



Automatic Python Source Code Generation using Artificial Intelligence Techniques

Prepared by

Samer ALHaddadin

Supervised by

Dr. Ayad T. Al-Zobaydi

Co-Supervised

Prof. Mohammad S. Saraireh

A Thesis

**Submitted to Faculty of Information Technology as a Partial Fulfilment of
the Requirements for Master Degree in Software Engineering**

2022, January

قرار تفويض

أنا سامر باسم الحدادين - افوض جامعة الاسراء بتزويد نسخ من رسالتي ورقيا والكترونيا للمكتبات، او المنظمات، او الهيئات، او المؤسسات المعنية بالأبحاث والدراسات العليا عند طلبها.



التوقيع:



التاريخ:

Authorization Statement

I am Samer Basem ALHaddadin, authorizing Isra University to provide hard copies or soft copies of my thesis to libraries, institutions, or individuals upon their request.

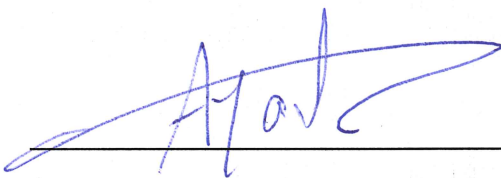
Signature:

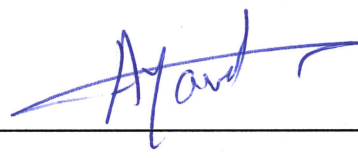


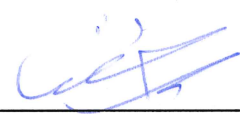
Date:

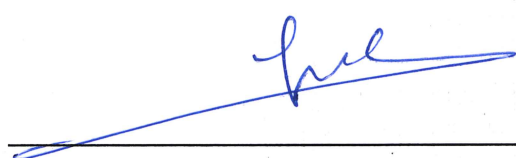


The undersigned have examined the thesis entitled 'Automatic Python Source Code Generation using Artificial Intelligence Techniques' presented by **Samer ALHaddadin**, a candidate for the degree of Master of Science in Software Engineering, and hereby certify that it is worthy of acceptance.

19, Jan, 2022 
Date Prof. (Associate) Dr. Ayad Tareq Imam

24 January 2022 
Date Prof. Mohammad S. Sarairoh
(on behalf)

23-5-2022 
Date Prof. Dr. Ja'afar Al-Sarairoh

24-01-2022 
Date Prof. Dr. Thamer Rousan

DEDICATION

I want to dedicate this message to my parents, family, and my supervisors, Dr. Ayad T. Al-Zobaydi and Prof. Mohammad S. Saraireh. This work would not have been completed without their support.

I will always be grateful for everything they have done for me, especially my friends Baha Yasin and Alaa Samara for supporting me and giving me all necessary help, for the many hours of reading, for assisting me, and for providing the necessary work circumstance to complete this work.

To all colleagues of the Master's Trip, all thanks and respect, and I wish you ever success in your life journey....

ACKNOWLEDGMENTS

First of all, I would like to say Thanks to God for all the blessings I have been blessed with, my parents, family, supervisors, friends, and teachers. I would like to express my sincerest gratitude to my advisors Dr. Ayad T. Al-Zobaydi and Prof. Mohammad S. Saraireh.

I would like to my thesis committee for their time and endeavors to aid me with my work and their constant encouragement through the good and bad times.

I also would like to extend my thanks to all my colleagues, and my professors who without I wouldn't be the person I am today.

Finally, and most importantly, I would like to thank my parents and my wife for giving me the opportunity to grow and become a better version of myself. Their constant support, help, and caring are what helped me throughout my life.

TABLE OF CONTENTS

قرار تفويض	2
AUTHORIZATION STATEMENT	2
DEDICATION.....	4
ACKNOWLEDGMENTS.....	5
TABLE OF CONTENTS	6
LIST OF TABLES.....	9
LIST OF FIGURES.....	10
LIST OF ABBREVIATIONS	11
CHAPTER 1: INTRODUCTION.....	14
1.1. OVERVIEW	14
1.2. RESEARCH QUESTION.....	14
1.3. RESEARCH AIMS AND OBJECTIVE	14
1.4. MOTIVATIONS.....	14
1.5. CONTRIBUTION(S).....	15
1.6. RESEARCH METHODOLOGY	15
1.7. THESIS OUTLINES.....	17
CHAPTER 2: BACKGROUND AND RELATED WORKS.....	18
2.1. BACKGROUND.....	18
2.1.1. THE CODE GENERATOR	18
2.1.2. THE WIZARD.....	20

2.2. THE RELATED APPROACHES AND WORKS.....	21
2.3. SUMMERY.....	25
CHAPTER 3: THE PROPOSED INTELLIGENT WIZARD TECHNIQE	28
3.1. INTRODUCTION	28
3.2. THE PROPOSED APPROACH.....	28
3.2.1. LIFESTYLE, INTERNET, AND USER	29
3.2.2. WIZARD	30
3.2.3. QUESTIONS MANAGER.....	30
3.2.4. CODE COMPOSER	31
3.2.5. QUESTIONS DATABASE.....	31
3.2.6. ANSWERS DATABASE	32
3.2.7. SOURCE CODE FILE	32
3.3. SMART HOME COMPOSER CASE STUDY	32
3.3.1. LIFESTYLE, INTERNET, AND USER.....	34
3.3.2. THE WIZARD PART OF THE SMART HOME SOFTWARE COMPOSER.....	35
3.3.3. QUESTIONS MANAGER PART OF THE SMART HOME SOFTWARE COMPOSER	36
3.3.4. CODE COMPOSER PART OF THE SMART HOME SOFTWARE COMPOSER	37
3.3.5. QUESTIONS DATABASE PART OF THE SMART HOME SOFTWARE COMPOSER	38
3.3.6. ANSWERS DATABASE PART OF THE SMART HOME SOFTWARE COMPOSER	39
3.3.7. SOURCE CODE FILE	40
3.4. SUMMERY.....	40

CHAPTER 4: EVALUATION OF IWT	42
4.1. INTRODUCTION	42
4.2. EVALUATION OF SMART HOME SOFTWARE COMPOSER	42
4.3. ACHIEVEMENTS OF IWT.....	47
4.4. SUMMERY.....	48
CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS	49
5.1. CONCLUSIONS.....	49
5.2. RECOMMENDATIONS	49
REFERENCES.....	51
APPENDIX A: THE SURVEY	56
A.1 THE STRUCTURE OF THE USED SURVEY	56
APPENDIX B: THE SUBMITTED PAPER	59
B.1. INTRODUCTION.....	59
B.2. THE IJACSA JOURNAL RANK IN SCOUPS AND ISI INDEXES	60
B.3. FIRST PAGE OF THE PAPER.....	63

LIST OF TABLES

No.	Table	Page
1.1	Timeline to Accomplish the Research Work	7
2.1	Abbreviated List of the Related Research Works	17
3.1	Lifestyle Table	25

LIST OF FIGURES

No.	Figure	Page
1.1	The Work Plan of the Thesis	6
3.1	The Architecture of the Proposed Intelligent Wizard Technique	20
3.2	The Graphical User Interface (GUI) of the Smart Home Software Composer	26
3.3	The Wizard Method of the Smart Home Software Composer	27
3.4	Question Manager Method	28
3.5	A Screenshot of Code Composer Part of the Smart Home Software Composer	28
3.6	A Screenshot of the 'Question' Database File	28
3.7	A Screenshot of the 'Answers' Database File	29
3.8	Sample of the Python Code	30
4.1	The GUI of the Smart Home Software Controller	33
4.2	The First Page of the Survey	34
4.3	The Succeded Tries of the Manual Approach vs the Automatic Approach	36
4.4	The Average Time of the Manual Approach vs the Automatic Approach	37
4.5	The Usability Measures of the Smart Home Software Composer	38
A.1(a)	The First Page of the Used Survey	47
A.1(b)	The Second Page of the Used Survey	47
A.2	Sample of a Feedback from a Programmer	48
A.3(a)	Statistics of Usability Criteria	49
A.3(b)	Statistics of Average Time	49
A.3(c)	Statistics of Performance	49
B.1	The Submission Approvement	50
B.2	Email to Acknowledge the Reception of the Paper	51
B.3	The Statistics of the IJACSA Journal on Scopus	51
B4(a)	The Statistics of the IJACSA Journal on Web of Science	52
B4(b)	The Statistics of the IJACSA Journal on Web of Science	53
B4(c)	The Statistics of the IJACSA Journal on Web of Science	53

LIST OF ABBREVIATIONS

Abbreviation	Word
4GLs	Fourth generation Programming Languages
ACAI	Automatic Coder using Artificial Intelligence
ACG	Automatic Code Generation
AI	Artificial Intelligence
API	Application programming interface
ASCG	Automated Source Code Generation
ASTs	Abstract Syntax Trees
CASE	Computer Aided Software Engineering
CBR	Case-based reasoning
ECG-RF	Expert Code Generator Rule-based
ES	Expert Systems
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HCI	Human- Computer Interaction
I_CASE	Intelligent Computer Aided Software engineering
IDE	Integrated Development Environment
IDEs	Integrated Development Environments
IR	Information Retrieval
IWT	Intelligent Wizard Technique
JAD	Java decompiler
LFW	Learning from Wizard
RNNs	Recurrent Neural Networks
SE	Software Engineering
UI	User Interface
UML	Unified Modelling Language
WoZ	The Wizard of oZ

Automatic Python Source Code Generation using Artificial Intelligence Techniques

Prepared by

Samer ALHaddadin

Supervised by

Dr. Ayad T. Al-Zobaydi

Co-Supervised

Prof. Mohammad S. Saraireh

Abstract

While the current Computer Aided Software Engineering (CASE) tools give a notable help to the developers in composing programs, there is still a need for more flexible supporting software tools to address the raises in the complexity of composing programs. The automating of the human's intellectual activities that are required to compose a program can be the answer for such need.

While the traditional Wizard suffers from the ability to collect the answers else than human, this research work proposes the definition of the Intelligent Wizard Technique (IWT) as a new Automatic Code Generator (ACG) strategy to collect answers to certain questions from different resources (in addition to the user as the usual wizard does) to automate the generation of source code. Based on this proposing, a Smart Home Software Composer case study of the defined IWT have been developed that can generate a Python language source

code of a smart home controller. The resulted Python code has been tested on a real home and the results showed the soundness of the code. IWT can be classified as an Intelligent Computer Aided Software Engineering (I-CASE) tool.

The evaluation of the Smart Home Software Composer case study of the defined IWT was achieved by using the objective measure of the performance, which evaluates to 91.6 %, and the subjective measure of usability, which evaluates to 85% for satisfaction, 91% for efficiency, and 97% for ease to use. These values show preferable indications to the programmer.

Keywords: Inferencing; learning by observation; Source Code Generation; Wizard; Smart Home; Raspberry Pi; Python; I-CASE

CHAPTER 1: INTRODUCTION

1.1.Overview

This is the introductory chapter. It is about the constitutional parts of the research. In this chapter, the research question, aims, objectives, motivations, contributions, and methodology will be elaborated. This information establishes the reader mind to the research work that has been achieved.

1.2.Research Question

“Can we automate the development of a Python code to program the Raspberry Pi in a smart home application using Artificial Intelligence techniques?”

1.3.Research Aims and Objective

The aim of this research work is to develop a code generator that composes a Python code to program the Raspberry Pi in a smart home application. To achieve this goal, a set of objectives are to be accomplished, which are:

1. Studying the code generator approaches and techniques to select the most appropriate one(s) for the Raspberry Pi 4 in a smart home application.
2. Proposing as much as possible, an accurate, simple, and effective code generator by using Artificial Intelligence (AI) techniques
3. Applying and implementing the proposing solution
4. Testing and measuring the effectiveness of the resulted Python code in a real situation.

1.4.Motivations

1. Reduce the time and the efforts of coding,
2. Get more flexible code generator by using AI techniques.

3. Minimize possible coding errors.
4. Give a simple programming tool for to someone who cannot do programming..
5. Accommodate different programming languages for coding.

1.5.Contribution(s)

By completing our research work, we expect to have the following contributions that constitute future works in this fields:

1. Defining the Intelligent Wizard Technique (IWT)
2. Implementing a Smart Home Composer case study
3. Examining the resulted Python code (the Smart Home Controller) from the Smart Home Composer case study on real domain
4. Evaluation the results

These contributions, as we believe, would enhance the work on the development of software testing tools and provide more control on the quality of developing a software project.

1.6.Research Methodology

As illustrated in Figure 1.1, the work plan that has been developed to accomplish this thesis encompasses a number of steps, which are:

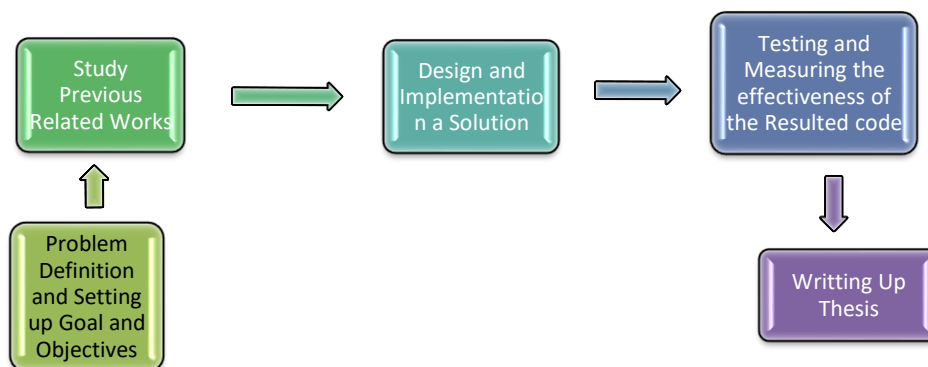


Fig 1.1: The Work Plan of the Thesis

1. **Problem definition and setting up goal and objectives:** the problem has been defined, which is the development of a source code generator for raspberry pi. Also, the objectives of this research have been listed as shown in fig1.1
2. **Study previous related works:** some previous related works have been collected. An analysis of these collected works has been made to help develop and define an approach that is suitable to solve the problem defined above.
3. **Design and implementation of a solution:** an approach, which is called the (IWT) has been designed. This approach will be implemented by using the C# programming language.
4. **Testing and measuring the effectiveness of the resulted code:** testing and evaluating strategies and techniques will be applied to show the soundness and effectiveness of the proposed implemented solution.
5. **Writing up the thesis:** this is the final step, which aims to document and illustrate the research work.

A timeline plan has been maintained. Table 1.1 shows the timeline for accomplishing each step of the work plan I have maintained.

Table 1.1: Timeline to Accomplish the Research Work

Months Activities	August 2021	September 2021	October 2021	November 2021	December 2022
Problem Definition					
Study previous related works					
Design and implementation					
Testing					
Writing up thesis					

1.7.Thesis Outlines

- Chapter 1: The introduction in which we are introducing our work.
- Chapter2: The preliminary information and background of the automatic source code generation, the smart home application, the Raspberry Pi, the Python programming language, and the related works.
- Chapter 3: The Proposed Intelligent Wizard (IWT)
- Chapter 4: Evaluation Of IWT
- Chapter 5: Conclusions and recommendations for future work.

2.1. Background

In the software Engineering (SE), the automatic source **code generation** is a technique that is used to quickly update and develop software using Automatic Code Generation (ACG) software tool [1] [2]. ACG software is an automated process intended for normal coding activity of software design. No doubt that ACG is of great potential for developing programs in a faster way because it helps save time and effort, improve program quality, and become more accurate, and help developers get rid of tedious routine processes. The technology of code generation is widely used and to facilitate the development of many various kinds of source code generators. For example, the Java decompiler (JAD) converts byte code to Java source code [3]. We cannot ignore the promising results of current ACG technologies and note that most of these techniques, especially official models, need input by humans. This is expected because the programmer's job requires innovation and creativity and is considered a creative (non-routine) job, ACG's curriculum prefers to force students to specialize in software engineers to work on non-routine jobs rather than to replace software engineering entirely. Passive code origin is a type of code generator that also produces code that needs to be modified and also modified by the programmer [3]. ACG technique is used to develop many applications. In our proposed research work, we intend to use ACG to develop the source code of **smart home** applications.

2.1.1. The Code Generator

In the field of SE, ACG is a common approach. The research community has created a number of tools for creating source code, ranging from graphical modeling tools like unified modelling language (UML) to target programming languages like Java and C# [4]

The code generation process is divided into many stages: formal implementations are translated into programs in a specific programming language using a translator's tool chain, and these programs are then compiled. This method has numerous advantages: the translation process is as straightforward as feasible, and it can be easily checked. [5]

The code verification is the most difficult step in the code generation process. The reason for this is because the preservation of architectural attributes at the code level is only ensured if the underlying platform is right and the final system is accurate when filling in the stubs for internal operations into the automatically produced code. [5]

Automatic code generator (ACG) software is used to generate code, which is a technique for fast software development. ACG program's objective is to automate routine coding activities in software development. ACG offers tremendous promise for quick software development since it saves time and effort, improves software quality and accuracy, and relieves developers of tedious repetitive chores. [3]

The process of code generation is becoming more popular, resulting in the creation of a variety of code generators. One example is the code wizard, which is featured in most Integrated Development Environments (IDEs) (that can be viewed as a frame-driven code generator). A further example is the forward engineering tool set (which includes reverse engineering software tools and compilers) that is incorporated into modeling tools and turns a solution description into code; for instance, the Java decompiler (JAD) transforms the bytecode into Java source code. Regardless the inability to overlook the encouraging outcomes of contemporary ACG approaches, it is worth noting that the majority of these techniques, particularly formal models of the intended system, are still in their infancy , Human involvement is still required. [3].

2.1.2. The Wizard

The Wizard of oZ (WoZ) technique involves participants who interplay with a system that appears to be independent, which is in actual is operated by a hidden human operator in a nearby place [6].

WoZ Systems that are currently in use. The majority of extant WoZ systems were created to investigate the use of natural languages in Information Retrieval (IR) systems. Experiments on the services of telephone information, such as phone directories, travel or train information, and reservation services, have proven fruitful [7] [8] . The experimental setup is straightforward: the wizard takes calls and acts as though callers are speaking with an automated information system. The wizard's voice is filtered via a distortion mechanism (like a vocoder), to add a robotic flavor to the voice for providing callers the impression that they are truly discussing with a computer.

Tape recordings of the questions and replies are made for subsequent transcription and analysis. Interrogation of databases or advisory systems [9] [10] [11] as well as conversations with Expert Systems (ES) [12] [13] , are examples of other case studies. The majority of them attempt to gather vocabulary corpora in order to fine-tune and improve the robustness of natural language recognizers, whether spoken or written. **Dahlbäck** [13] describes a framework that tries to allow the observation of graphical direct manipulation coupled with natural language. Turvy, an intelligent entity mimicked using a WoZ, may be trained via voice and direct manipulation, according to a recent article [14].

WoZ experiments have already produced a fascinating body of research regarding wizards and assessment experts, despite their restricted reach. Wizards have taught us a lot. The fact that wizards' duties are cognitively costly, despite their seeming simplicity, is an

intriguing consequence of the WoZ findings. The equipment' realism necessitates that the wizard's activities be constant in substance, manner, and tempo. Specifically, [7]:

- 1) A particular order from the subject must elicit the same response from the wizard in identical situations.
- 2) The wizard's reaction time must meet the subject's expectations: if the wizard reacts too slowly, the subject may avoid utilizing simulated functions or feel the system is overburdened.

In conclusion, wizards cannot afford to improvise. Wizards must be taught in well-defined duties and aided by strong tools in order to attain acceptable consistent behavior. To this aim, certain WoZ systems provide limited but helpful methods like a set of prepared responses or menus with pre-stored sections of answers. [15].

Recent studies recommend a two-wizard setup to reduce cognitive stress [7], with one wizard specializing in I/O and the other performing task level processing. The wizard's task is to understand the requests, which are translated by an I/O wizard and produces the answers. The I/O wizard gets user requests and conducts virtual answers. Consistency is more probable with this collaborative work sharing. If the wizards are well taught, it has no discernible effect on reaction time. Another experiment [16] that used a two-wizard setup was successful.

2.2. The Related Approaches and Works

In this section, we will show different approaches that are used in the Automatic Source Code Generation (ASCG) using Artificial Intelligence. We present and discuss some of the works that are relevant to the potential application. Relevant curricula and references were identified by searching the literature published since 2010. The focus should be on

relatively recent work; So that the field of artificial intelligence is large and wide and extends from the 50s to the present [17].

The first part of this section is the main content of this page and a survey of ASCG methods using AI. Secondly, a potential discussion about the relevant works that does not focus directly on a specific issue of the ASCG but may generate general and comprehensive ideas on how to approach it, will be given. [17].

Several ACG technologies have been developed. The first one we are mentioning here is the design pattern strategy that was introduced by Eric Gama [18]. It created code based on Floyd's "Code Processing Specification" research business model [19]. Also, grammar is a strategy that Knuth [20] says developed in his linguistic research and has also been used to express the semantics of a programming language, and to support compiler constructs building an Integrated Development Environment (IDE).

AI techniques are used also to automate code generation, such as genetic programming, evolutionary algorithms, and Case-based reasoning (CBR) [21] [22] [23]. Danilchenko and colleagues [22] proposed the mechanism A programmer using an artificial intelligence system that integrates routine design, state-based logic, and template programming that develops programs that deal directly with database operations. The author recommends the process of extending Automatic Coder using Artificial Intelligence (ACAI) by testing other types of layouts, as well as optimizing the number of criteria a user will use to select an icon rather than distance versus speed switch. Several software development tools appeared in the 1970s and 1980s: user interface(UI) developers, wizards, fourth generation Programming Language (4GL) application generators, and state tools. Compilers, assemblers, and the fourth generation Programming Languages (4GLs) and Enhancers These are tools that have been widely used and used to create code since long ago in computer science. These

examples are also considered ACG techniques [23]. And that all these symbols that stand out and that resulted from the current ACG methods and tools, still have a great need and human contribution. This is clearly and explicitly displayed by the latest ACG systems used by the companies or various organizations (such as NASA) that need to define very accurate models for the system under development By developers before creating the code for the system [3] [20].

The wizard technique is used to create a code to control a robot as the work of [24] This work used Learning from Wizard (LFW) to answer the question of “whether robots can be effectively programmed for autonomous social interaction through learning from demonstrations recorded via Wizard of-Oz teleoperation”.

The Generating of source code from a text query and predefined cases using case-based reasoning is given by [22]. The software developed in this work takes the specifications used in the text, specifically a medical database query, and generates similar Java code. It uses a combination of case-based thinking, routine design, and model-driven programming. Requirements for the programmer who uses the Automatic Coder using Artificial Intelligence (ACAI) system to define states using XML, and the advantage of the topic is that the user does not need to learn SQL at now, things are not clear about the success of this approach. Currently, ACAI "only solves database problems" with a focus on the medical database field. The authors show only a modest ability to combine schemas to produce simple database queries, which requires defining states using XML. In summary, this approach requires a lot of basic work to perform unimportant tasks, and it is not clear how easily the approach can be extended to more realistic conditions [17].

BAYOU [25] generates the source code from given label (like a number of application programming interface(API) calls or types), which having a little bit of information regarding

the desired code and a corpus of labelled programs. BAYOU generates Java code with the creation of labels and groups - training not only on the code but also on knowing and sketching the program - and its formation and construction in concrete.

The work of [26] produces the "glass box", which used the requirements and specifications information got from the auditor's source code of a specific program to validate the generated source code. The verification output is used to direct the creation of the program that meets the specifications. This work is based on the normal approach for verification and validation a source code. Normally, a program should be produced firstly and later passed into a verification and validation process to know whether the program is adequate or not. As this approach requires the writing of a program as a first step, which is considered unsatisfactory since it requires experience in the development and programming process. However, it may be easier to write a program to validate the answer than to write a program that produces the correct answers. The approach is interesting, but the problem of requiring a program to be written makes it more difficult to recommend it because not all people or users are experienced in programming or development.

Imam et al [3] proposed the Expert Code Generator Rule-based (ECG-RF) generate a source code. The user-directed inference system populates a predefined frameworks for static structures using code snippets from the knowledge base. Questions are given to the user, and the system proceeds the code's editing based on their answers. This approach is easy to understand and use, however creating rule-based systems becomes more difficult as rules become more complex, and the paper does not provide guidance on how to do this.

Other possibly related works include the various attempts have been made to produce natural language from source code (i.e., the opposite direction of ASCG) [27] [28] Recurrent Neural Networks (RNNs) are neural networks that support sequence training, which is useful

for applications like natural language translation and source code generation. As [28] points out, the RNN method has shown to be extremely successful for a variety of issues. Rather of trying to create text directly, some techniques focus on establishing a higher-level abstraction for source code. Abstract Syntax Trees (ASTs), for example, are used by many researchers like [29]

There is a believe that utilizing representations that make it easier to describe these higher-level structures will make it easier to employ well-known approaches like RNN models, as well as expand existing techniques to deal with the underlying structure of source code. Higher-level models like this should make it simpler for learning systems to see patterns and generate syntactically valid code (e.g., ensuring that block ending marks match block beginning marks) [17].

2.3. Summery

In this chapter, a careful reviewing has been made to get an idea about the previous tries that have been made to develop ASCG. Table 2.1 is an abbreviated list of the previous related research works. Based on this list, we can say that to develop an intelligence based ASCG software tool, it is necessary to make this software tool able to get the required information to compose a source code by means of intelligent techniques like learning from around and searching the Internet.

While all the previous related works has only one intelligent technique, our proposed solution would encompass more than one intelligent technique like inferring an answer searching the internet ,and learning from the environment using observation and sensors .

Thus our proposed approach is contributing a more flexible tool to compose a source code.

Table 2.1: Abbreviated List of the Related Research Works

#	Research Title	Input	Process	Output	Problem(s)
1	An Expert Code Generator using Rule-Based and Frames Knowledge Representation Techniques [3]	Answers to questions delivered to the user via graphical user interface (GUI)	Filling a predetermined frame of a given fixed structure program with code fragments that are retrieved from knowledge base by using heuristics	Assembly language code	Composing using predefined chunks of software
2	Automated Code Generation Using Case-Based Reasoning, Routine Design and Template-Based Programming [22]	Natural language description of a task at some level of abstraction	Integrating: routine design + state-based logic + programming templates	creating a program to solve number of medical database domain queries and Sub-queries from the list	Natural language complexities
3	Learning from the Wizard: Programming Social Interaction through Teleoperated Demonstrations [24]	Instructions / Commands issued by an operator (Oz teleoperation Wizard)	Learning From Wizard (LFW) (Recording the movement series issued by a child play with the robot or by Oz teleoperation Wizard)	A code to control a robot learns some things to deal with the child	Complexities of on-line learning approach
4	Neural sketch learning for conditional program generation [25]	A set of ready-made programs that contain labels	Supervised learning using number of API calls or the types utilized in the code.	Java like code	Pseudocode not programming code
5	Glass-Box Program Synthesis: A Machine Learning Approach [26]	Data use either input/output examples or rich execution traces	Given a partial program and the glass-box issue, an intelligent search system learns the probabilities over the space of programs.	A successful solution to a programming issue	Complexities of searching using probabilities

6	The Use of Natural Language Processing Approach for Converting Pseudo Code to C# Code [27]	Answers [Y/N] to questions given to the user	IF-THEN Rules, a predefined programs' structures, and a knowledge base of code snippets	A code of an MS-DOS device driver	shortcoming of the solutions of natural language ambiguity problems
7	A syntactic neural model for general-purpose code generation [29]	The underlying syntax of the target programming language	RNN + Patterns	A high-level abstraction in a natural language of the source code	Difficulties in collecting training data

CHAPTER 3: THE PROPOSED INTELLIGENT WIZARD TECHNIQUE

3.1. Introduction

This chapter is about the proposed solution to the defined problem in this research work, which is “The defining an (IWT) to automatically develop a source code”. As this, IWT aims to compose a source code, it is classified as an ASCG, and as it is intelligent, it has a distinguishing feature that it can get the required information by means of intelligent techniques as shown in the previous chapter.

To show the soundness of this definition, a case study that follow the definition of the IWT, has been developed. This case study is the development a Smart Home Composer that function to develop a Smart Home Controller. The resulted Smart Home Controller was composed in Python programming language and ran on Raspberry Pi platform, which controls set of objects in a home.

3.2. The Proposed Approach

The software wizard or setup assistant is a user interface that presents the user with series of dialog boxes, and the user fills in the data and transfers it to the other box depending on the inputs that the user has entered and leads the user through a series of well-defined steps. Tasks that are complex, erratic, or unfamiliar with the wizard may be easier to do [3]. The will-be adopted methodology aims to define and implement an (IWT). Fig 3.1 illustrates the process flow of the suggested IWT solution, which contains the following parts:

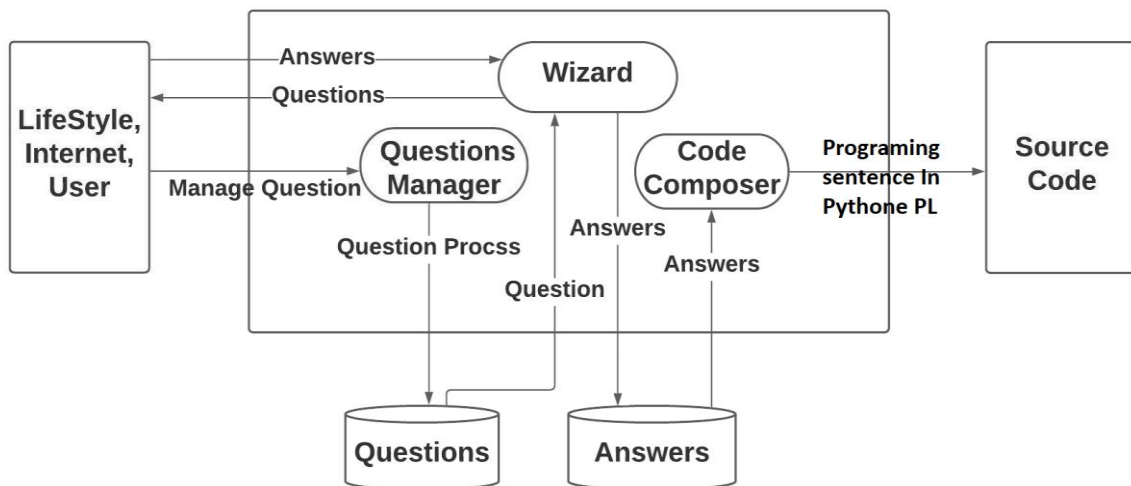


Fig 3.1: The Architecture of the Proposed Intelligent Wizard Technique

3.2.1. Lifestyle, Internet, and User

These are the resources from which our proposed IWT gets the information that are required to compose the code of a targeted application. The lifestyle is a recorder that records the way or style or behaviours of the beneficiary of the targeted (resulted) software application, that could be another application or a human. The lifestyle could be developed as a table to be part of the database of the IWT. The lifestyle is updated frequently by monitoring the user’s activities while handling (or solving) a certain problem. The Internet is other source that IWT can get the information from. No doubt that the Internet is a vast storage of information that anyone can get benefit of, but the main problem is how to develop a wise search for a specific datum from the Internet. Finally, the user is the human that run the IWT, who may give direct information to IWT to help composing the code of the targeted software. A sample example of the user source of information is what we experience in some preparing programs likes windows.

3.2.2. Wizard

It is the process that read the questions from the question database, and finds their answers either from a lifestyle table, the Internet, or the user. After conducting the answers, Wizard puts them in the answers database. The Pseudocode of the wizard is:

```
Wizard ()  
{  
  
    Read Question from 'Questions' database  
  
    Look for the answer in the Lifestyle Table  
  
    If the answer is not found in the Lifestyle Table  
  
        Then Look for the answer on the Internet  
  
        If the answer is not founded I the Internet  
  
            Then get the answer from the User  
  
    Save the answer in 'Answer' database  
  
}
```

3.2.3. Questions Manager

It is the process that manages the question database via adding, removing, updating, or listing the contents of the questions database. The questions manager process mainly the contents of the questions database regularly revising the contes by the admin of IWT to keep sure the suitability of the questions database to the targeted software application aimed to be composed. The Pseudocode for the questions manager is:

```
Questions Manager ()  
{  
  
    Read Option  
  
    If Option == 'Edit' Then Call EditQuestion method
```

```

If Option == 'Delete' Then Call DeleteQuestion method

If Option == 'Append' Then Call AppendQuestion method

If Option == 'List' Then Call ListQuestions method

}

```

3.2.4. Code Composer

It is the process that is responsible for creating/ editing/ composing a textual form of the targeted program in a certain programming language code based on the answers, which were saved in the answers database. The code composer process uses a standard template and fills it out with a programming statement that would utilize the answers in the answers database to compose the code. The Pseudocode for the code composer is:

```

Code Composer ( )
{
    Open Answer database for reading
    Open the code template for writing
    While not EOF Answer database
        Read an answer from Answer database
        Apply text processing to the answer to compose a programming statement
        Write the programming statements in the code template
    End While
}

```

3.2.5. Questions DataBase

It is a database that contains a set of questions, which help supplying the composer with the information it needs for composing a code. These questions are used by the wizard component of IWT, which are given to the lifestyle, the Internet, or the user. Each question

is saved textually in a file. The questions are varied from application type to another and should be set by the user of the IWT prior to running it to compose the code of a targeted application. The Question file is controlled by 'Question Manager' process, which is described earlier.

3.2.6. Answers DataBase

It is a database that contains the answers to the set of questions, which were given by wizard component to the lifestyle, the Internet, or the user. These answers are used by the composer component of IWT to compose the code of the targeted software. Each answer is saved textually in the Answer file. .

3.2.7. Source Code File

This is the text file of a source code that is generated automatically by our proposed IWT. It comes as a template that is defined according to the style of the IDE that is used to edit, compile, and running a programming source code. This template should be defined prior to running IWT, and to be filled by the composer part of the IWT.

3.3. Smart Home Composer Case Study

As a case study for our proposed IWT, we developed a Smart Home Software Composer, which is implemented by using C# programming. **A smart home** application aims to provide people with a comfortable life that contains all the means of comfort and protection [30]. A smart home consists of a group of sensors and controllers that are equipped with different objects in the home and are connected with each other by using modern tools and technologies such as Ethernet wires. A smart home consists of electronic devices and photovoltaic energy systems connected to each other and there is a responsible and controlling system for every part of it [31], and their information can be controlled and transmitted from/to outside the house by using smart home gates like **Raspberry Pi controller**.

Raspberry Pi is a small, palm-sized computer with an ARMV8 microprocessor and 4GB ram. The Raspberry Pi was developed in the UK by the Raspberry Pi Foundation. They first brought it to market in 2012 and was a huge hit. Raspberry Pi meet the needs of many things and people. For beginners and hobbyists, it was the ideal device and the best option due to its low price and at the same time powerful enough that can be easily used anywhere or to run small applications. In fact, there are some safety advantages of the Raspberry Pi, like its cheap price. Good for all groups, it can be easily installed around the house to run whatever application. Raspberry Pi often related to monitoring or voice interaction and controlling home matters such as lighting, water consuming management, and doors' controlling. In addition, they can be part of complex projects. For example, use them to control all parts of the house. Also, the advantage when making a small project, you will get results as soon as possible [32]. Raspberry Pi can be programmed using certain programming languages like **Python programming language**. The IWT case study that is the Smart Home Software Composer aims to generate a Python source code as Smart Home Controlling software.

The Python programming language works as the most common programming language. The highly interactive nature and a mature ecosystem of scientific libraries offer a best choice for software algorithm development and data analysis [33] [34]. However, as a code language, it is used not only in the field of computers and it works in the fields of industry as well and in many programs. [35]. Python is a very easy programming language for learning and reading. The origin of the word Python is taken from the English comics group Monty Python [36].

As shown in the following sections, each part of the IWT has been redefined as a case dedicated for smart home application. The procedure for each part hasn't change. Only the

data that is related to the pin configuration and the enabling/disabling value were customized for this special case of IWT.

3.3.1. Lifestyle, Internet, and User

In this case study, we utilized each of the Lifestyle, the Internet, and the user to be the resources from which the Smart Home Software Composer can get the information that are required to compose the code of a smart home software. As shown in Table. 3.2, the lifestyle was developed as a table, and it was displayed in the GUI of the Smart Home Software Composer. Its data is got from the daily uses of the home’s resident, while he/she uses the objects controlled by Raspberry Pi. Each controlled object has its own record, which holds the information of the last use by the home’s resident, and it will be used for future composing of the Smart Home Software. The metadata of this file encompasses each of the name of the object to be controlled (Name), the status of the object (status), the timing engaged with status (s-time), and finally the pin configuration in which the object was connected General Purpose Input/Output (GPIO).

Table 3.2: Lifestyle Table

	Name	status	s_time	gpio
▶	OutSide Light	On	19:18:12	26
	AC	On	19:18:19	24
	TV	On	19:18:34	19
	Light	On	19:18:46	16
	OutSide Light	On	19:31:49	26
	AC	On	19:32:06	24

The second source of the data was the Internet, which was used to get the daily sunrise and sunset times of the location of the home. Such information is of big importance to automatically compose the code of controlling the lights (specially the outdoor lights)

depending on such information. Of course, other unusual conditions such as being cloudy were considered and had been got from the weather website.

The last resource was the user, who gives the information directly either by selection a predetermined alternatives or type the data directly. Worth to mention here, that Smart Home Software Composer embedded the voice recognition facility in the resulted code to give the user the ability to ask the Smart Home Software controller about some information like time, date, or even invoke a setting process of an object under controlling.

3.3.2. The Wizard Part of the Smart Home Software Composer

This is the part that is responsible for collecting the answers to the questions that are used to compose the required source code. Fig 3.2 illustrates the Graphical User Interface (GUI) of the Smart Home Software Composer.



Fig 3.2: The Graphical User Interface (GUI) of the Smart Home Software Composer

The wizard part had the inference ability and the flexibility property that made it able to collect the data with minimum need to the user. The inference ability was achieved by

utilizing the lifestyle table and the Internet to get the answers. The flexibility property was achieved via the existing of three alternatives as a source of answers. Fig. 3.3 illustrates the method that function as wizard part of the Smart Home Software Composer.

```
//wizerd To Select Objects
var input = checkedListBox1.CheckedItems.Count;
//var input = Int32.Parse(Alaa.Text);
for (int i = 1; i <= input; i++)
{
    // string promptValue = Prompt.ShowDialog(xxx + " " + arr[i + count], "Button " + i);
    Button btn = new Button();
    // btn.Text = promptValue.ToString();
    btn.Name = count.ToString();
    btn.Size = new Size(150, 100);
    btn.Font = new Font(btn.Font.FontFamily, 16);
    btn.Location = new Point(50, 110 * y);
    // -----
    // MySqlConnection connection = new MySqlConnection("server=86.108.18.247;port=13606;Database=gpioDB;User ID=root;Password=123");
    MySqlConnection connection = new MySqlConnection("server=94.249.2.85;port=39379;Database=gpioDB;User ID=root;Password=123");

    MySqlCommand commandDatabase = new MySqlCommand("INSERT INTO `Answers` (`id`, `gpio_no`, `status`, `Name`, `used`) VALUES ('1','" + arr[i
    count--;
    commandDatabase.CommandTimeout = 60;
```

Fig 3.3 : The Wizard Method of the Smart Home Software Composer

3.3.3. Questions Manager Part of the Smart Home Software Composer

It is the process that is responsible for managing the contents of the ‘Question’ database file. It offers the services for adding, removing, modifying, and deleting the questions in the ‘Questions’ file. The importance of this process is shown by the required need to set up the composer to compose a source code for a certain application, which is differ from other type of applications, and thus, it needs different set of questions. Fig. 3.4 represents a screenshot for the code of this method.

```
//Questions Manager
MySQLConnection con2 = new MySqlConnection("server=94.249.2.85;port=39379;Database=gpioDB;User ID=root;Password=123");
//MySQLConnection conc = new MySqlConnection("server=86.108.18.247;port=13606;Database=gpioDB;User ID=root;Password=123");

MySQLCommand cmd2 = new MySqlCommand("select question from Questions where id=2", con2);

con2.Open();

MySQLDataReader rdr = cmd2.ExecuteReader();

rdr.Read();

//var reader = cmd.ExecuteReader();
var xxx = rdr.GetString(0);
// MessageBox.Show(xxx.ToString());
rdr.Close();
```

Fig 3.4 : Question Manager Method

3.3.4. Code Composer Part of the Smart Home Software Composer

This is the heart part of the Smart Home Software Composer. Code composer process functions at creating or generating a Python source code for controlling a home (smart home application). Code composer fills (by writing) a predefined template, which is a textual file that meets the structuring of a Python source code. In addition to the header and other complementary statements, the main type of statements is the ‘IF – THEN’ statement, which requires the condition part and the action part. Both of these two parts were got from ‘Answers’ database file, which had been filed already by the wizard part of the Smart Home Software Composer. Fig. 3.5 is a screenshot of this method.

```
//Generating Python Files
using var client = new SshClient("94.249.2.85",34459, "pi", "123");
client.Connect();
// client.RunCommand("mkdir "+DateTime.Now.ToString());
var cmd = "import os\nos.system(\"echo \" + "'1'\" + \" > /sys/class/gpio/gpio\" + arr[i + count] + \"/value\\");";
var cmdx = "import os\nos.system(\"echo \" + arr[i + count] + \" > /sys/class/gpio/unexport\\")\n\" +
"os.system(\"echo \" + arr[i + count] + \" > /sys/class/gpio/export\\")\n\" +
"os.system(\"sudo chmod 777 /sys/class/gpio/gpio\" + arr[i + count] + \"/value\\")\n\" +
"os.system(\"echo 'out' > /sys/class/gpio/gpio\" + arr[i + count] + \"/direction\\");";
// MessageBox.Show(cmdx);
client.RunCommand("echo '" + cmdx + "' > /home/pi/samer/gpio" + arr[i + count] + ".py");
cmd = "import os\nos.system(\"echo \" + "'1'\" + \" > /sys/class/gpio/gpio\" + arr[i + count] + \"/value\\");";
// MessageBox.Show(cmd);
client.RunCommand("echo '" + cmd + "' > /home/pi/samer/gpio" + arr[i + count] + "off.py");
cmd = "import os\nos.system(\"echo \" + "'0'\" + \" > /sys/class/gpio/gpio\" + arr[i + count] + \"/value\\");";
// MessageBox.Show(cmd);
client.RunCommand("echo '" + cmd + "' > /home/pi/samer/gpio" + arr[i + count] + "on.py");

client.Disconnect();
```

Fig. 3.5 : A Screenshot of Code Composer Part of the Smart Home Software Composer

3.3.5. Questions DataBase Part of the Smart Home Software Composer

These are the questions aim to collect the data related to the Smart Home controller. These questions had been set by us and saved in the ‘Questions’ database file. Examples of these questions are:

- ‘How many buttons you want to connect your system?’
- ‘What is the time you like to turn on the TV’
- ‘How much degrees you like the temperature of the room to be?’

Note that some questions are chained together since they form conditions or actions of a single programmed object. Fig. 3.6 is a screenshot of the ‘Question’ database file, which is developed as a table that consisted of 25 questions. Each question was saved as a record that encompasses three fields namely: id (the question number), question (the question’s text), and note (if there is something to be considered about this question).

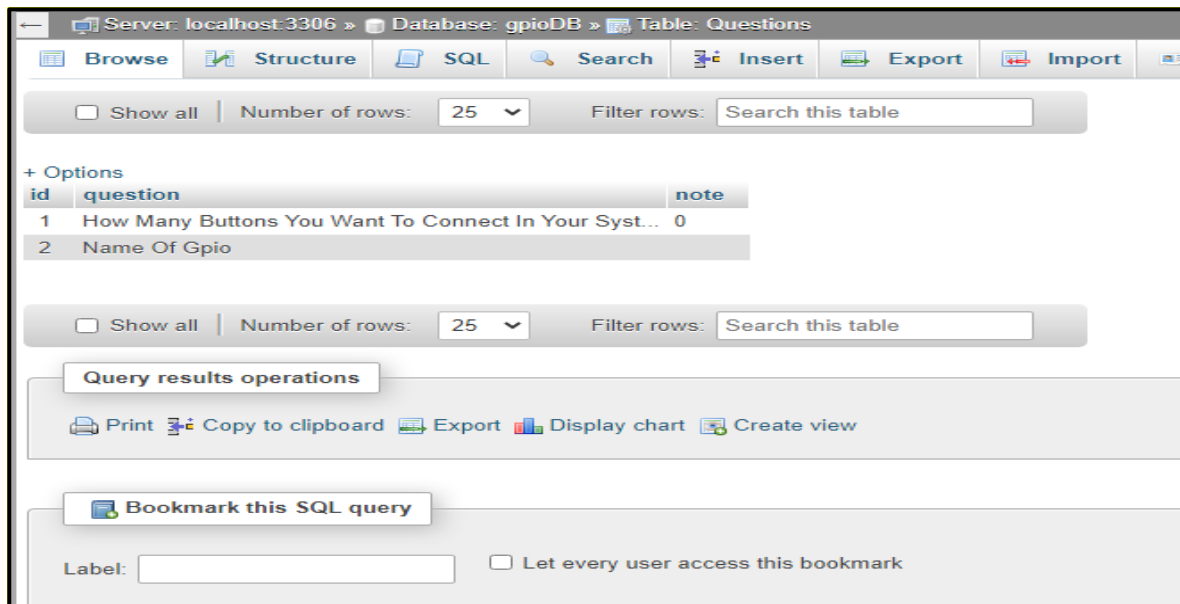


Fig. 3.6: A Screenshot of the ‘Question’ Database File

3.3.6. Answers DataBase Part of the Smart Home Software Composer

This database file holds the answers that would be collected by the wizard part of the Smart Home Software Composer. As shown in Fig. 3.7, 'Answer' database file had been designed as a table that consists of five fields, which are:

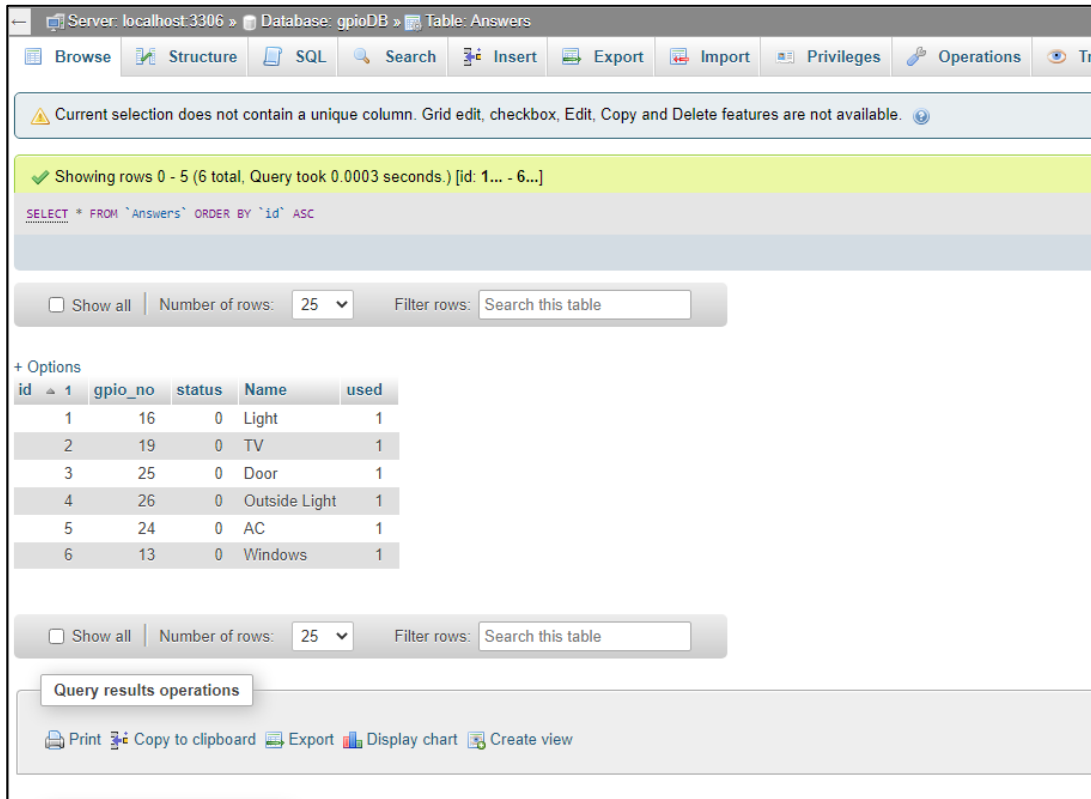


Fig 3.7 : A Screenshot of the 'Answers' Database File

- id: the sequence of the answer
- gpio_no: the Raspberry Pi pin number, where a certain object had been controlled.
- status: numerical representation of on/off
- Name: the device name
- used: numerical representation of the used/unused pin

Finally, this database file was got its contents from wizard part and used by composer part to assist composing the Raspberry Pi Python source code.

3.3.7. Source Code File

This is the text file of the Raspberry Pi Python source code that had been generated automatically by the composer part of our proposed IWT. It came as a template that is defined according to the style of the Raspberry Pi Python source code. Fig. 3.8 shows a sample of the Python code.

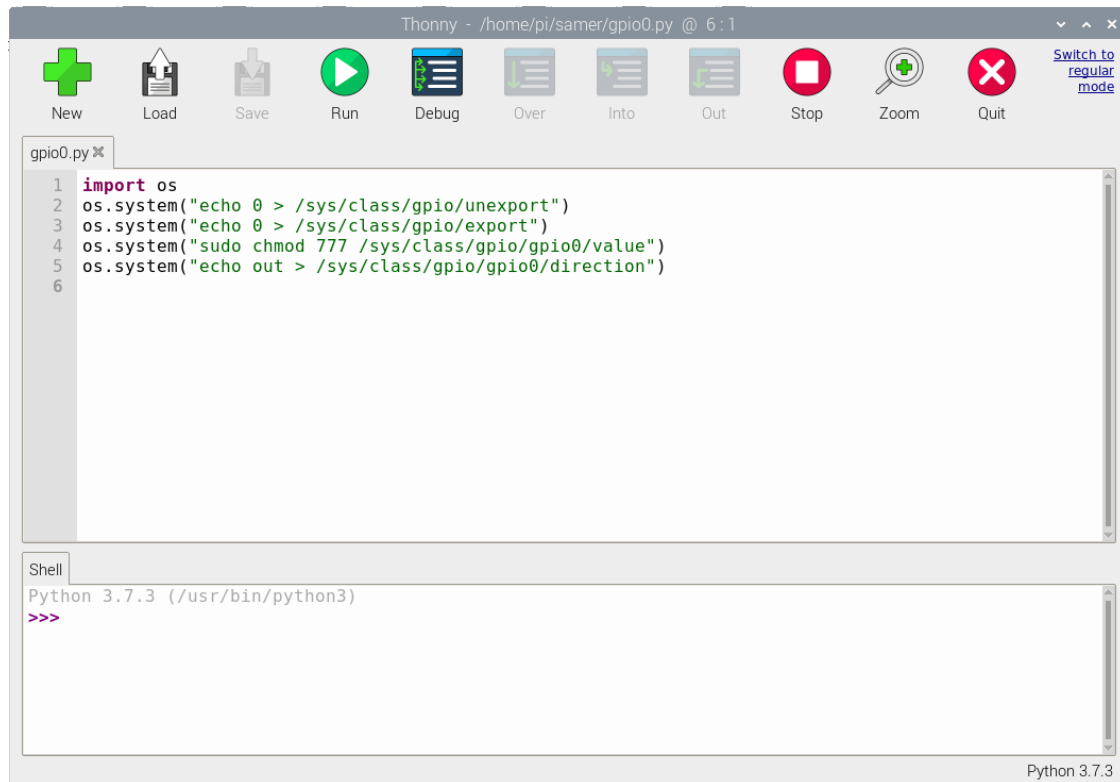


Fig 3.8 : Sample of the Python Code

3.4. Summery

In this chapter, the definition of the Intelligent Wizard Technique (IWT) has been given in details. Also, a case study that follow IWT definition has been developed and implemented using C# programming language. The Python code that was resulted from the Smart Home Software Compose case study of the IWT was tested on a model of a house to show the soundness of the resulted Python code. The house model was controlled by gpio controller. A template for Python code was developed in prior to compose the Python code.

Thus, each of template definition for the source code, the using of learning from daily user lifestyle, the retrieving of information from the Internet, and the using of Wizard techniques have been used to contribute a new I-CASE software tool for ASCG.

CHAPTER 4: EVALUATION OF IWT

4.1. Introduction

This chapter aims to evaluate the IWT through the evaluating of its case study that is the Smart Home Software Composer. The evaluation strategy depends on the evaluating some non-functional requirements since the result of the wizard in general is a source code. As we shall see in the next sections, the evaluation was implemented as a survey that reports the feedbacks of programmers who were asked to evaluate the Smart Home Software Composer case study of IWT. We were keen to select the people who can give us technical feedbacks, and this is the reason to exclude other users of non-technical background.

The feedbacks got from the survey have been quantified and statistically processed to reflect the gaining of IWT as a principle

4.2. The Evaluation of Smart Home Software Composer

Fig 4.1 illustrates the GUI of the Smart Home Software Controller. This interface was the fixed part of the template that had been filled by the composer.

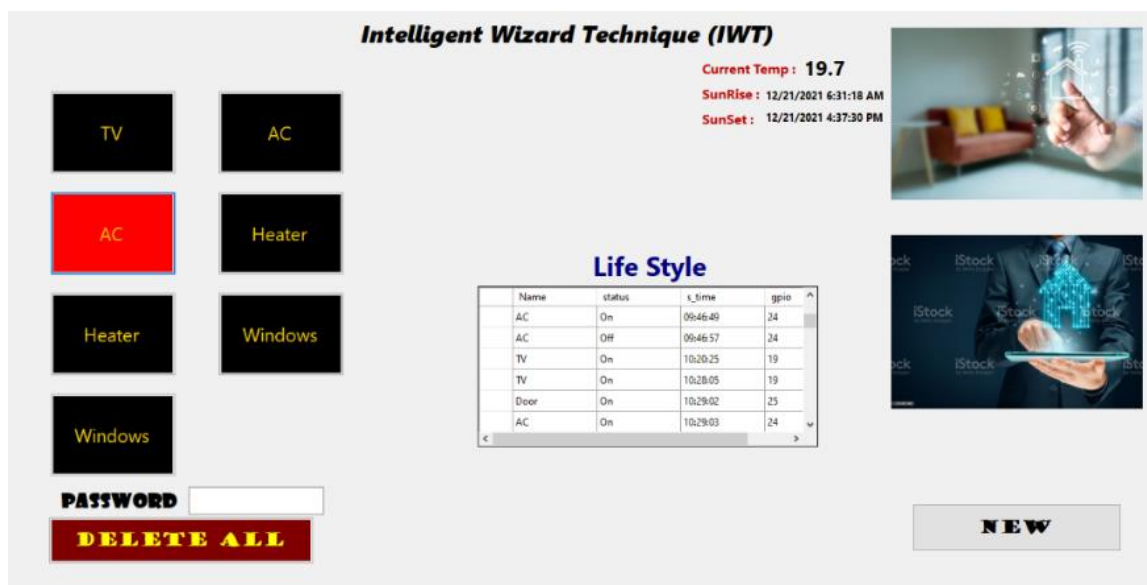



Fig 4.1: The GUI of the Smart Home Software Controller

The resulted Smart Home Software controller is very useful in monitoring and controlling the smart home environment

Fig. 4.2 illustrates the first page of the survey, which its details are given in the Appendix A of this thesis.


جامعة الإسراء
ISIRA UNIVERSITY

Smart Home Software Composer Survey

Please take a moment to help us improve your experience at smart home software composer. When you're done, please give us your feedback and provide us your suggestions to enhance the work.

Evaluate the Performance

How many times did you need to manual code the smart home controller?	How many times did you need to automatically code the smart home controller?
<input type="checkbox"/> First time	<input type="checkbox"/> First time
<input type="checkbox"/> 2 times	<input type="checkbox"/> 2 times
<input type="checkbox"/> 3 times	<input type="checkbox"/> 3 times
<input type="checkbox"/> Never Succeeded	<input type="checkbox"/> Never Succeeded

Average Time

How long did it take to manually code the smart home controller ?	How long did it take to automatically code the smart home controller ?
<input type="checkbox"/> 5-10 min	<input type="checkbox"/> 5-10 min
<input type="checkbox"/> 10-20 min	<input type="checkbox"/> 10-20 min
<input type="checkbox"/> 20-30 min	<input type="checkbox"/> 20-30 min
<input type="checkbox"/> 30-40 min	<input type="checkbox"/> 30-40 min
<input type="checkbox"/> 40-60 min	<input type="checkbox"/> 40-60 min

Evaluate the Usability

Your Satisfaction rate ranging (1-10) in the smart home software composart ?	Your Efficiency rate ranging (1-10) in the smart home software composar ?
<input type="checkbox"/> 1	<input type="checkbox"/> 1
<input type="checkbox"/> 2	<input type="checkbox"/> 2
<input type="checkbox"/> 3	<input type="checkbox"/> 3
<input type="checkbox"/> 4	<input type="checkbox"/> 4
<input type="checkbox"/> 5	<input type="checkbox"/> 5
<input type="checkbox"/> 6	<input type="checkbox"/> 6
<input type="checkbox"/> 7	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 8
<input type="checkbox"/> 9	<input type="checkbox"/> 9
<input type="checkbox"/> 10	<input type="checkbox"/> 10

Created By Samer AL Haddadin

Fig. 4.2: The First Page of the Survey

The evaluation of Smart Home Software Composer has been done by using the objective measure of the performance, and the subjective measure of Usability.

Software Performance is a gauge of how well a software achieves its requirements for correctness, which is measured by using throughput or response time. Note that the response time is the necessary time to make a response to a request (a single transaction, or an end-to-end) task [1] [2].

Software Usability evaluates the ease to use, which is a **quality attribute** encompasses number of important qualities like [1] [2]:

- **The Satisfaction** that measures the pleasant of using a software
- **The Efficiency** that measures the speed of performing tasks by user
- **The Ease of Use, that measure the** user's ability to achieve essential function(s) from the first time
- **The Memorability that measures the** simplicity of reestablishing proficiency after period of software disusing.
- **The Errors**, which measures error related issues (making errors, errors' severeness, simplicity of error recovery).

These two measures were evaluated depending on the feedbacks we have got after using Smart Home Software Composer by 120 programmers, who had been asked to compose the Smart Home Software Controller manually at first, and later automatically using the Smart Home Software Composer, and later to fill a survey to get their opinion about the developed Smart Home Composer case study of the IWT.

To evaluate the performance (the objective measure), we counted the number of people who failed to compose the Smart Home Software Controller manually compared to the automatic composing of it using Smart Home Software Composer. Table 4.1 shows the number of succussed tries of the manual approach vs the automatic approach.

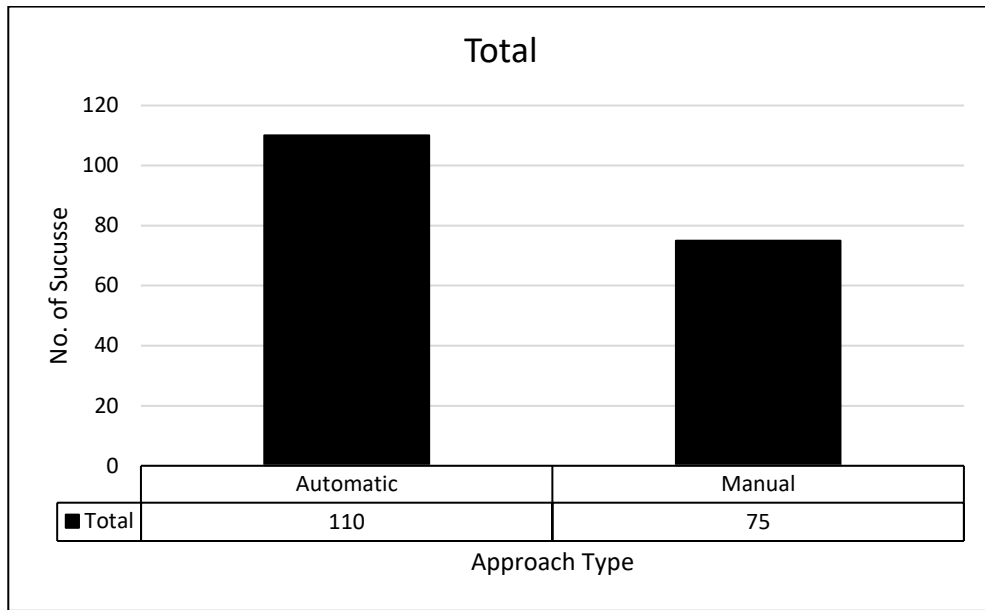


Fig 4.3: The Succeeded Tries of the Manual Approach vs the Automatic Approach

It should be noted that the "failure" may be one of three alternatives. The first one is the failing to completed coding the Smart Home Software Controller within a given time. The second one is the failing due to fail to find the correct answer to the questions of the wizard part. The third type of failure is that caused by giving the wrong answer to the questions of the wizard part. From the Table 4.1, it can be seen that fewer failures were recorded with the Smart Home Software Composer than with the manual one. The percentage of success tries using the manual approach to the success tries using the automatic approach was 68%, which is biased in favour of manual approach.

Not only the number of failings, the speed of accomplishing soundness coding had been measured also. Table 4.2 shows the average time (in minutes) spent by each programmer by using the manual approach, and the automatic approach by using the Smart Home Software Composer. It is obvious that the time taken to develop Smart Home controller by using

manual approach was much bigger than the using of automatic approach, counting the number of tries to get an error free program.

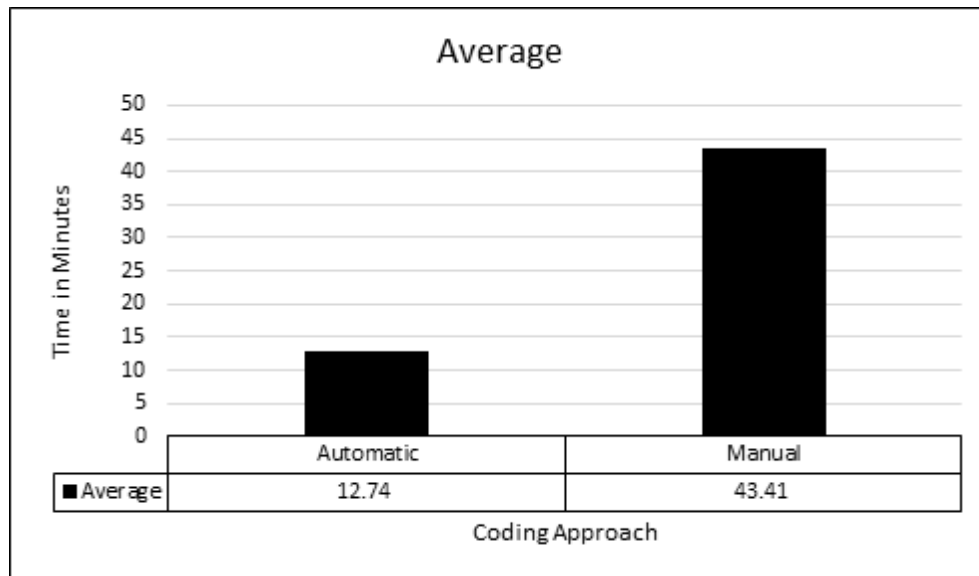


Fig 4.4: The Average Time of the Manual Approach vs the Automatic Approach

Such result is normal and justifies the need for using software tools in the development process of a software.

To evaluate the Usability (the subjective measure), each programmer of the 10 participated individuals was asked to give her/his impression of the of using the Smart Home Software Composer in terms of three factors namely **the Satisfaction, the Efficiency, and the Ease to use** in a range from 1 to 10. The results are shown in the Table 4.3. Here we can see that the Smart Home Software Composer were – in general - favourable in the responses. This is shown by the high degree given to the three criteria, especially the ease of use

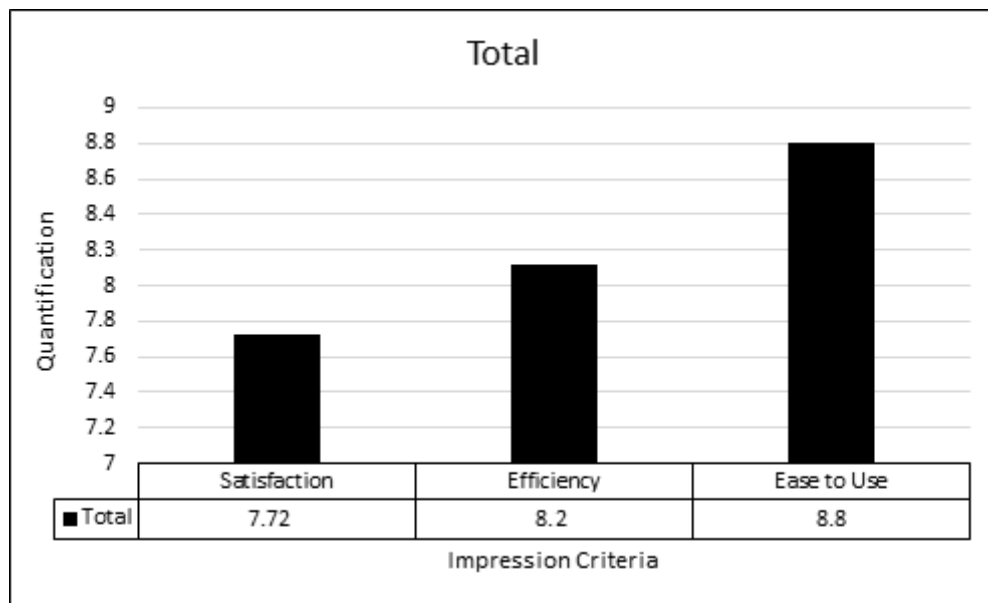


Fig 4.5: The Usability Measures of the Smart Home Software Composer

4.3. Achievements of IWT

IWT delivered number of achievements in both Software Engineering (SE) and in Artificial Intelligence (AI) areas. In general, IWT proves itself as a software tool for ASCG, and thus, it is one of other techniques for ASCG like the software tools that convert the textual description of a solution (like pseudocode for example) to source code, or the software tools that convert the diagrammatic descriptions (flowchart for example) to a source code, in addition of course to the compilers. Since it has intelligent techniques for answers acquisition in addition to the usual technique of getting answers from human, IWT pushes wizard class of software up to a higher level that make the development of software easier.

The most interesting thing to report here is that IWT opens the door towards involving machine in the performing of cognitive processes of programming, which are the problem solving. We should admit that IWT is just a beginning step for making the dream of "thinking machine" become true. Also, IWT deliver a new implementation for machine learning

principle (as we saw in the Smart Home Code Composer case study of IWT) when it records the daily uses of a human in a table that would be used later for answering certain questions.

4.4. Summery

In this chapter, an evaluation study has been accomplished by using the **Performance** objective measure, and the **Usability** subjective measure. These two measures were evaluated depending on the feedbacks that had been got from 70 programmers after their use of the Smart Home Software Composer. The feedbacks had been statistically processed the performance and usability indicators were quantified. Such evaluation approach is more realistic and can show how much is the efficiency of the proposed approach.

CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS

5.1. Conclusions

In this research, the definition of an Intelligent Wizard Technique (IWT) is produced. Based on this definition, a Smart Home Composer case study has been developed, which produced a Python source code software controller for smart home. This code was successfully running on Raspberry Pi controller that controls a house model.

The results of the evaluation show that the IWT approach scores higher in both subjective usability and objective performance when compared with the traditional manual approach to coding. Worth to note here, that the data presented here is based on a small sample (ten programmers) and should be deemed as preliminary.

5.2. Recommendations

As future work, we recommend the development of IWT as an (IDE) and supporting it with library functions. Also, making IWT able to engage with well-known design, coding, and testing software tools.

Also, we recommend supporting IWT with more intelligent techniques, including the Natural Language Processing (NLP). Such ability would facilitate the using of IWT by non-programmers.

Another recommendation that may be reported here is extend the function of the IWT to produce a diagrammatical description of the source code in addition to the textual source code.

IWT may also be integrated with other IDEs to be part of them, as much as the code composer of the user interface, which becomes a usual part of every visual programming language.

REFERENCES

- [1] R. S. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed., USA: McGraw-Hill Education, 2014.
- [2] I. Sommerville, *Software Engineering*, 9th ed., USA: Addison Wesley, 2010.
- [3] A. T. Imam, T. Rousan and S. Aljawarneh, "An expert code generator using rule-based and frames knowledge representation techniques," in *5th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 2014.
- [4] D. Méry and N. K. Singh, "Automatic code generation from Event-B models," in *The Second Symposium on Information and Communication Technology*, Hanoi, Vietnam, 2011.
- [5] N. K. Singh, "EB2ALL: An Automatic Code Generation Tool," in *Using Event-B for Critical Device Software Systems*, London, UK, Springer, 2013, pp. 105-141.
- [6] C. Geison, "AnswerLab," 20 Aug 2019 . [Online]. Available: <https://www.answerlab.com/insights/wizard-of-oz-testing>.
- [7] D. Salber and J. Coutaz, "Applying the Wizard of Oz Technique to the Study of Multimodal Systems," in *Third International Conference on Human-Computer Interaction (EWHCI '93)*, Moscow, Russia, 1993.
- [8] R. Gulndon, "Grammatical and ungrammatical structures in user-adviser dialogues: evidence for sufficiency of restricted languages in natural language interfaces to

- advisory systems,” in *The 25th annual meeting on Association for Computational Linguistics*, Stanford, California, USA , 1987.
- [9] S. Whittaker and P. Stenton, “User studies and the design of Natural Language Systems,” in *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, England, 1989.
- [10] A. Jönsson and N. Dahlbäck, *Talking to a computer is not like talking to your best friend*, Linköping, Sweden: Linköping University, 1988.
- [11] D. Diaper, “The Wizard's Apprentice: A Program to Help Analyse Natural Language,” in *The fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V*, Nottingham, U.K, 1989.
- [12] Y. Polity, J.-M. Francony, R. Palermi, P. Falzon and S. Kazma, “Recueil de dialogues Homme-machine en langue naturelle écrite,” *Les cahiers du Criss*, vol. 17, 1990.
- [13] N. Dahlbäck, A. Jönsson and L. Ahrenberg, “Wizard of Oz studies-why and how,” *Knowledge-Based Systems*, vol. 6, no. 4, pp. 258-266, 1993.
- [14] D. Mulsby, S. Greenberg and R. Mander, “Prototyping an intelligent agent through Wizard of Oz,” in *The INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993.
- [15] A. J. N Dählback, “Empirical studies of discourse representations for natural language interfaces,” in *The fourth conference on European chapter of the Association for Computational Linguistics*, Manchester, England, 1989.

- [16] J.-M. Francony, "Towards a methodology for wizard of oz experiments," in *Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.
- [17] F. L. Loaiza, D. A. Wheeler and J. D. Birdwell, "A Partial Survey on AI Technologies Applicable to Automated Source Code Generation," Institute for Defense Analyses, Alexandria, USA, 2019.
- [18] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns Elements of Reusable Object-Oriented Software*, vol. 99, New York: ADDISON-WESLEY, 1995.
- [19] R. W. Floyd, "A Descriptive Language for Symbol Manipulation," *Journal of the ACM*, vol. 8, no. 4, pp. 579--584, 1961.
- [20] D. Knuth, "Semantics of Context-Free Languages," *Mathematical systems theory*, vol. 2, pp. 127--145, 1968.
- [21] S. L. Graham, "Table-driven code generation," *Computer*, vol. 13, no. 8, pp. 25-34, 1980.
- [22] Y. Danilchenko and R. Fox, "Automated code generation using case-based reasoning, routine design and template-based programming," in *Midwest Artificial Intelligence and Cognitive Science Conference*, Cincinnati, USA, 2012.
- [23] E. Kitzelmann, "Inductive programming: A survey of program synthesis techniques," in *Third International Workshop on Approaches and Applications of Inductive Programming*, Edinburgh, UK, 2009.

- [24] W. B. Knox, S. Spaulding and C. Breazeal, "Learning from the wizard: Programming social interaction through teleoperated demonstrations," in *The 2016 International Conference on Autonomous Agents & Multiagent Systems*, Singapore, 2016.
- [25] V. Murali, L. Qi, S. Chaudhuri and C. Jermaine, "Neural sketch learning for conditional program generation," in *Sixth International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2017.
- [26] K. Christakopoulou and A. T. Kalai, "Glass-Box Program Synthesis: A Machine Learning Approach," in *Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, 2018.
- [27] A. T. Imam and A. J. Alnsour, "The Use of Natural Language Processing Approach for Converting Pseudo Code to C# Code," *Journal of Intelligent Systems*, vol. 29, pp. 1388--1407, 2020.
- [28] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," github, 21 may 2017. [Online]. Available: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>.
- [29] P. Yin and G. Neubig, "A Syntactic Neural Model for General-Purpose Code Generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017.
- [30] M. li, W. Gu, W. Chen, Y. He, Y. Wu and Y. Zhang, "Smart home: architecture, technologies and systems," in *8th International Congress of Information and Communication Technology*, Tianjin, 2018.

- [31] E. I. Davies and V. Anireh, "Design and Implementation of Smart Home System Using Internet of Things," *Journal of Digital Innovations and Contemporary Research In Science, Engineering and Technology*, vol. 7, pp. 33--42, 17 Jan 2019.
- [32] J. F. Nusairat1, "Raspberry Pi," O'Reilly online learning, 2020. [Online]. Available: https://www.oreilly.com/library/view/rust-for-the/9781484258606/html/481443_1_En_8_Chapter.xhtml.
- [33] P. F. Dubois, "Python: Batteries Included," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 7-9, 2007.
- [34] K. Milmann and M. Avaizis, "Python for Scientists and Engineers," *Computing in Science and Engineering* , vol. 13, no. 2, pp. 9-12, 2011.
- [35] P. Fabian, V. Gaël, G. Alexandre, M. Vincent, T. Bertrand, G. Olivier, B. Mathieu, . P. Peter, W. Ron and D. Vincent, "Scikit-learn: Machine learning in Python," *The Journal of machine Learning research*, vol. 12, pp. 2825-2830, 2011.
- [36] B. Ballmann, Python Basics, Uster: Springer, 2020.
- [37] A. T. Imam, A. J. Al-Nsour and A. Hroob, "The Definition of Intelligent Computer Aided Software Engineering (I-CASE) Tools," *Journal of Information Engineering and Applications*, vol. 5, no. 1, pp. 47-56, 2015.

APPENDIX A: THE SURVEY

A.1 The Structure of the Used Survey


Fig. A.1 (a) and (b) illustrate the contents of the survey that had been used to evaluate the Smart Home Software Composer case study of the IWT.

The image shows two pages of a survey form. Page (a) is the first page, featuring the logo of Irbid University (جامعة اليرموك) and the title 'Smart Home Software Composer Survey'. It includes an introductory paragraph, a section for 'Evaluate the Performance' with two columns of questions about manual and automatic coding frequency, an 'Average Time' section with two columns of questions about manual and automatic coding duration, and an 'Evaluate the Usability' section with two columns of questions about satisfaction and efficiency rates. Page (b) is the second page, containing a question about ease of use, an 'Additional Comments' section, and an 'About You (optional)' section with fields for Name, Address, Phone, and Email. Both pages end with a 'Thank you for your participation!' message and the creator's name, Samer AL Haddadin.

Fig A.1: (a) The First Page of the Used Survey
(b) The Second Page of the Used Survey

The survey had been given to 70 programmers along with copy of the Smart Home Software Composer to get their feedbacks. The programmers were of different experience and skill levels of programming using Python code. These programmers were either fresh graduates for information technology faculties, or who newly finished a programming course in Python, or professional programmers in Python. Fig A.2 Shows a sample the feedbacks got from a programmer

14


 جامعة أم القرى
 UMM AL-QURA UNIVERSITY

Smart Home Softwear Composet Surrey

Please take a moment to help us improve your experience at smart home software composer. When you're done, please give us your feedback and provide us your suggestions to enhance the work.

Evaluate the Performance

How many times did you need to manual code the smart home controller?	How many times did you need to automatically code the smart home controller?
<input type="checkbox"/> First time <input type="checkbox"/> 2 times <input checked="" type="checkbox"/> 3 times <input type="checkbox"/> Never Succeeded	<input checked="" type="checkbox"/> First time <input type="checkbox"/> 2 times <input type="checkbox"/> 3 times <input type="checkbox"/> Never Succeeded

Average Time

How long did it take to manually code the smart home controller ?	How long did it take to automatically code the smart home controller ?
<input type="checkbox"/> 5-10 min <input type="checkbox"/> 10-20 min <input checked="" type="checkbox"/> 20-30 min <input type="checkbox"/> 30-40 min <input type="checkbox"/> 40-60 min	<input type="checkbox"/> 5-10 min <input type="checkbox"/> 10-20 min <input checked="" type="checkbox"/> 20-30 min <input checked="" type="checkbox"/> 30-40 min <input type="checkbox"/> 40-60 min

Evaluate the Usability

Your Satisfaction rate ranging (1-10) in the smart home software composer?	Your Efficiency rate ranging (1-10) in the smart home software composar ?
<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input checked="" type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9 <input type="checkbox"/> 10	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input checked="" type="checkbox"/> 8 <input type="checkbox"/> 9 <input type="checkbox"/> 10

Created By Samer AL Haddadin

Your Ease to use rate rang (1-10) in the smart home software composer ?

1
 2
 3
 4
 5
 6
 7
 8
 9
 10

Additional Comments

About You (optional)

Name BAYHA YASEEN

Address _____

Phone _____

Email _____

Thank you for your participation!

Created By Samer AL Haddadin

Fig A.2: Sample of a Feedback from a Programmer

The feedbacks had been tabulated and visualized as shown by Fig A.3 (a), A.3(b), and A.3(c). These results were used to evaluate our proposed IWT as given in Chapter 5 earlier.

#	Satisfaction	Efficiency	Ease to Use
1	8	10	9
2	7	9	9
3	8	9	7
4	7	6	7
5	9	8	10
6	6	6	8
7	8	7	8
8	8	9	9
9	6	6	9
10	8	6	9
11	6	9	9
12	8	9	10
13	8	9	10
14	7	8	9
15	9	8	9
16	8	9	8
17	8	9	9
18	9	10	9
19	8	9	9
20	7	8	9
21	7	9	9
22	8	8	9
23	9	9	9
24	7	6	9
25	9	9	9
26	8	10	9
27	8	9	10
28	8	8	9
29	10	10	9
30	9	9	9
31	8	6	8
32	7	6	9
33	9	10	10
34	7	8	8
35	8	9	7
36	8	8	10
37	8	10	8
38	7	6	9
39	7	8	9
40	9	8	10
41	8	8	10
42	9	7	9
43	7	10	10
44	8	7	8
45	9	10	8
46	6	7	8
47	8	9	9
48	8	7	9
49	10	8	9
50	10	9	9
51	7	9	9
52	6	9	8
53	8	7	9
54	10	9	9
55	5	8	9
56	8	6	10
57	6	9	6
58	8	9	10
59	6	8	10
60	5	4	8
61	8	9	8
62	9	9	9
63	8	8	9
64	9	8	10
65	8	8	9
66	1	4	8
67	8	8	10
68	10	7	6
69	7	9	7
70	7	8	8
Total	7.728571429	8.11428571	8.8

(a)

#	Automatic	Manual
1	10	30
2	20	40
3	5	30
4	30	40
5	10	40
6	30	60
7	10	50
8	20	40
9	30	40
10	30	40
11	10	60
12	10	60
13	10	40
14	20	30
15	10	45
16	20	60
17	12	34
18	5	40
19	10	20
20	10	60
21	10	30
22	5	40
23	10	40
24	30	40
25	10	30
26	10	30
27	10	60
28	10	40
29	10	40
30	10	40
31	10	60
32	30	40
33	5	60
34	10	40
35	20	40
36	10	40
37	10	60
38	20	40
39	10	40
40	5	40
41	10	40
42	30	40
43	10	60
44	10	60
45	10	60
46	5	30
47	10	40
48	10	30
49	5	40
50	30	30
51	10	60
52	5	40
53	10	40
54	5	60
55	10	40
56	10	30
57	10	40
58	10	40
59	30	60
60	10	60
61	10	60
62	10	40
63	10	30
64	10	30
65	10	20
66	5	60
67	10	40
68	10	60
69	10	60
70	10	30
verag	12.7428571	43.41429

(b)

#	Automatic	Manual
1	0	1
2	0	1
3	2	1
4	0	0
5	0	2
6	0	1
7	1	0
8	3	1
9	0	0
10	0	2
11	3	1
12	0	1
13	1	0
14	3	1
15	0	1
16	0	0
17	1	0
18	0	1
19	3	1
20	2	1
21	3	2
22	0	2
23	3	1
24	3	0
25	3	1
26	0	2
27	2	1
28	3	2
29	2	0
30	0	1
31	0	0
32	0	3
33	3	2
34	3	1
35	2	2
36	3	2
37	3	1
38	3	0
39	2	2
40	2	2
41	0	2
42	0	0
43	2	2
44	3	0
45	3	1
46	3	0
47	3	2
48	2	2
49	3	0
50	3	2
51	3	2
52	2	0
53	2	1
54	0	1
55	3	1
56	0	0
57	0	2
58	2	1
59	2	2
60	0	0
61	0	2
62	3	2
63	2	1
64	2	0
65	2	1
66	1	0
67	2	2
68	1	0
69	0	1
70	2	2
Total	110	75

(c)

Fig A.3: (a) Statistics of Usability Criteria
(b) Statistics of Average Time
(c) Statistics of Performance

APPENDIX B: THE SUBMITTED PAPER

B.1. Introduction

As shown by Fig B.1 and Fig. B.2, the paper has been submitted to the International Journal of Advanced Computer Science and Applications (IJACSA) volume 13 number 1, year 2022. The IJACSA is published in the U.K with ISSN 2158-107X and eISSN 2156-5570.

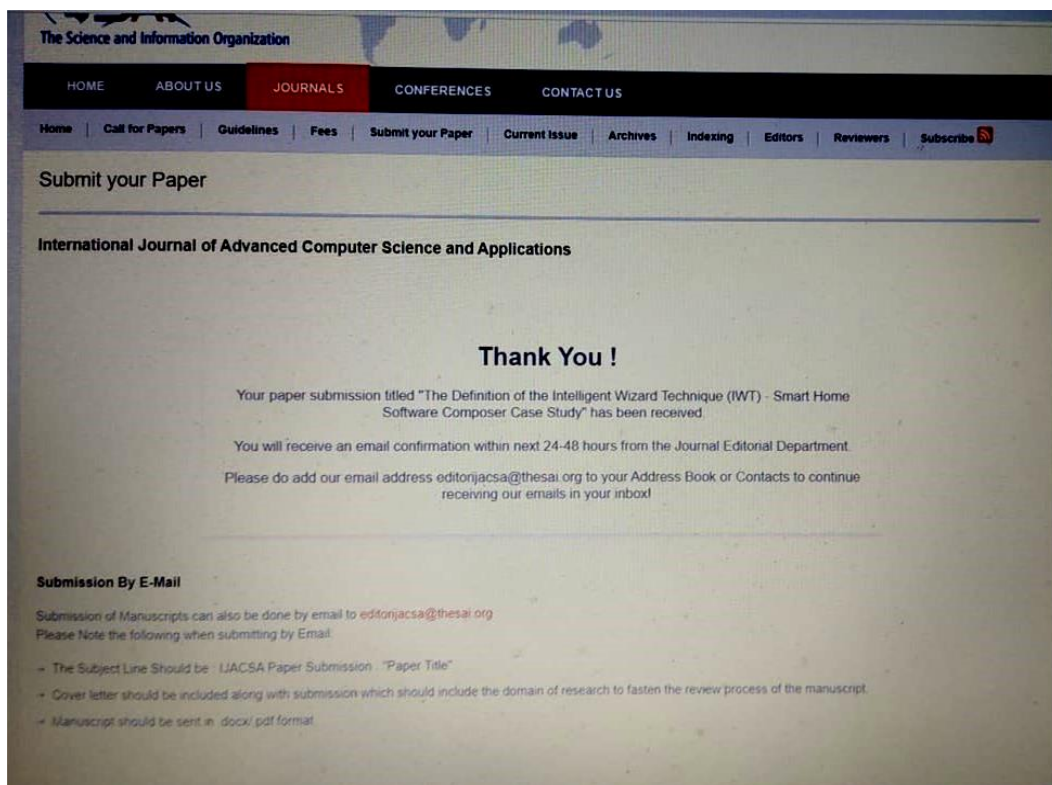


Fig. B.1: The Submission Approval

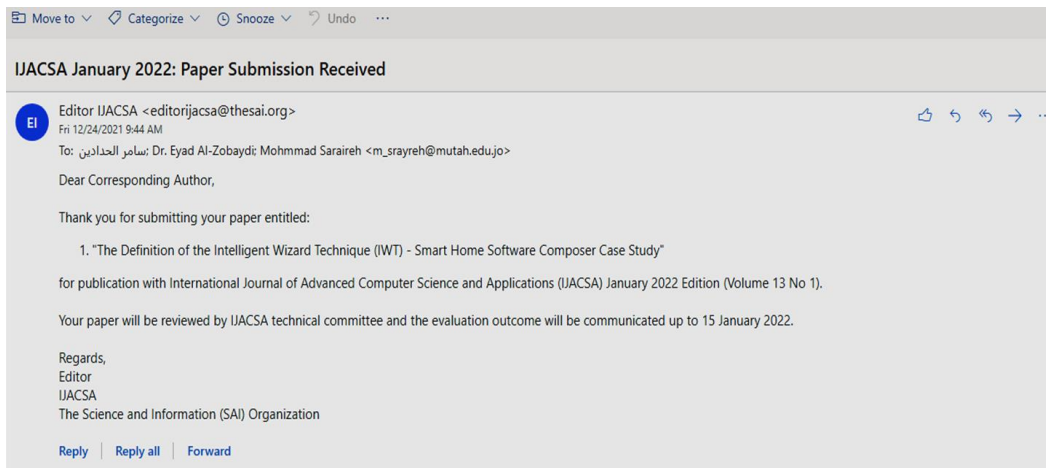


Fig. B.2: Email to Acknowledge the Reception of the Paper

B.2. The IJACSA Journal rank in Scoups and ISI Indexes

Fig. B.3 shows the Scoups rank of the IJACSA journal, which is Q3. Fig. B.4 (a, b , and c) shows the ISI rank of the IJACSA which is Emerging Source Citation Index (ESCI) with Journal Citation Index (JCI) of 0.17 for year 2020.

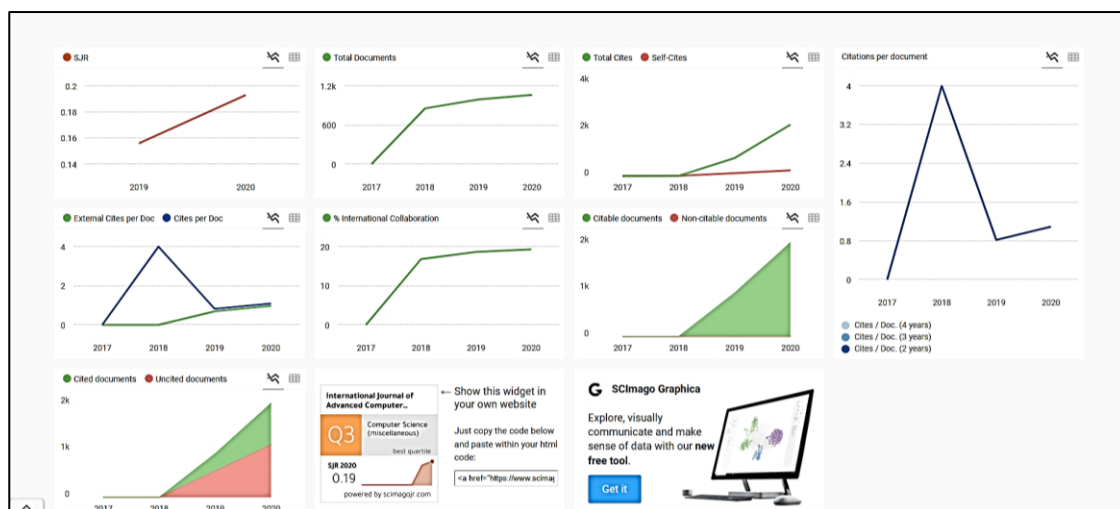


Fig. B.3: The Statistics of the IJACSA Journal on Scopus

12/27/21, 6:34 AM Web of Science Master Journal List - Journal Profile

Welcome, Ayad Tareq Imam

Web of Science Group Master Journal List Search Journals Match Manuscript Downloads Help Center Settings Log Out

Check out our new metric to help you evaluate journals! Dismiss Learn More

About

General Information

Web of Science Coverage

Journal Citation Report

Peer Review Information

Return to Search Results

INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS

Share This Journal

ISSN / eISSN 2158-107X / 2156-5570

Publisher SCIENCE & INFORMATION SAI ORGANIZATION LTD, 19 BOLLING RD, BRADFORD, WEST YORKSHIRE, ENGLAND, 00000

About

IJACSA is a high profile, leading edge platform for researchers and engineers alike to publish state-of-the-art research in the respective fields of Computer Science and Application methodologies. The journal will feature a diverse mixture of publication articles including core and applied computer science and information security related topics. See more at: <http://thesai.org/Publications>

General Information

Journal Website	Visit Site	Publisher Website	Visit Site
1st Year Published	2010	Frequency	Monthly
Issues Per Year	12	Country / Region	UNITED STATES OF AMERICA
Primary Language	English		

Web of Science Coverage

<https://mjl.clarivate.com/journal-profile> 1/3

Fig. B.4 (a): The Statistics of the IJACSA Journal on Web of Science

12/27/21, 6:34 AM Web of Science Master Journal List - Journal Profile

Collection Index Category Similar Journals

Core Collection Emerging Sources Citation Index (ESCI) Computer Science, Theory & Methods [Find Similar Journals](#)

Search a topic within this journal

Search a topic within this journal... Search

Journal Citation Report™ (JCR) Journal Citation Reports™ 2021

Journal Citation Indicator (JCI) NEW METRIC

The Journal Citation Indicator is a measure of the average Category Normalized Citation Impact (NCI) of citable items (articles & reviews) published by a journal over a recent three year period. It is used to help you evaluate journals based on other metrics besides the Journal Impact Factor (JIF).

2020	2019
0.17	0.18
Category: Computer Science, Theory & Methods	Category: Computer Science, Theory & Methods

[Learn About Journal Citation Indicator](#)

<https://mjl.clarivate.com/journal-profile> 2/3

Fig. B.4 (b): The Statistics of the IJACSA Journal on Web of Science

12/27/21, 6:34 AM

Web of Science Master Journal List - Journal Profile

Peer Review Information

Publons Partner	No, and this journal does not explicitly endorse Publons	Claimed Reviews on Publons	1,093
Public Reports on Publons	No	Signed Reports on Publons	No
Publons Transparent Peer Review Partner	No	Publons User Endorsements	88







Sign up for a free [Publons](#) account to track your publications, citation metrics, peer reviews, and editing work for this journal.

Editorial Disclaimer: As an independent organization, Clarivate does not become involved in and is not responsible for the editorial management of any journal or the business practices of any publisher. Publishers are accountable for their journal performance and compliance with ethical publishing standards. The views and opinions expressed in any journal are those of the author(s) and do not necessarily reflect the views or opinions of Clarivate. Clarivate remains neutral in relation to territorial disputes, and allows journals, publishers, institutes and authors to specify their address and affiliation details including territory.

Criteria for selection of newly submitted titles and re-evaluation of existing titles in the Web of Science are determined by the Web of Science Editors in their sole discretion. If a publisher's editorial policy or business practices negatively impact the quality of a journal, or its role in the surrounding literature of the subject, the Web of Science Editors may decline to include the journal in any Clarivate product or service. The Web of Science Editors, in their sole discretion, may remove titles from coverage at any point if the titles fail to maintain our standard of quality, do not comply with ethical standards, or otherwise do not meet the criteria determined by the Web of Science Editors. If a journal is deselected or removed from coverage, the journal will cease to be indexed in the Web of Science from a date determined by the Web of Science Editors in their sole discretion - articles published after that date will not be indexed. The Web of Science Editors' decision on all matters relating to journal coverage will be final.

Clarivate.™ Accelerating innovation.

© 2021 Clarivate [Copyright Notice](#) [Terms of Use](#) [Privacy Notice](#) [Cookie Policy](#) [Manage cookie preferences](#) [Help Center](#)

Follow us:      

<https://mjl.clarivate.com/journal-profile>


 3/3

Fig. B.4 (c): The Statistics of the IJACSA Journal on Web of Science

B.3. First Page of the Paper

(IJACSA) International Journal of Advanced Computer Science and Applications
Vol. 13, No. 1, 2022

The Definition of the Intelligent Wizard Technique (IWT)

Smart Home Software Composer Case Study

Samer Alhaddadin¹

Dept. of Software Engineering
Isra University
Amman, Jordan, Country
ad0884@iu.edu.jo

Ayad Tareq Imam²

Dept. of Software Engineering
Isra University
Amman, Jordan, Country
alzobaydi_avad@iu.edu.jo

Mohammad Sariara³

Faculty of Engineering
Mutah University
Karak, Jordan
m_saravreh@mutah.edu.jo

Abstract—While the current Computer Aided Software Engineering (CASE) tools are of notable help to the developers in composing programs, there is still a need for more flexible supporting software tools to address the raises in the complexity of composing programs. The automating of the human's intellectual activities that are required to compose a program can be the answer for such need.

This paper proposes the definition of the Intelligent Wizard Technique (IWT) as a strategy to collect answers to certain questions from sort of resources (in addition to the user as the usual wizard does) to automate the generation of source code. Based on this proposing, a new Automatic Code Generator (ACG) that can generate a Python language source code for smart home application is developed in this paper as a case study for IWT. The resulted code has been tested on a real home and the results showed the soundness of the code. IWT can be classified as an Intelligent Computer Aided Software engineering (I-CASE) tool.

The evaluation of the resulted code was achieved by using the objective measure of the performance, and the subjective measure of usability.

Keywords— Source Code Generation; Wizard; Smart Home; Raspberry Pi; Python; I-CASE

I. INTRODUCTION

Code building is a technique that is used to quickly update and develop software using Automatic Code Generation (ACG) software. ACG software is an automated process intended for normal coding tasks of software design. ACG has

great potential for developing programs in a faster way because it helps save time and effort, improve program quality, and become more accurate, and help developers get rid of boring routine tasks. Code generation technology is widely used and has helped in the development of many different types of token generators. For example, the Java decompiler (JAD) converts byte code to Java source code [1]. We cannot ignore the promising results of current ACG technologies and note that most of these techniques, especially official models, need input by humans. This is expected because the programmer's job requires innovation and creativity and is considered a creative (non-routine) job, ACG's curriculum prefers to force students to specialize in software engineers to work on non-routine jobs rather than to replace software engineering entirely. Passive code origin is a type of code generator that also produces code that needs to be modified by the programmer [1]. ACG technique is used to develop many applications like wizard.

The Wizard of Oz (WoZ) technique involves participants interacting with a system that appears to be autonomous but is really controlled by an unseen human operator in the adjacent building. [2]

Woz Systems that are currently in use. The type of extant Woz systems was created to investigate the use of natural languages in Information Retrieval (IR) systems. Experiments on telephone information services, such as phone directories, travel or train information, and reservation services, have proven fruitful [3] [4]. The experimental setup is straightforward: the wizard takes calls and acts as though

إنشاء رمز مصدر Python تلقائيًا باستخدام تقنيات الذكاء الاصطناعي

اعدت من قبل

سامر الحدادين

أشرف عليها

د. أياد طارق الزبيدي

مع

أ.د. محمد سليمان الصرايرة

الملخص

على الرغم من أن أدوات هندسة البرمجيات بمساعدة الحاسوب (CASE) الحالية تقدم مساعدة ملحوظة للمطورين في تأليف البرامج، إلا أنه لا تزال هناك حاجة إلى أدوات برمجية داعمة أكثر مرونة لمعالجة الزيادات في تعقيد برامج التأليف. يمكن أن تكون أتمتة الأنشطة الفكرية البشرية المطلوبة لإنشاء برنامج هي الحل لهذه الحاجة.

بينما يعاني المعالج التقليدي (wizard) من القدرة على جمع الإجابات من مصادر غير الإنسان، يقترح هذا العمل البحثي تعريف تقنية المعالج الذكي (IWT) كاستراتيجية جديدة لمولد الكود التلقائي (ACG) لجمع إجابات لأسئلة معينة من مصادر مختلفة (بالإضافة إلى المستخدم كما يفعل المعالج المعتاد) لأتمتة إنشاء شفرة المصدر. بناءً على هذا الاقتراح، تم تطوير دراسة حالة Smart Home Software Composer - IWT المحددة والتي يمكنها إنشاء كود مصدر بلغة Python لوحدة تحكم المنزل الذكي. تم اختبار كود Python الناتج على منزل حقيقي وأظهرت النتائج سلامة الكود. يمكن تصنيف IWT كأداة ذكية لهندسة البرمجيات بمساعدة الحاسوب (I-CASE).

تم إجراء تقييم دراسة حالة Smart Home Software Composer الخاصة بـ IWT المحدد باستخدام المقياس الموضوعي للأداء، والذي يقدر بـ 91.6٪، والمقياس الشخصي لقابلية الاستخدام، والذي يقدر بـ 85٪ للرضا، و91٪ للكفاءة و97٪ لسهولة الاستخدام حيث تظهر هذه القيم مؤشرات مفضلة للمبرمج.