AutoCAD automation through Python scripting

Let Python do the job for you

34th INGEGRAF

Antonio Fernández[1], Lucía Díaz Vilariño[1], Francesco Bianconi[2]

[1]Universidade de Vigo, [2]Università degli Studi di Perugia

antfdez@uvigo.es antfdez.webs.uvigo.es tonechas

Outline

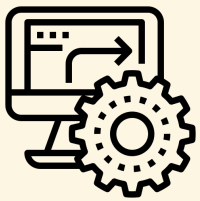
1. Introduction
2. Method
3. Results
4. Discussion

Motivation

Why automate tasks in design workflows?

- Increase efficiency
- Improve accuracy
- Ensure consistency

AutoCAD



What can be automated?

- Adding, removing, and editing elements
- Layer management and configuration
- Import/export from/to other formats
- Block creation and handling

Automation via AutoLISP

Pros

- Native to AutoCAD (since the 1980s)
- Lightweight and fast for small tasks

Cons

- Difficult to read and maintain
- Not ideal for complex workflows

Automation via AutoLISP

Pros

- Native to AutoCAD (since the 1980s)
- Lightweight and fast for small tasks

Cons

- Difficult to read and maintain
- Not ideal for complex workflows

```
(defun c:xlay ()
  (vl-load-com)
  (setq ms (vla-get-ModelSpace
             (vla-get-ActiveDocument
               (vlax-get-acad-object))))
  (vlax-for ent ms
    (if (/= (vla-get-Layer ent) "MyLayer")
      (vla-put-Layer ent "MyLayer")))
  (princ)
)
```

Automation via VBA and ActiveX

Pros

- Provides structured access to AutoCAD objects
- Integrates with other Windows apps

Cons

- Limited scalability and integration
- Deprecated and no longer maintained

Automation via VBA and ActiveX

Pros

- Provides structured access to AutoCAD objects
- Integrates with other Windows apps

Cons

- Limited scalability and integration
- Deprecated and no longer maintained


```
Sub SetLayer()
  Dim ent As AcadEntity
  For Each ent In ThisDrawing.ModelSpace
    If ent.Layer <> "MyLayer" Then
      ent.Layer = "MyLayer"
    End If
  Next
End Sub
```

Automation via C++ and ObjectARX

Pros

- Full access to AutoCAD API and internals
- High performance and scalability

Cons

- Requires advanced C++ programming skills
- Longer development and debugging time

Automation via C++ and ObjectARX

Pros

- Full access to AutoCAD API and internals
- High performance and scalability

Cons

- Requires advanced C++ programming skills
- Longer development and debugging time

```
void SetLayer(AcDbDatabase* db) {
  AcDbBlockTableRecord* ms;
  db->getBlockTableRecord(
    ACDB_MODEL_SPACE,
    ms, AcDb::kForWrite);
  AcDbEntity* ent;
  for (auto it = ms->begin();
       it != ms->end(); ++it) {
    ms->getAt(*it, ent, AcDb::kForWrite);
    if (ent->layer() != "MyLayer")
      ent->setLayer("MyLayer");
    ent->close();
  }
  ms->close();
}
```

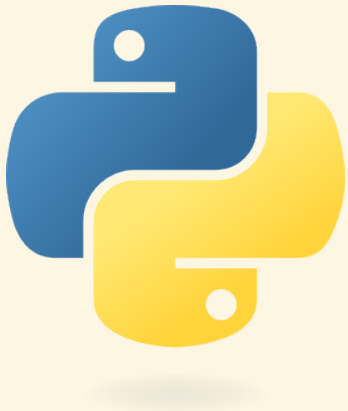
Automation via Python scripting

Pros

- Clear and readable syntax
- Rich ecosystem of libraries
- Suitable for data and AI tasks
- Cross-platform compatibility

Cons

- No direct native API
- Limited access to advanced features
- Needs external package



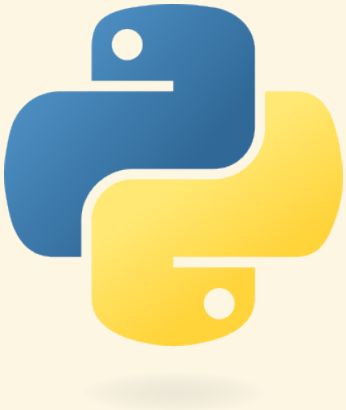
Automation via Python scripting

Pros

- Clear and readable syntax
- Rich ecosystem of libraries
- Suitable for data and AI tasks
- Cross-platform compatibility

Cons

- No direct native API
- Limited access to advanced features
- Needs external package



```python
from pyautocad import Autocad

acad = Autocad()
for obj in acad.iter_objects():
    if obj.Layer != "MyLayer":
        obj.Layer = "MyLayer"
```

Research question

Common packages

- `win32com`
- `pyautocad`
- `ezdxf`
- `pyautogui`

What is the most convenient package?

Moving on to...

1. Introduction

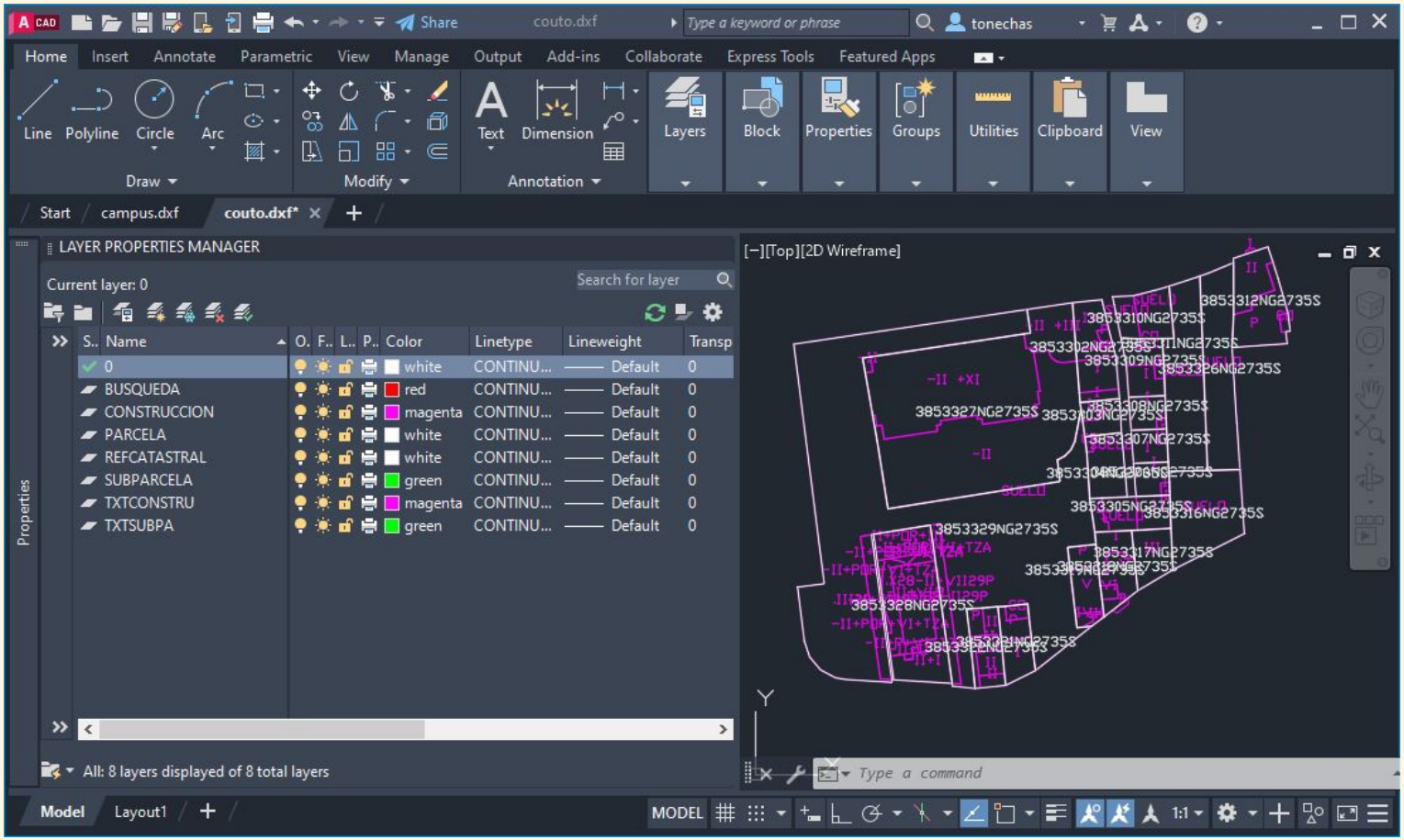
2. Method

3. Results

4. Discussion

Case study

Add the prefix CADASTRE_ to all layer names in a drawing downloaded from the Spanish cadastre



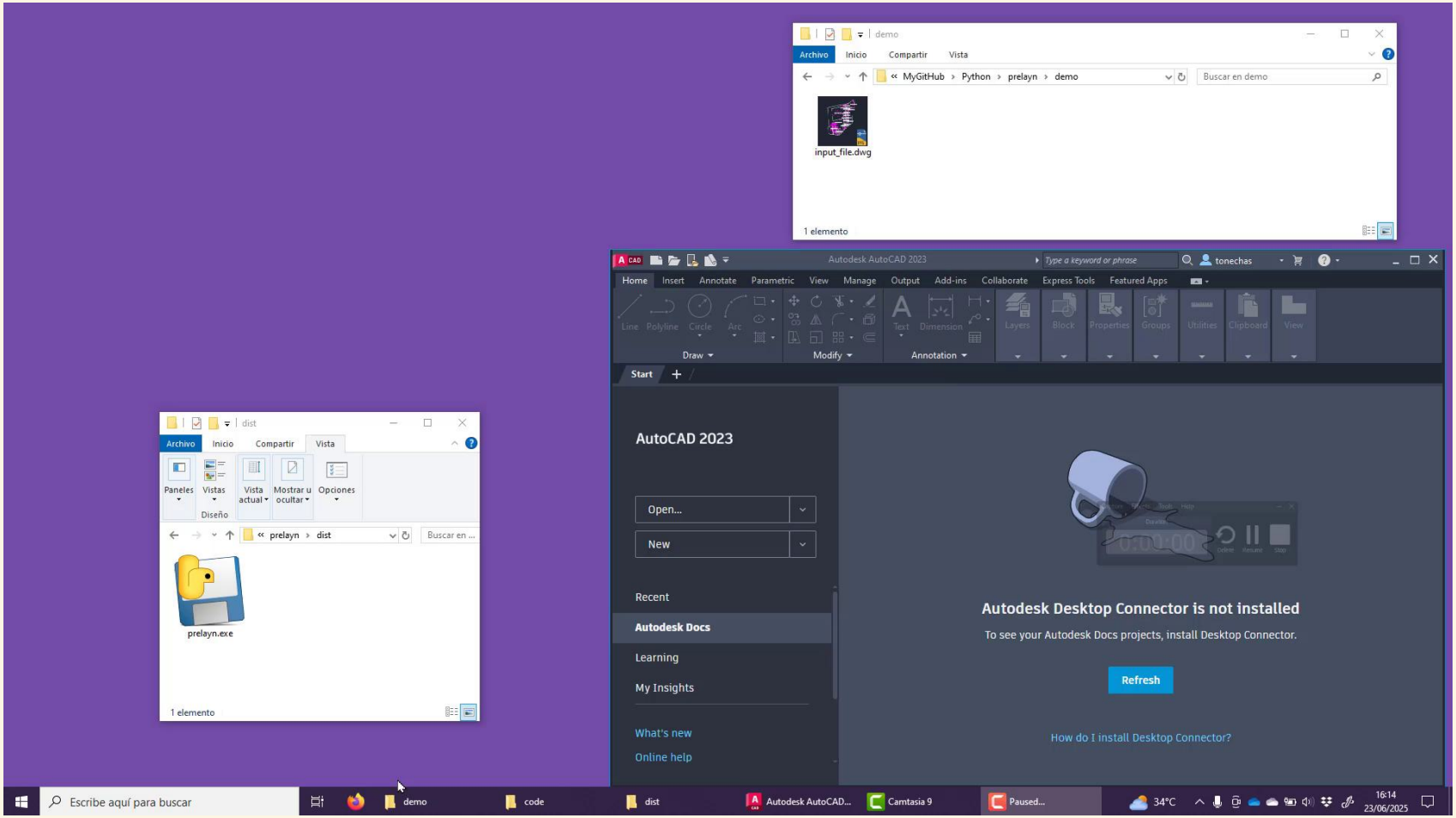
win32com

- Uses the COM interface via `pywin32`
- Full control of the AutoCAD environment
- Requires AutoCAD to be running
- Prone to runtime errors and needs error handling

```
from win32com.client.dynamic import Dispatch

prefix = 'CADASTRE_'
acad = Dispatch('AutoCAD.Application')
acad.Visible = True
doc = acad.Documents.Open('input_file.dwg')
for layer in doc.Layers:
    name = layer.Name
    if name not in ('0', 'Defpoints'):
        layer.Name = prefix + name
doc.SaveAs('output_file.dwg')
```

Automating the task with COM scripting (`win32com`)



pyautocad

- Simplified access to AutoCAD through COM
- Easier to learn than `win32com`
- Limited access to the full COM API
- Not actively maintained

```python
from pyautocad import Autocad

prefix = 'CADASTRE_'
acad = Autocad(create_if_not_exists=True)
for layer in acad.doc.Layers:
    name = layer.Name
    if name not in ('0', 'Defpoints'):
        layer.Name = prefix + name
```

ezdxf

- No need for AutoCAD or Windows
- Easy to use and well documented
- Fast and lightweight
- Works only with DXF files

```
import ezdxf

prefix = 'CADASTRE_'
doc = ezdxf.readfile('input_file.dxf')
clayer_name = doc.header['$CLAYER']
doc.header['$CLAYER'] = '0'
names = [layer.dxf.name for layer in doc.layers]
for name in names:
    if name not in ('0', 'Defpoints'):
        layer = doc.layers.get(name)
        layer.rename(prefix + name)
doc.header['$CLAYER'] = prefix + clayer_name
doc.saveas('output_file.dxf')
```

pyautogui

- Keyboard and mouse simulation
- Works on any platform
- Slow and fragile due to UI dependencies

```
import pyautogui as pgui
import os, time

def typewrite(keyboard_input, delay=1):
    pgui.typewrite(keyboard_input)
    pgui.keyDown('enter')
    pgui.keyUp('enter')
    time.sleep(delay)

prefix = 'CADASTRE_'
names = ['BUSQUEDA', 'CONSTRUCCION', 'PARCELA', 'REFCATASTRAL',
         'SUBPARCELA', 'TXTCONSTRU', 'TXTSUBPA']
os.startfile('start input_file.dwg')
time.sleep(3)
for name in names:
    typewrite('-LAYER')
    typewrite('Rename')
    typewrite(name)
    typewrite(prefix + name)
    pgui.hotkey('escape')
typewrite('SAVEAS')
typewrite('output_file.dwg')
pgui.hotkey('alt', 's')
```

Moving on to...

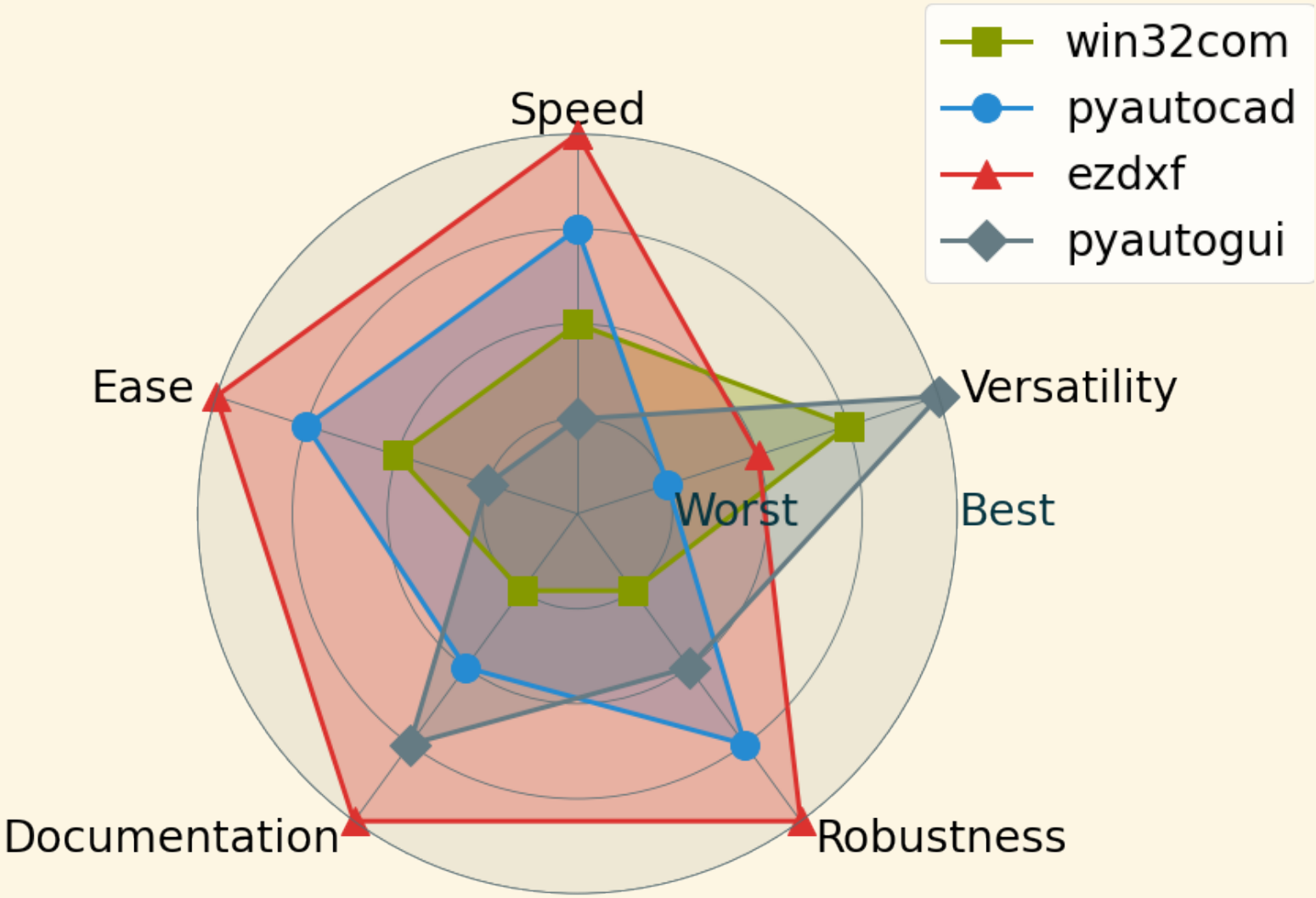
1. Introduction

2. Method

3. Results

4. Discussion

Results


win32com
pyautocad
ezdxf
pyautogui
Speed
Versatility
Ease
Worst
Best
Documentation
Robustness

Moving on to...

1. Introduction
2. Method
3. Results
4. Discussion

Discussion

Main conclusions

- Choice depends on task and user needs
- `pyautocad` is easy but limited
- `win32com` is powerful but error-prone
- `ezdxf` is the most convenient package

AutoCAD automation through Python scripting

Let Python do the job for you

34th INGEGRAF

Antonio Fernández[1], Lucía Díaz Vilariño[1], Francesco Bianconi[2]

[1]Universidade de Vigo, [2]Università degli Studi di Perugia

antfdez@uvigo.es antfdez.webs.uvigo.es tonechas