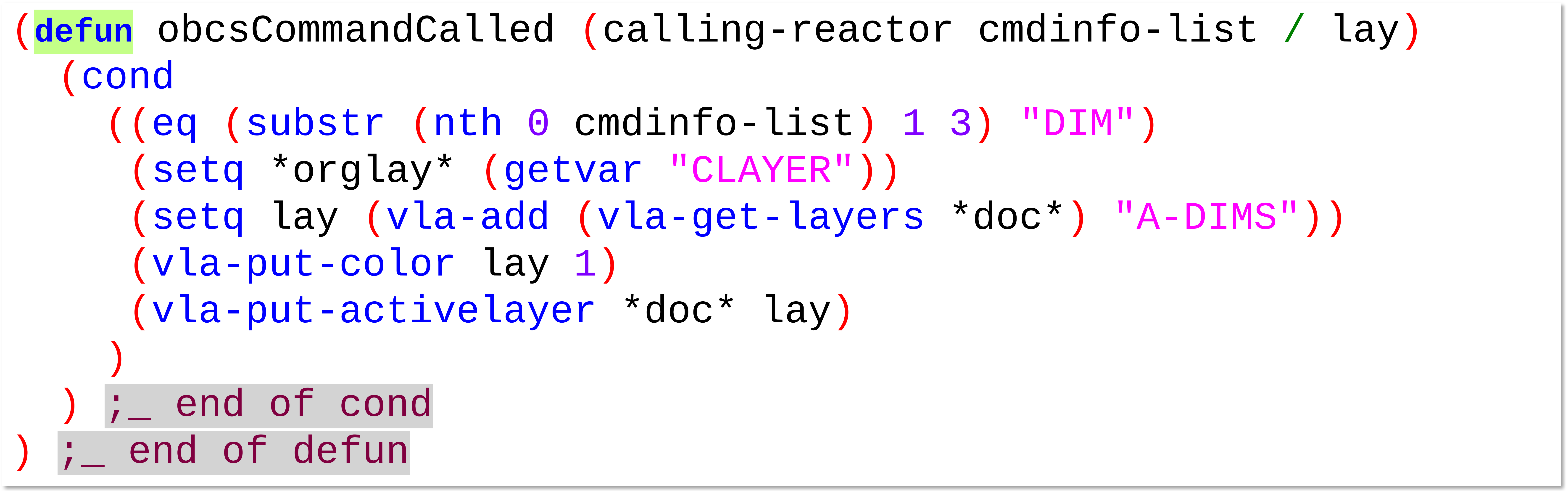
Hijacking Existing Commands

- AutoCAD Reactors
- “*respond to one or more AutoCAD events*”
  - When a command starts, ends, is cancelled
  - When an object in the drawing is changed, copied, or deleted
  - When then drawing is saved or a variable is changed

Programming the Work out of: Customization

Defining Command Reactors

- Example: Set “A-DIMS” layer current for all dimension commands



```
(defun obcsCommandCalled (calling-reactor cmdinfo-list / lay)
  (cond
    ((eq (substr (nth 0 cmdinfo-list) 1 3) "DIM")
     (setq *orglay* (getvar "CLAYER"))
     (setq lay (vla-add (vla-get-layers *doc*) "A-DIMS"))
     (vla-put-color lay 1)
     (vla-put-activelayer *doc* lay)
    )
  ) ;_ end of cond
) ;_ end of defun
```

FIGURE 20 RUNS WHEN A COMMAND IS CALLED

Programming the Work out of: Customization

Defining Command Reactors

```
(defun obcsCommandEnded (calling-reactor cmdinfo-list /)
  (cond
    ((eq (substr (nth 0 cmdinfo-list) 1 3) "DIM")
     (vla-put-activelayer
       *doc*
       (vla-item (vla-get-layers *doc*) *orglyr*)
     ) ;_ end of vla-put-activelayer
    )
  ) ;_ end of cond
) ;_ end of defun
```

FIGURE 21 RUNS AFTER COMMAND ENDS

Programming the Work out of: Customization

Assigning Reactors

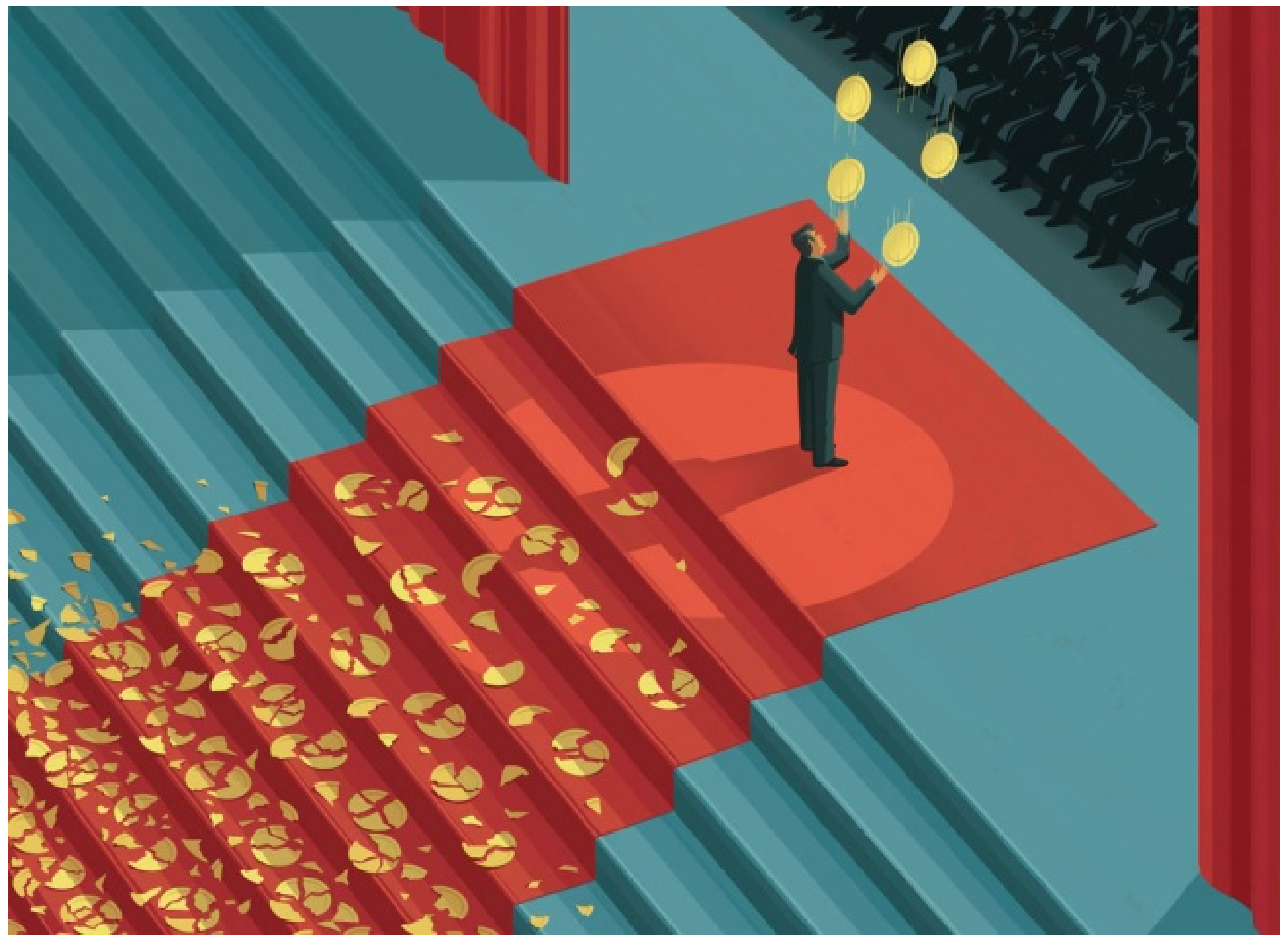
```
(vlr-editor-reactor nil '((:vlr-commandwillstart . obcsCommandCalled)
                          (:vlr-commandended     . obcsCommandEnded))
```

FIGURE 22 ASSIGNING REACTORS

Let the adventure begin…

- Embrace failure. Fail fast. Fail often.
- www.onebuttoncad.com/programmingCM – add’l information, links and code.

- Contact:
  - Email: chris@onebuttoncad.com
  - LinkedIn: cslindner
  - Twitter: chrislindner
  - Hashtag: #programmingCM



Thanks for Attending! *Please do the survey.*