CES500017

Automation Thinking for Civil Engineers & BIM Professionals

Shinu Mathew

ARCADIS Consulting (i) Pvt. Ltd.

Learning Objectives

- Automation thinking vs Geometrical thinking
- Essential steps to develop an effective BIM/Design automation
- Structure your Automation the most practical way
- How to develop Automation Logic
- Visual Programming or Text Coding?

Description

The technology evolution in the Design & Construction industry is forcing all of us to adapt to the fast-changing digital landscape. Digital invasion into the regular design space is so rapid and overwhelming, so much so that many processes that were robust, tried & tested few years ago, now considered obsolete. New tools & technologies, new capabilities are springing up everywhere. These new tools help create newer technologies and as of now Automation is one such key topic.

This class is aimed at the beginners, who have either started automation, or planning to start. The examples used are basic ones, using AutoCAD problems.

Speaker(s)

Shinu Mathew heads the Implementation of BIM & Design Automation across ARCADIS – GEC India. Having worked in the industry for almost quarter of a century, initially as a CAD Draftsman with a strong flair for programming, and later donning many hats such as GIS Consultant, Survey Manager, Programmer, Team Lead which provided him with an extensive experience in almost every realm of Civil Design process. He supports all project teams across all geographies with BIM implementation – Process, Tools, Review – and Design Automation.

When he is not busy working, he likes to ride his motorcycle, and is a voracious reader & love acryllic painting.

Innovation

A presentation by Harvard Business School Professor, Clayton M Christensen, narrates the dilemma of Inventors. He talks about two types of innovations. Sustaining Innovation & Disruptive innovation.

Sustaining Innovation

Sustaining Innovation thrives in improving the current product performance by reducing defects, increasing efficiency & speeding up production time.

Disruptive Innovation

Disruptive innovation often challenges the existing practices and initially appears to be doing everything wrong. It does not conform to the framework set by the present practices and the proponents of Sustaining Innovation would often neglect it.

Automation

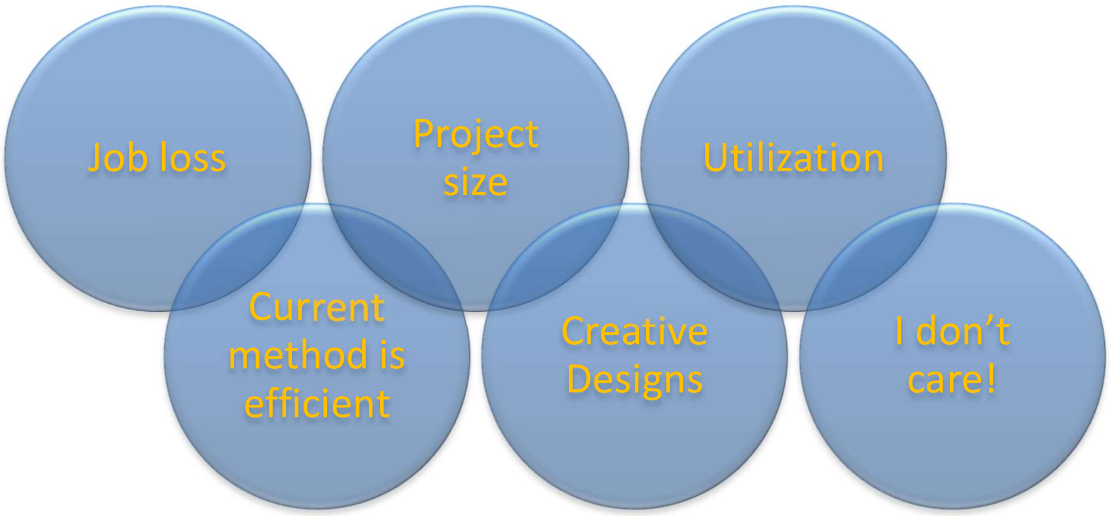
Automation is the process of certain tasks or processes done with minimal human intervention, leaving our precious brain power to do more creative & original tasks.

What is Automation Thinking?

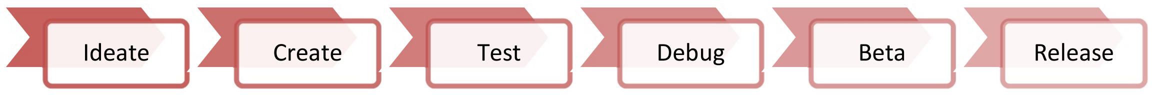
Automation thinking is the process of identifying / recognizing / executing opportunities in everyday life. We will be talking only about our work life here, as Civil Engineers / BIM Practitioners.

Why Automations?

Howmuchever we think that the existing tools are matured enough to carry out every task, Automations will, most times, improve the efficiency of the tool, reduce rework / wasted work. In this segment we will explore the most common excuses not to automate and break some pre-concieved notions. Some of the common notions are;



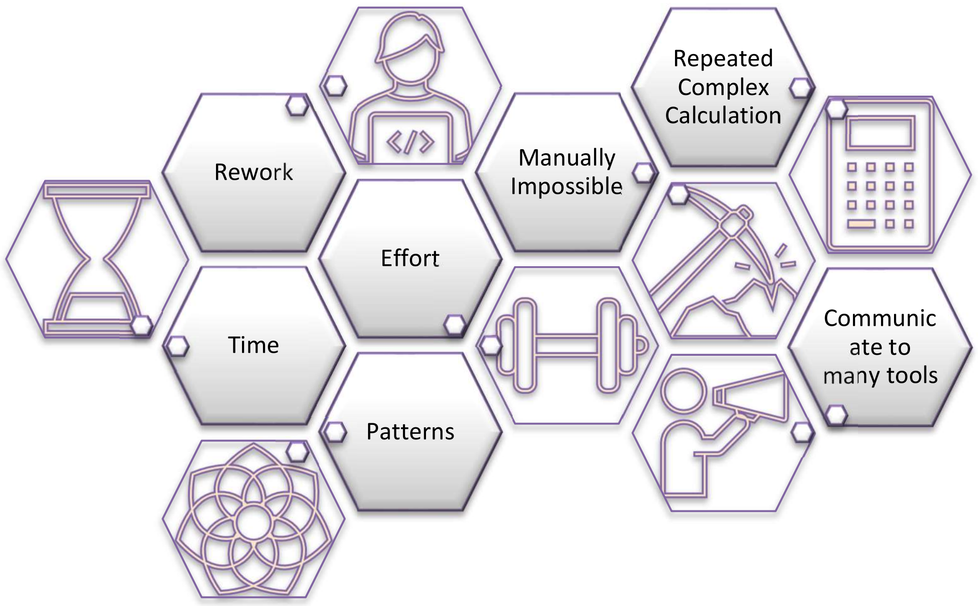
How to automate?



Ideate

Find opportunity / Define the problem

To find an opportunity and determine its viability takes years of experience, logical thinking & reasoning.



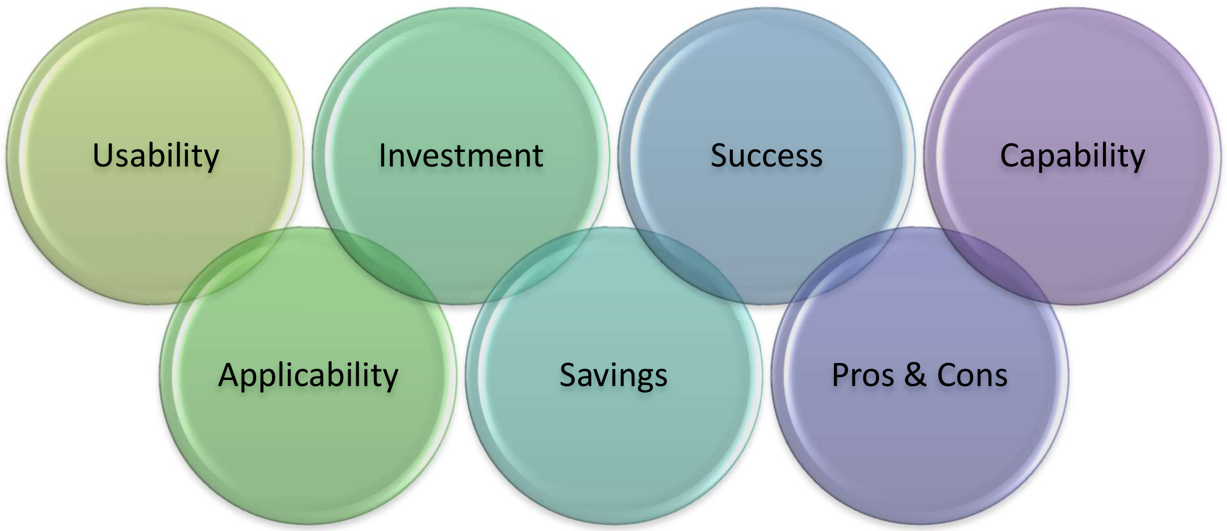
Brainstorm / collaborate

Most times, Automation beginners would think every opportunity is unique and it needs to be created. Furthermore, there is the "how-do-I-learn-if-I-don't-write-code?" part.

While one need practice to learn, think about the possible pitfalls. May be a better or similar solution already available? Maybe you will spend too much time R & D-ing? May be your knowledge level is not sufficient to crack the puzzle? It is better to seek opinion from experts and with their help, you would learn more and the right way.

Is it worth Automating?

How do we determine the worthiness of a certain case? Again, this is where your Automation Community can come to your rescue. If immediate help is not available ask the basic questions to yourself.



Business Buy-In.

Now that you have assessed all preliminary requirements and have a good case for automation, time to get the necessary approvals.

Unless it is a small script, talk to your Line Manager and get the buy-in. Discuss the plan, advantages & savings, but share the downside of it too. The cost, your time, success probability etc. should be explained properly.

Geometry thinking Vs Logical Thinking

As designers / Modelers, we tend to think in geometrical terms. Simple example is when we create a rectangle, what we see is 4 lines connected at the ends at right angles! But for programming, we need to see the relation between coordinates to be able to draw that.

APIs (Application Programming Interface)

Most times there will be a function / method within the programming language / tool to do most common calculations if the tool is specifically built for the modelling platform. Examples are Dynamo, GC or on a 2D level, LISP. But if it is a tool designed not exclusively for the authoring platform, then we might need to write the algorithms. Examples VBA, VB.Net, C#, Python etc.

Application Programming interface (API) will enable the Coding Platform talk to the Target Application, by exposing the Target's internal functions to the code language, thus the programmer can call them in the code.

Intelligence of program

Branching / Conditional logic.

Intelligent decision making is important in coding. If a certain value meets a criterion, then it should execute a branch of the code, and if it does not a different one

Data types

Most programming tools use the data types. Data type is the type of a certain value / parameter with which the programmer intends to process.

Events

Earlier programming languages, such as LISP, had a structural process, meaning executing code one block after the other. With the advent of Object-Oriented languages, Event-driven running of blocks of code brought in more efficiency & ease of execution.

Platform - Standalone / Add-On, Visual Program or Text Based?

Deciding the platform to do the automation is also important. A less-competent platform would result in not achieving the full capability envisioned. On the other hand, using a complex platform for mundane tasks would result in unnecessary complexities & more effort for small outcomes.

Create

Once all these aspects are considered, there are few more steps if you want to build a good, sustainable code.

Scalability

Perhaps the solution you think of is a unique one! Perhaps not! If it has a wider application, then we must think & provision for that.

Decomposition

At first glance, most problems would look intimidating. It raises doubts in our mind, too many complexities, etc. To effectively manage coding / graphing, we need to break down these challenges into simpler, smaller chunks.

Charting the flow

Once we have the smaller pieces of problem, preparing a high-level flow chart would be of great help. Relying on a person's memory for how the data should flow, what branching logic we are thinking of etc., would not be an ideal way to work on a complex problem.

Prototype / MVP

As a Prototype / Minimum Viable Product, we would produce a working, skeletal frame. It is to see the flaws in the logic, process & structure.

Capture or Pre-empt Errors?

Capturing / Trapping Errors is another integral part of programming. Without proper error handing, the program could crash at the smallest error, can corrupt databases, or simply not work.

Testing / Debugging

In order to produce a working solution that works seamlessly, it needs to be tested vigorously. The first phase of testing is done by the developer.

Beta

In the beta version, only a limited group of people, who have domain expertise will test the tool. This is to ensure that it works on all systems, different computers & environment, practical parameters, and outputs. During this stage there will be comments / errors coming back which the developer should incorporate / address.