Why Most Design Engineers Never Find Design Automation

Date	April 20, 2026
Topic	Why design engineers are unaware design automation exists and why it is not hard to learn

The awareness gap, the three barriers keeping you stuck, and why the learning curve is shorter than you think

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

🚀 If this guide shifts something for you, the next step is inside ScriptedCAD. scriptedcad.com

---

The Thing Nobody Told You

SolidWorks has an API. AutoCAD has an API. Revit has an API.

Every action you take manually in your CAD tool, every click, every file export, every drawing update, can be triggered, automated, and extended with C# code. The API is fully documented. It has been there for decades. And the vast majority of design engineers have no idea it exists.

This is not a knowledge gap about a niche feature. It is a structural blindspot. Nobody tells design engineers the API is there. It does not come up in university. It does not come up during onboarding. It is not covered in any CAD training course most engineers have taken.

💡 The engineers doing CAD automation are not smarter than you. They are not from a software background. Most of them found the API by accident, by Googling how to stop doing a specific repetitive task, and stumbling onto a macro that changed how they saw their tools.

This guide explains why design engineers do not find automation, what actually keeps them stuck, and why the learning curve is not what they think it is.

Barrier 1: Nobody Teaches It

Engineering degrees teach you how to design. They teach stress analysis, tolerancing, material selection, drawing standards. Some programs teach MATLAB for numerical work. Almost none of them teach you how to automate the tools you will use every day for the rest of your career.

CAD software vendors sell licences and training courses. Those courses teach you how to use the software, not how to control it with code. So engineers graduate, start using SolidWorks or AutoCAD or Revit, get efficient at the manual workflow, and never look any deeper.

The companies that hire design engineers are usually in the same position. The team lead knows CAD. The senior engineers know CAD. Nobody is building automation tools internally, so nobody shows the new hire that it is possible.

> ⚠️ The result: most design engineers spend 5 to 10 years in the industry before they learn the API exists. And a lot of them only find it because they hit a point where the manual work became unbearable enough to search for an alternative.

Barrier 2: The Identity Problem

Even when engineers do learn the API exists, many do not pursue it. Not because they cannot. Because they do not see themselves as the kind of person who writes code.

I am an engineer, not a programmer. That sentence ends the conversation before it starts. And it is based on a false distinction.

The engineers who build the best CAD automation tools are not software developers who learned CAD. They are design engineers who learned to code. The design knowledge is the hard part. Knowing what to automate, why a rule works, where the edge cases are in a parametric model, that all comes from years of CAD experience. The code is just how you execute it.

🔥 A software developer learning the SolidWorks API from scratch has no idea what IPartDoc or IAssemblyDoc is trying to do in a real manufacturing context. You do. That context is the advantage. The code is the easy part.

Calling yourself an engineer rather than a programmer is not a reason to avoid automation. It is actually the credential that makes your automation worth building.

Barrier 3: The Assumed Difficulty

When most design engineers imagine writing code to automate SolidWorks, they picture something like building software from scratch. A blank IDE. Hundreds of lines of unfamiliar syntax. Months of learning before anything works.

That is not how CAD automation starts.

SolidWorks has a built-in macro recorder. You press record. You do your manual task. You press stop. The recorder writes the VBA code for you. You then have a working script that mirrors exactly what you just did manually. Your first automation is written before you understand a single line of code.

```
' SolidWorks auto-generated macro - Save active doc as PDF
Dim swApp As SldWorks.SldWorks
Dim swModel As SldWorks.ModelDoc2
Set swApp = Application.SldWorks
Set swModel = swApp.ActiveDoc
swModel.SaveAs "C:\output\drawing.pdf"
```

That is not a code literacy test. It is a starting point. You edit it. You make it run on a folder of files instead of one. You add a simple loop. Each change teaches you something. You are not studying to code. You are coding to solve a problem you already understand.

💡 Most engineers write their first working automation in a single weekend. Not a polished tool. A script that actually runs and saves them real time. That first win changes the perception of difficulty permanently.

What Design Automation Actually Is

Here is the part most engineers do not reach because they stop at barrier 1 or 2.

CAD automation is not just removing repetitive tasks. That is the entry point. The real value is design automation: building tools where engineering logic drives geometry.

- A parametric bracket tool where an engineer enters load and material, and the model updates to a compliant design automatically
- A variant generator that produces 40 product configurations from a single base model in minutes
- A standards checker that validates every drawing against your company's title block and annotation rules before release
- A product configurator where a customer selects options and a compliant model, drawing set, and BOM are generated without a designer touching CAD

These tools do not require a software team. They require one engineer who understands the design domain and knows enough C# to connect that knowledge to the CAD API. That engineer is not mythical. That is what a design engineer with 3 to 6 months of API learning becomes.

Why the Learning Curve is Short

Every software engineer starting on the SolidWorks API has to learn CAD concepts from scratch. What a part document is. What features mean. How assembly mates work. How drawing views are structured.

You already know all of that.

The CAD API is a thin layer on top of the tool you have been using for years. The objects in the API, IPartDoc, IFeature, IDrawingView, are direct representations of things you already work with every day. When you read API documentation, you are not learning a foreign system. You are reading a technical description of what you already know.

> ✅ The transition from design engineer to automation engineer is shorter than any other path to the same destination, because you already have 80% of the knowledge. The remaining 20% is how to express what you already know in code.

The macro recorder gives you a starting point. C# gives you the language to express real logic. The CAD API gives you the objects. Your engineering knowledge gives you the understanding of what to build. Nothing in that list requires starting from zero.

What Crossing the Gap Changes

Once you write your first working add-in and see it solve a real problem inside your CAD tool, two things shift.

First, you start seeing every manual process differently. Not as work to be done, but as a problem to be solved once and automated. That shift in how you see your job is not reversible.

Second, your value inside an engineering company changes. Most engineering teams have no one who can build internal tools on top of CAD. If you can, you stop being one of the design engineers and start being the person the team depends on. That is a different career position, and it happens faster than most people expect.

> 🎯 The awareness gap is the only real barrier. Once you know the API exists, once you know the first step, and once you see that your design knowledge is an advantage rather than a handicap, the path is shorter than anyone told you.

🚀 Ready to cross the gap? I teach the full path from first macro to production add-in inside ScriptedCAD. scriptedcad.com

---