The CAD Engineer's First Week with C#

Date	April 17, 2026
Topic	CAD engineers learning C# for SolidWorks, AutoCAD, and Revit API work

No coding background needed. Just your CAD tool, Visual Studio, and this guide.

By Prasath | ScriptedCAD | https://scriptedcad.com

🚀 Want to go further after week one? Everything I teach is inside ScriptedCAD. scriptedcad.com

---

The advice you were given was wrong

If you searched 'mechanical engineer learning to code', almost every result told you to learn Python. Python is a great language. But if you want to automate SolidWorks, AutoCAD, or Revit, Python is the wrong starting point.

The SolidWorks API runs on C# and VB.NET. The AutoCAD API runs on C# and VB.NET. The Revit API runs on C# and VB.NET. These platforms were built on .NET. Their official examples, their documentation, their community resources: all C#.

💡 Learning Python to automate SolidWorks is like buying a spanner to drive a screw. Technically possible. Practically painful. Learn the tool the platform was built for.

I spent 8 years building CAD automation tools across SolidWorks, AutoCAD, and Revit. C# is what made the work possible. This guide gives you your first week with it, built around real CAD scenarios.

Before Day 1: Three things to install

You need three things. All of them are free.

1. Visual Studio Community: the free version of Microsoft's IDE. Download at visualstudio.microsoft.com. Select the .NET desktop development workload during setup.
2. Your CAD tool: SolidWorks, AutoCAD, or Revit. You need it installed and licensed. The API runs inside the application.
3. The CAD API SDK or type library: this lets Visual Studio understand the CAD objects you will be working with. SolidWorks installs it automatically. AutoCAD uses the ObjectARX or .NET SDK. Revit uses the Revit API SDK.

> ⚠️ Do not skip the workload step in Visual Studio setup. Without .NET desktop development selected, your C# projects will not have the right templates.

Day 1: Connect to your CAD tool and read one value

Your first goal is not to build anything useful. Your first goal is to make C# talk to your CAD tool. That moment when your code reads the name of the open document is the proof that the connection works.

Here is what that looks like in SolidWorks. Create a new C# Console Application in Visual Studio, add a reference to SolidWorks.Interop.sldworks, then write this:

```
using SolidWorks.Interop.sldworks;

// Connect to the running SolidWorks session
SldWorks swApp = (SldWorks)System.Runtime.InteropServices.M
arshal.GetActiveObject("SldWorks.Application");

// Get the active document
ModelDoc2 swModel = (ModelDoc2)swApp.ActiveDoc;
```

```
// Print its name
Console.WriteLine("Connected to: " + swModel.GetTitle());
```

Open a part in SolidWorks. Run this. You will see the file name printed in the console. That is your first connection. Everything else builds from here.

> ✅ Day 1 win: C# is talking to SolidWorks. You now know the connection pattern that every SolidWorks automation tool uses.

Days 2 and 3: Learn the object model

CAD APIs are built around an object model. Everything in your CAD tool is an object. SolidWorks is organised like this: the Application contains Documents. Documents contain Features. Features contain Sketches and Bodies. Learning to navigate this tree is the core skill.

Here is how to walk through every feature in an open part:

```
// Start at the first feature in the tree
Feature swFeat = (Feature)swModel.FirstFeature();

while (swFeat != null)
{
    // Print the feature name and type
    Console.WriteLine(swFeat.Name + " | " + swFeat.GetTypeN
ame2());

    // Move to the next feature
    swFeat = (Feature)swFeat.GetNextFeature();
}
```

Run this on any open part. You will see every feature in the tree printed to the console with its type name. This is exactly how automation tools audit feature trees, rename features in bulk, and detect non-standard modelling practices.

💡 Spend time on the SolidWorks API Help file. Search any object name and read what methods and properties it exposes. The API help is your map.

Days 4 and 5: Build your first real automation

Now you do something useful. A common task engineers do manually every week: saving a drawing as a PDF. Here is the automated version.

```
// Define the output path
string savePath = @"C:\Exports\" + swModel.GetTitle() + ".p
df";
int errors = 0;
int warnings = 0;

// Save as PDF using the Extension object
swModel.Extension.SaveAs(
    savePath,
    (int)swSaveAsVersion_e.swSaveAsCurrentVersion,
    (int)swSaveAsOptions_e.swSaveAsOptions_Silent,
    null,
    ref errors,
    ref warnings
);

if (errors == 0)
    Console.WriteLine("Saved to: " + savePath);
else
    Console.WriteLine("Failed. Error code: " + errors);
```

Open a SolidWorks drawing. Create the C:\Exports folder. Run this. The PDF appears. That is a real workflow automated in under 20 lines.

You can now loop this across every drawing in a folder. That is batch processing. A task that takes an engineer an hour of opening, printing, and

naming files takes your script under a minute.

> 🔥 This pattern, connect to the CAD app, get the active document, call a method on it, scales to almost everything you will ever need to automate at the task level.

What week 2 looks like

Task automation is the entry point. It is not the destination.

Week 2 is where you start touching design automation. Instead of saving files, you start creating geometry from code. You write a script that takes a length and a diameter as inputs and generates a turned part. You change one value and the model updates. That is rule-based design.

Further along, engineers who have gone through this process build things companies actually pay for: automated quoting systems that take customer inputs and generate a drawing package, product configurators that produce variants from a base model, drawing QA tools that audit every sheet before release.

> 🎯 The C# you learn this week is the same C# that powers those enterprise tools. You are not learning a toy language. You are learning the foundation of serious CAD engineering work.

Your week 1 checklist

- Visual Studio Community installed with .NET desktop development workload
- CAD tool open and ready
- First connection to SolidWorks working: swApp and swModel confirmed
- Feature tree traversal running: you can read every feature name and type
- First PDF export automated: one drawing saved without touching the export menu

- API Help file bookmarked: you know how to look up any object or method

---

🚀 Ready to go beyond week one? ScriptedCAD teaches the full path from first macro to production-ready automation tools. scriptedcad.com