MACRO SOFTWARE INTEGRATION IN OPERATING SYSTEM FOR OFFICE AUTOMATION

Sudesh Pawar[1], Yogen Ghodke[2]
[1] Student, Department of Computer Engineering, MESCOE, Pune
[2] Student, Department of Computer Engineering, MESCOE, Pune

ABSTRACT

Automation is the creation of tools to automate any task or a group of tasks sequentially without human intervention. Such automation systems are widely used in the industry in a huge variety of applications, especially on the production line of machines. Automation is not only limited to performing physical tasks but is also used to carry out virtual operations. Applications and methods of automation in terms of software are pretty limited. In current times, employees in corporate sector invest an unnecessarily high amount of time in performing menial repetitive tasks, thus hampering their work productivity and resulting in a loss to the company. The time saved if these tasks could be dealt more effectively will help in the growth of overall work execution. After understanding this issue, the authors decided to work on this issue and this paper is a result of these efforts. The main objective of this paper is to develop a model which would address the issues present in the current scenario and be viable at the same time. The main issues while developing such a system was that it could result in heavy usage of computational resources, but the model proposed solves this issue. The paper combines the principles of sub-domains of Artificial Intelligence i.e. Cognitive Action Selection, Self-Management, Network Management Systems, Expert Systems to design the model, thus being competent as a Smart System. This paper focuses on a fundamentally different approach to software automation in corporate environments in reducing time-complexity, resulting in a paradigm shift in the conventional approach towards enhancing organizational productivity.

Keywords: *Office Automation, Productivity, Macro Tool, Scripts, Visual Coding, Software Modelling, AI.*

1. INTRODUCTION

Many corporate organizations today, have very sophisticated networking and database management systems. This helps them save time, leading to an overall increase in their management productivity. But what all these companies fail to notice is that, most of their energy and time are spent in performing menial repetitive tasks on their computers, such as downloading email attachments manually, navigating through the directories to store them in the dedicated drive and then locating that file later on to forward it to the managers in the higher organizational hierarchy working on the respective consignment. Another example is manually editing the same settings repetitively over and over again before printing every file, just because there is an absence of a save settings option. Imagine the amount of time which will be saved, if these repetitive tasks could be automated with a smart scripted software which would download attachments, one-click print a file, etc.

There is an increasing need of a single OS-integrated macro tool, which has access to system functions and browser navigation menus and be able to operate at the backend with nominal yet pertinent computational requirements thus not hampering the frontend operation of the system and hence the work of the employees.

1.1 What is Software Automation?

The terminology Automation, derived from the Greek word Automaton, means creation of tools and technological counter-measures in order to control and oversee the production and execution of disparate services without human intervention [1]. It accomplishes errands which were previously performed by humans. It crosses all domains within the industry from installation, integration, maintenance to design, procurement, and management. Automation involves a boundless genus of technologies which span from robotic and expert systems, electro-optics, process-control, sensors, systems integration and many more.

Software Automation, uses computer software as an applied tool to design systems which will meet the goals of automation such as expeditious execution, superior management and reduction in time-complexity. In layman's terms, software automation can be said to be the 'Usage of software tools to automate routine tasks.'

1.2 Need for Software Automation.

In recent years the corporate world has witnessed many technological advances, which has increased the urgent demand for premium quality products and services which can be only supplied by a high level of productivity and workflow management. Corporate life can be exhaustive. Employees work under condensed deadlines and constant pressure to meet the client's ultimatums. But that doesn't mean that they must exert themselves to bear with the unnecessary decision-making which thwarts their work focus and results in lackluster focus in key areas. Decisions such as selecting fonts and sizes before typing in an email and then choosing the proper recipients on your team, squander a large part of the brain, considering the task to be done is just a message to be passed on.

Now imagine having an "email_script.exe" file on your desktop, double-clicking on which would upshot in a Pop-up text box, where you type in the subject and your message and just hit enter. The list of the recipients, the text formatting options, your signature at the bottom etc. all would be pre-programmed by you in your script file, tailored just the way you want it ; so when you hit enter, it figures out the rest and your task is executed immaculately without you having to make a lot of choices.

2. BASIC MODEL OF SOFTWARE AUTOMATION

The Following Block Diagram explains the Model of the Automation Software we propose, we shall see the individual components in detail later on, in 2.1 and 2.2.

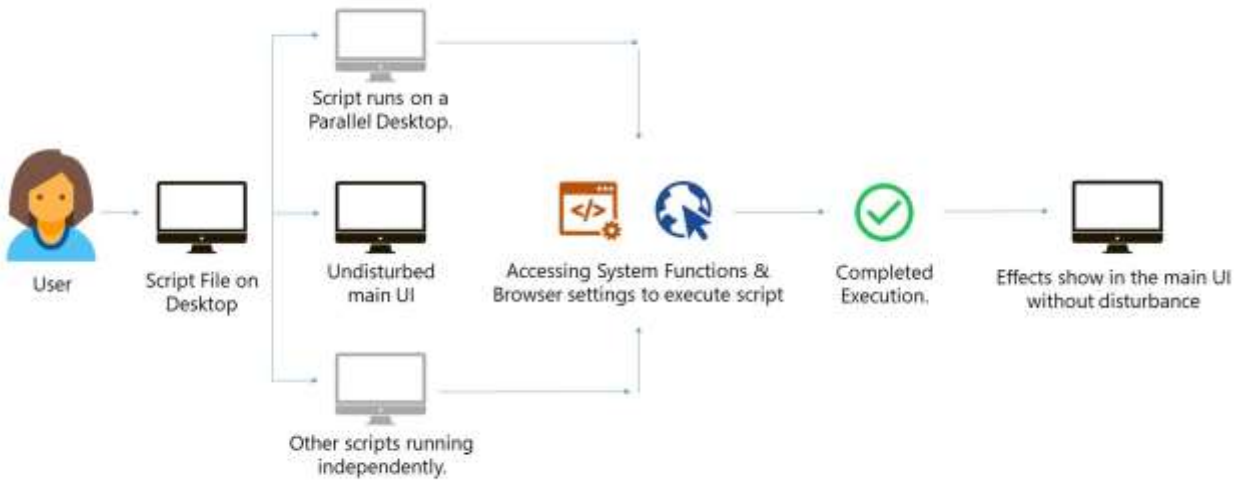

Figure -1: Block Diagram of the Software Model

2.1 User Interface (UI)

The UI of the proposed software model is the platform where the user can record and customize their own scripts. Initially, it consists of a 'Record Script' button, clicking on which minimizes the application and brings the user on the desktop. Now the user has to navigate through the UI performing the task they want to automate, just as they would normally do. The application runs in the background recording the user actions in the script, in the form of system functions accessed from the backend (hence the OS Integration). We call this as "Visual Coding", where the user doesn't really write the code, but it is automatically written through the users actions. Once the user presses the 'Stop Recording' hotkey, they will be brought to a screen where the actions they performed while recording would be shown sequentially as a list. The user can choose to manually edit the script, adding delays, triggers, jumps, conditional statements and deleting unnecessary actions. Once the script is ready, it can be exported as an executable file on the desktop to be used for running multiple times. Double clicking the .exe results in the formation of a parallel desktop which runs in the background [shown in Figure 1] without disturbing the current user experience. The following table [Table 1] is an example of how the script file perceives user's actions and automatically writes code by accessing system functions from the backend.

Table -1: User Actions vs. Recorded Scripts

	Opening Browser	Opening a Website	Marking a Checkbox	Mouse move
User Navigation	Double clicks Chrome icon.	Types gmail.com in address box.	Marks the "don't show this message again" checkbox.	Moves mouse from one corner to another corner of the screen,
Recorded Script	run chrome.exe	goto_url https://www.gmail.com	system_toggle::checkbox_1	mouse.move:(253,357),(570,640) [in brackets are the screen co-ordinates in the form of pixels]

2.2 Core Software Model (CSM)

The Core Software Model is a cardinal component that handles the backend system of the entire process. It will have access to system functions and would carry out all the processes without human intervention. The core will be made using the utilities of Artificial Intelligence for smart processing and conventional development of the simple processes, thus being the perfect blend of the conventional meets smart.
The domain of Artificial Intelligence is pretty extensive, but we shall require only the following sub-domains:

- Cognitive Action Selection
- Self-Management & Network Management Systems
- Expert Systems

Even though the main focus of the paper is not on the AI aspect of the model, the paper could not be well-explained in absence of it.

For the working of the Core, we shall take a look at the example explained in Table 1:
First, the Core shall receive the incoming email address from the user in form of script-code and then it will call the Expert Systems designed for the process of spam filter which will smart classify the emails and prioritize them. The useful email attachments will be downloaded and stored in the designated directory with separate folders for each email address. Then the Core shall give a call to the expert system designed for the process of selection of recipients which shall read the contents of the mail and determine the correct recipients, forwarding the mail to them. As soon as the mail is delivered, a return value of 1 shall be returned to the UI Mechanism which will generate an Acknowledgement Message.

2.2.1 Cognitive Action Selection

Action Selection is a way of characterizing the most basic problem of intelligent systems: 'What to do next'. Typically for any one action-selection mechanism, the set of possible action is predefined and fixed. Without usage of Cognitive Action Selection, our model would not be able to actively select a path ahead in the execution of script. The nodes in the Core Model have to be equipped enough to make a selection and this is where Cognitive Action Selection comes into play. We can use either of the symbolic, distributed or dynamic approach for this purpose, but the system has to execute a fixed script so a dynamic approach fits in pretty well.

2.2.2 Self-Management and Network Management Systems.

Even though Self-Management isn't exactly a subdomain of Artificial Intelligence, the model emphasizes on this part because it is essential for the smooth operation of the system. Self-Management is the process by which computer systems manage their own operation without human intervention. Similarly, Network-Management is process of managing system networks. Such Self-management systems will pervade next generation of Network Management Systems. Designing such Self-Management Systems is complex and not exactly under purview of this paper but there can be simplification in designing by utilization of design patterns such as Model View Controller which will help in encapsulation of functional concerns. [2]

2.2.3 Expert Systems and its need in Model

Expert Systems are a subdomain of Artificial Intelligence. An expert system is a computerized system that emulates the ability of decision making in a particular field as found in a human expert in that field.[3] As the expert systems primarily focus on only one domain, they are extremely useful in designing an automation system. Our entire backend of the system is nothing but a complex network of small Expert Systems pertaining to respective fields.

3. SCRIPT EXECUTION

Every Automated program has three Phases:

a. Trigger
b. Execution
c. Result

3.1 Trigger

A trigger can be any stimuli, which causes the program to go into the execution phase. These triggers can be added not only in the beginning but also in the middle of a program. The triggers at the beginning of the script are called as executional triggers. Some examples of such stimuli are:

a. Waiting for a pixel color to change.
b. Incoming of an email in the inbox.
c. Waiting for the user to press a specific key.
d. Passing of a return value to the feed.
e. System time showing [2/7/2018 _ 9:41:00 AM]

3.2 Execution

Once the program is triggered, it works in the file explorer and browser environments through the backend, accessing directives, working according to the pre-programmed model script.

3.3 Result

The Result is the only part of the program, visible to the user inside the main UI. It is generated after the successful execution of a script.

4. INTERNAL WORKING OF THE CORE SOFTWARE MODEL

As seen above in the example, the Core Software Model is an intelligent system which will be the backbone of the coding scripts which would be generated by the visual coding of the user in the first phase of recording scripts. Without explaining the internal working, this paper would have been incomplete in a sense.
The Core internal structure can be briefly divided into three parts:

1) Reading Nodes
2) Central Nodes
3) Output Node

4.1 Reading Nodes (or Input Nodes)

The Reading Nodes are the ones which will read the User Inputs from the Visual Code one generates using the UI Console. The Model will consist of following reading nodes:

- Mouse Node: Will read the Mouse Interactions and convert in Vector form.
- Pixel Node: Reads information related to the pixel such as pixel color hex code, co-ordinates, etc.
- Keyboard Node: Take the Keystrokes input and store in the corresponding ASCII format.
- Browser Node: All Interactions in respect to a Browser would be handled by this node.
- Interface Node: All OS-Interface interactions will be stored in this node.

Every Reading Node shall be connected to different Central Nodes and they shall pass on the information in a form that the Central Nodes should understand. Basically, the Reading Node acts in a manner similar to the namespace std in C++. The Namespace links and standardizes the various functions from the user code to the header declaration and definitions. Similarly, the Reading Nodes will link the visual script generated by the user to the functions stored in Central Nodes.

4.2 Central Nodes

The Central Nodes are the ones which will actually give call to system functions after getting the information in a readable manner from the reading nodes. The Node would execute in following manner:

- Get the interpretation of user inputs from the reading nodes.
- Determine the functions that shall be required by computing combined input from the reading nodes and store them temporarily in a list.
- Generate a Machine level code to access the system functions and call them one by one with the help of temporarily stored function list.
- After getting the calls executed, perform the required work with the modified data obtained post system function calling.
- After performing the required work, send the machine level output to the Output Node.

Every Central Node shall be connected to different Output Nodes and they shall pass on the output to the Output Node. Basically, the Reading Node acts as an amalgamation of an assembler and an algorithmic state machine. The reason for such a comparison is that it also converts the visual code into a machine level code but also executes the algorithm as required for smooth operation.

4.3 Output Node

The Output Node is the one which will read the Output obtained from the Central Node and generate a user understandable version of the same. Functioning of the Output Node is as follows:

1. Fetches the outputs from the various central nodes and store them.
2. Bind the outputs together and obtain the main output.
3. Convert the Main Output from the machine language to the Visual Output.
4. Pass the output to User Interface for display to user.
5. Interface Node: All OS-Interface interactions will be stored in this node.

Thus, we can say that Output Node is basically the user side Interpreter for the entire process and is primarily focused on obtaining user understandable outputs.

This concept of CSM is illustrated diagrammatically in Figure 2.

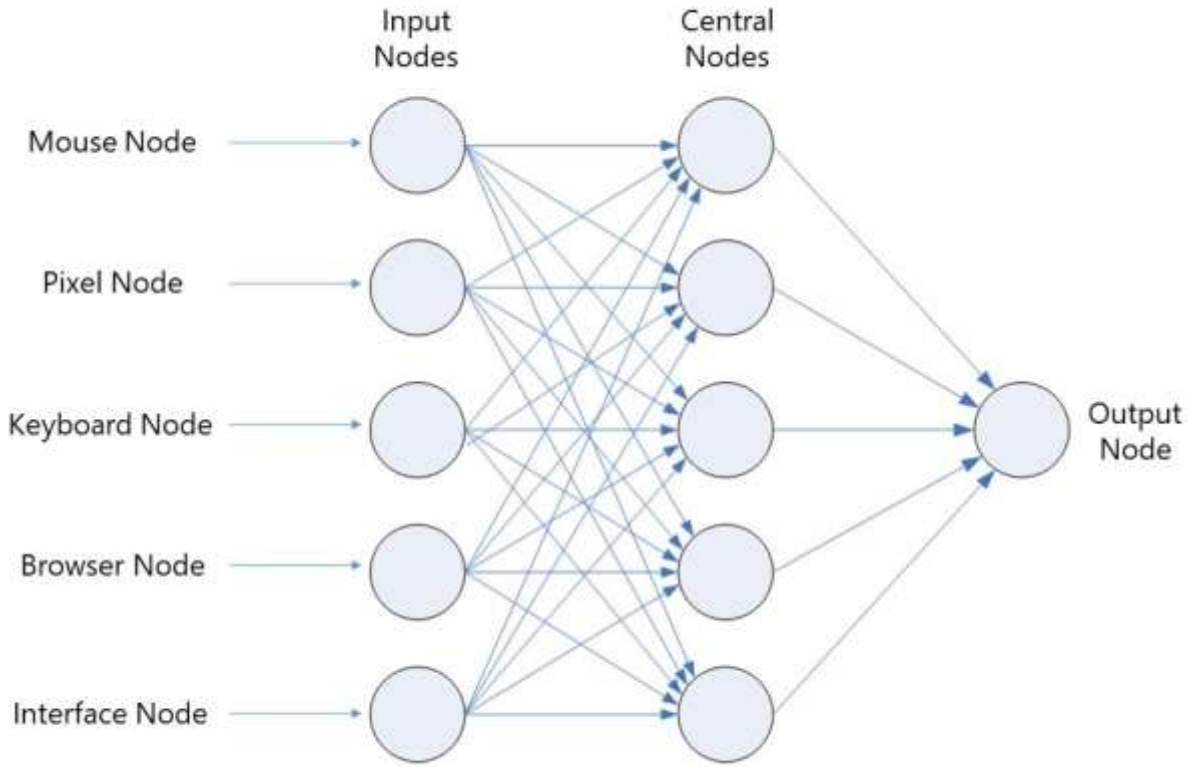

Figure -2: Block Diagram of the Core Software Model.

5. SCOPE

The scope of the proposed model extends not only in the areas of corporate management, but also has its roots in the enhancement of individual productivity. It has a wide array of applications, ranging from cross-application automation to system error solving. It is a compelling vision which we predict to be mainstream by 2030, where such systems will be as common as a screenshot is. Also, there will be a thriving ecosystem involving software engineers writing complicated scripts for companies and earning a buck, creating an entirely new profession. Also, this concept can be utilized on mobile platforms, making it a much broader sub-domain of automation.

5.1 Applications.

The following is a non-exhaustive list of scenarios where such a proposed model could function as a solution:

1. System Error solving :
   Imagine an elderly woman working in a convenience store, encountering an error on her computer. The solution of her error lies buried deep within her control panel, which she doesn't know how to navigate. She could easily search her error online, download the script file corresponding to the solution and just double click on it, which would magically solve her error without having to get into the control panel.

2. YouTube ViewBotting :

   You can create a YouTube ViewBot to increase the number of likes and views on your YouTube videos overnight. YouTube increments the view count on a video if it is visited by a new IP Address for the first time. So the algorithm for the ViewBot would be:

   a. Register a new Proxy IP Address.
   b. Go to the Video URL.
   c. Press the Like button and comment something nice (handled by AI for a variety of compliments).
   d. Go to step a.

   Here, the like button can give you the approximate count of the number of times the loop was executed. Overnight, this algorithm can give you thousands of views; enough to launch your video into orbit, so that it starts showing in people's recommendations. From there on you can start getting genuine views from real humans. Similarly, this can also help you earn money through Pay-Per-Click advertising by changing your IP Address each time before clicking your own Ad. But the authors strongly condemn such actions and view this in a negative light as it would lead to the collapse of a highly functional ecosystem of cash flow.

3. Automated Investing.

   Imagine, your investing strategy involves buying a stock for 5 days at a fixed price and selling it at a higher rate at another fixed price. You can easily write a script for it so that it works with your DEMAT account broker, executing your strategy for you without you having to sit in front of your computer, doing it manually.

4. Automated Form Filling:

   Whether you are on a shopping website or on a government site, there are a lot of form details that you need to fill. The AI would smart-identify the type of text fields from the browser backend by accessing the website's source code. Your script will contain information about the details to be filled corresponding to a particular text field. All you have to do is press the hotkey to trigger the script for automatic form filling and proceed to use the website without any hassle.

5.2 Misconceptions about Automation.

Myth: The Automation industry would result in massive layoffs and cause mass unemployment.
Fact: Contrary to popular belief, new technologies will create new jobs and change existing ones [4]. Based on studies, we can expect around 12 percent of US jobs to be automated over the next 10 years, displacing around 19 million workers. But these same studies also predict that these technologies should create 21 million new jobs maintaining the unemployment figures as they were before.

Myth: Automation always replaces an entire human.
Fact: Automation just takes over repetitive parts of the job. The creative parts are entirely to be handled by humans. For example, An AI technology can summarize the plot of a movie, but only a human movie critic can experience the emotions stirred by the movie and pen them down to rate the movie.

Conclusion: So automation isn't going to take away a teacher's job; rather, it'll make the teacher more effective.

6. CONCERNS

While using a script file downloaded from the internet, there might be an attempt to install malicious programs or steal user data across the internet, which would go unrecognized as the script runs in the background. The solution to this problem would be, hosting a verified platform (like Google Play Store or App Store) to share scripts, where experts could go through the code and review it as clean. Unexperienced users might rely on these ratings to remain safe from threats.

7. CONCLUSION

While researching on this topic, we found that many companies still follow age-old practice of manual execution of routine tasks. Our paper focuses on this problem and proposes a new Smart Automation System that helps to solve the issue of over utilization of time in trivial tasks. It brightens, with its futuristic vision, the small light which was already illuminated with courtesy to other research papers in this field. The theoretical computations clearly show that our proposed model shall help in saving tremendous time while doing repetitive tasks and avoid unnecessary complexity while performing error solving due to the merit which lies in the automation system.

8. ACKNOWLEDGEMENT

We would like to thank our First Year Head of Department, Dr. D.S.Adkar [M.Sc., M.Phil., M Ed, Ph.D.] for encouraging us towards research and Dr. S.B.Sharma [M.Sc., Ph.D.], for imbibing work ethics in us without which this paper would have never seen the light**.** We would also like to thank the Department of Computer Engineering, Modern Education Society's College of Engineering, Pune, India for their unconditional co-operation.

REFERENCES

[1] What is Automation? By Ken Goldberg, in IEEE Transactions on Automation Science and Engineering Vol. 9 No.1, January 2012
[2] Flexible Self-Management using the Model-View-Controller Pattern, by E. Curry and P. Grace, in IEEE Software, vol. 25,no. 3,pp. 84-90, May 2008
[3] Introduction to Expert Systems (3 ed.), Addison Wesley, p.2, by Peter Jackson, 1998, ISBN 978-0-201-87686-4.
[4] What to do when Machines do Everything, by Malcolm Frank, Paul Roehrig, Ben Pring (Wiley Publications), 2017

BIOGRAPHIES:

Sudesh Pawar[1]

Pursuing BE (Computer Engineering) at MESCOE, Pune (2nd Year of Course as of 2018).

Interests are Artificial Intelligence, Automation, Deep Learning, Human Computer Interaction, Natural Language Processing, Neural Networks and Web Development.

Yogen Ghodke[2]

Pursuing BE (Computer Engineering) at MESCOE, Pune (2nd Year of Course as of 2018).

Interests are Organic Chemistry, Space Sciences, Probability, Statistical Mathematics, Game Theory, Finance, Machine Learning, Big Data Analytics, and Automation.