Int. J. Nonlinear Anal. Appl. 14 (2023) 1, 2647–2654
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.29381.4143

Predicting and speeding up performance testing for application programs with automation

Abdulrahman Ahmed Khudhur, Ibrahim Mohamed*

Information Science and Technology Department, UKM, Malaysia

(Communicated by Ehsan Kozegar)

Abstract

Software was created as a result of the importance of measuring programmer performance. The amount of time and storage space required to implement a programmer largely determines its performance. This project used automation entrenched reliable rules to estimate the executive time. Our applied similar sophisticated criteria and software in this study to achieve the same automation results automatically and rapidly. Aside from time and storage space, the prepared software evaluates a program's performance using other criteria such as dependability, documentation, and others. All of these standards help to make sound performance judgments. This study concluded the performance testing of programmer samples written in Python as simple structures that provide a clear and easy starting point for testing programmer performance.

Keywords: Performance Testing, Application Programs, Automation
2020 MSC: 60G25

1 Introduction

Software testing is an important phase in the software development life cycle (SDLC). The program has achieved considerable usage in recent period. There aren't many devices or services nowadays which are not under the control or integrated using software [4]. Microprocessors and its software control an expanding range of operations, including autos, starting with the engine and moving up to the gearbox and brakes [5]. The dependability of the software systems used to support corporate activities or specific tasks is critical to the smooth operation of a firm or organization. One way to accomplish this is to properly review and test the developed software [6]. In order to finish the testing for this essay, we relied on certain standards, such as:

- A program's time complexity is the amount of computer time required to complete it.
- The quantity of memory required to run a programmer to completion is referred to as its space complexity.
- Software reliability is the chance of error-free software functioning in a given environment during a given time period. Another important factor influencing system dependability is software reliability [3]. In this work, we were able to build a program for testing accuracy, in which division is included in the program size and reliable mathematical terms to avoid division by zero.

*Corresponding author
Email addresses: p108993@siswa.ukm.edu.my (Abdulrahman Ahmed Khudhur), ibrahim@ukm.edu.my (Ibrahim Mohamed)

Received: November 2022 *Accepted:* January 2023

- Statistics analysis is helpful for examining a programmer or algorithm to discover, among other things, the amount of loops (for statements) and condition statements [8]. When we compare the qualifications of two programmers designed to meet the given problem, we choose the one with the fewest loops. Statistics can also help the user build software, especially when dealing with huge projects.
- Documentation is required to determine whether the project adds sufficient notes to demonstrate the operation of the program's orders or the inscription of some details referring to the program, such as the time. The written program begins, the test evolution comes to a close [9]. These remarks assist the user in comprehending the application and making it easier to use. Through accounting, the outcome ties the ratio of the program's executive lines to its remarks.

1.1 The Problem Study

The. problem of the research lies in that the failure to test software in companies before starting work leads to many problems and leads to falling into risks, including:

- Caused a lot of errors in the work system of the institution.
- Financial and moral losses to the institution due to errors.
- Damage that leads to great losses when the software is linked to medical areas and means of transportation such as cars and planes.
- Loss of time.

1.2 The Significance of Study

The significance of the research lies in the fact that testing and evaluating software in companies before starting to work with it is a clear scientific development in companies and others, but rather it is a guarantee of software quality and is the means that leads to work well. To work perfectly and avoid the risks that occur during the work of the software.

1.3 Aim of the study

- Focusing on the phenomenon of software testing, which is one of the most important factors affecting the quality of software
- This research acquires its importance due to the lack of studies that dealt with the study of the phenomenon of testing software.
- Provide suggestions and recommendations aimed at evaluating the software testing process and the benefits that accrue to institutions from software testing.

2 Related Work

This study focused on the definition of software testing and its importance and the identification of tools that are used for testing, such as the black and white box. This study defined that the idea of software testing is the most widely used method to verify and verify the quality of software. It is the procedure for implementing a program or system with the aim of finding defects for the software on the basis of the labor intensity scale and the cost, which constitutes 51% of the total cost. How can we get high quality and low cost software? What are the testing techniques? Are there tools that help us test software and is it reliable only? All these questions and more are answered in this study [2].

Software testing is a critical part of the process for high-quality software systems, and can consume more than fifty percent of the overall development effort. The majority of research on software testing has concerned the improvement of testing processes, testing standards, and the development of new techniques and tools for different types of testing. Ability and experience need to apply these technologies and tools such as personalities, education, and experience. This paper presents the views of software testers themselves as well as those of administrators and software developers, collected through an online survey method, on the importance of the formative factors influencing effective testing.

The test includes training, experience, skills, and human qualities. The survey was conducted on software testing groups on the site ”Linkedin and Yahoo” by distributing a questionnaire containing 29 questions, closed and open questions divided into eight parts [1].

Added [10] in his research some factors that affect the adoption of the test process. In addition, the results of the survey method are used to distinguish the factors that make software testing difficult in small-sized organizations. Software testing has the potential to provide assistance to improve the quality of an organization’s software because the objective of evaluation is how to meet customer requirements during software implementation. The testing activity cannot be considered an easy process due to some characteristics of the software product such as flexibility to change, complexity, and ambiguity. In addition to many factors that affect the success of the test, including:

1. Limited time
2. Limited source
3. Lack of skilled professionals
4. Insufficient knowledge of test planning, procedures, and techniques
5. Effectiveness of test specifications and requirements
6. Increased complexity of the system.

The survey method was applied to 7 companies by distributing a questionnaire to program testers, project managers and professors Universities, divided into three sections: representing the institution, defining the testing process and evaluating the factors, which included 17 factors. This study, from analyzing the data of questionnaires answered by company managers, found that 57% have official certificates in the field of industry, but they are not related to the specialization of software testing. The maturity model test is recognized by only two organizations but is not used. The most negative factor affecting the application of software testing is the lack of time. Also, the lack of the supporting tool, most of the tools that support the testing process are not free and the team’s link to the final stage of the project. budget and lack of knowledge about the benefits received from. The testing process affects the success of the business.

This study defined software testing as a formal process in which the software unit and the integrated units, and the entire program units are examined by running the programming on the computer [11]. The study also showed that tests play a pivotal role in quality assurance activities in many organizations. Finding more efficient ways to perform the most effective tests is the main challenge in testing. The study also showed that software quality has a direct relationship with testing programs, and therefore testing is an important stage in software [11].

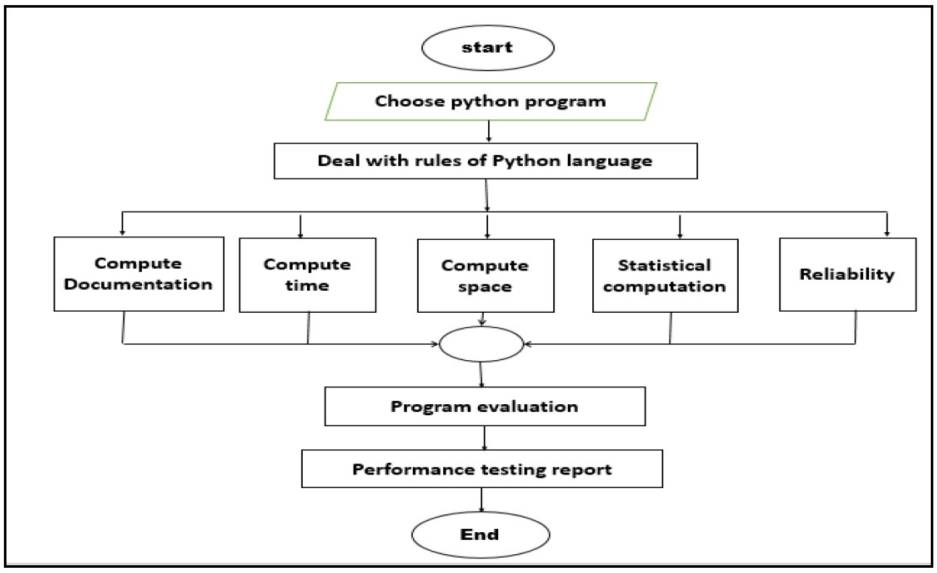


Figure 1: the proposed system’s general flow chart

3 Quality assurance, quality control, and testing

Most people get confused when it comes to the differences between quality assurance, quality control, and testing. Although they are somewhat interconnected, but there are distinct points that distinguish them. The following table lists the distinguishing points of quality assurance, quality control, and testing [7].

Quality Assurance	**Quality Control**	**Test**
Quality Assurance includes activities that ensure that processes, procedures, and standards are implemented in the context of verification of advanced programs and required requirements.	Include activities that ensure verification of advanced software with respect to documented requirements or, in some cases, not.	It includes activities that ensure bugs/error/defects are identified in the software.
Focuses on processes and procedures rather than actual testing of the system.	Focuses on actual testing through program implementation with the aim of identifying a bug/defect by implementing procedures and process.	Focuses on the actual test.
Process-oriented activities.	Product oriented activities.	Product oriented activities.
Preventive activities: a subset of the software testing life cycle.	It is a corrective process.	It is a preventive process.
	Quality control can be considered as a subset of quality assurance	Testing is the subset of quality control.

3.1 Algorithms proposed Algorithm of Executive Time

This algorithm is an accurate depiction of the guesswork or time spent implementing the programmer. We calculate time as n, where n is the magnitude of the income

Step 1: suppose that

$c =$ the time complexity of program

$n =$ size of input

$c = 0$

Step 2: Go over the programmer lines

Step 3: if program line content iteration statement: Then $c = c + (n + 1)$

Step 4: If a programmer line content iteration statement is followed by another:

$$\text{Then } c = c + n^2$$

Step 5: if program line content If-Then-Else Statement: Then $c = c + 2$

Step 6: If the programmer line content is an If-Then-Else statement followed by an iteration statement: Then $c = c + 2(n^2)$

Step 7: If the programmer line content contains Assignment Statement:

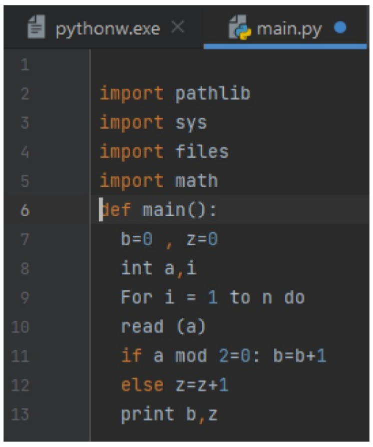
$$\text{Then } c = c + 1$$

Step 8: If the assignment statement was followed by an iteration statement, repeat the process

$$\text{Then } c = c + n^2$$

Example 1: The next program is able to be read via to intended program in order to determine its time complexity, as shown below (Fig. 2).

Lines 1, 2, 3, and 4 have no time complexity, lines 5 and 6 will be carried out once and will have a time complexity of one, line 7 will have n+1 times complexity Line 10 will now be executed (n) times, followed by either Line 11 or Line 13 being executed out (n) times, and the total execution time of line 11 and 13 will be (n) times. The overall execution time for all programmer lines will be (3+3n) iteration statement.


pythonw.exe main.py

```

import pathlib
import sys
import files
import math
def main():
  b=0 , z=0
  int a,i
  For i = 1 to n do
  read (a)
  if a mod 2=0: b=b+1
  else z=z+1
  print b,z
```

Figure 2:

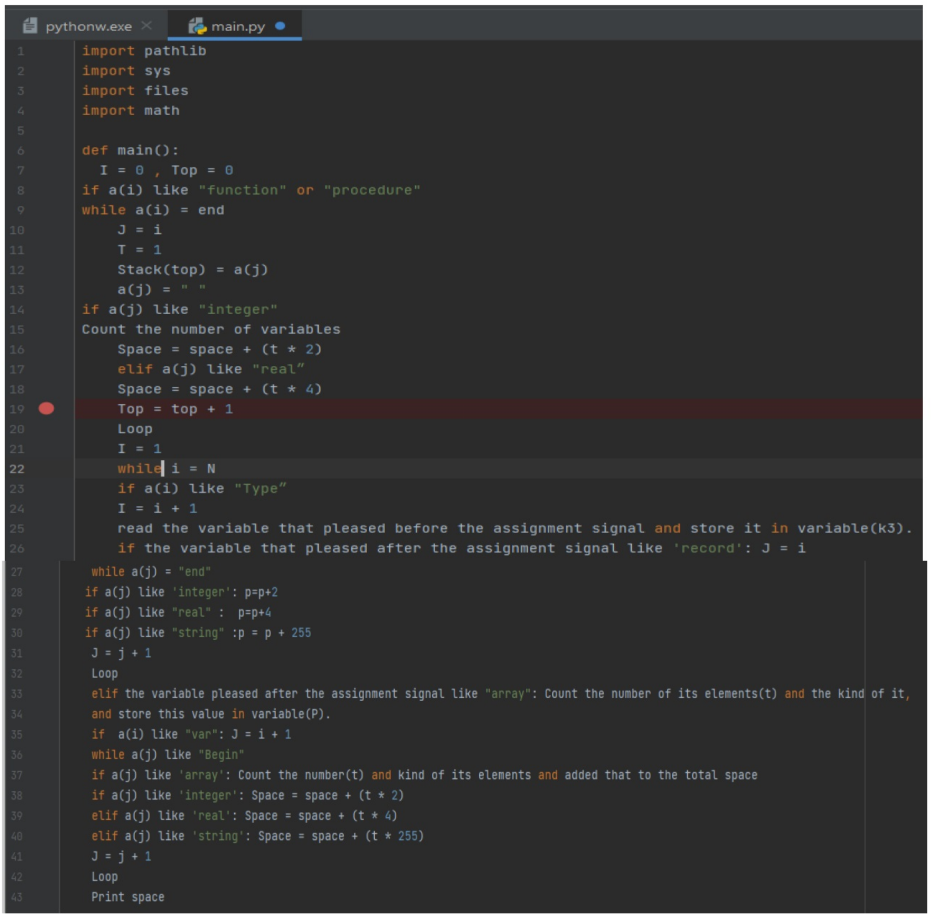
3.2 Algorithm for Space Python programmer as input

The value of spatial complexity as an output

The procedure:

Algorithm for Statistical Analysis

Python programmer as input / The output equals the number of control statements.

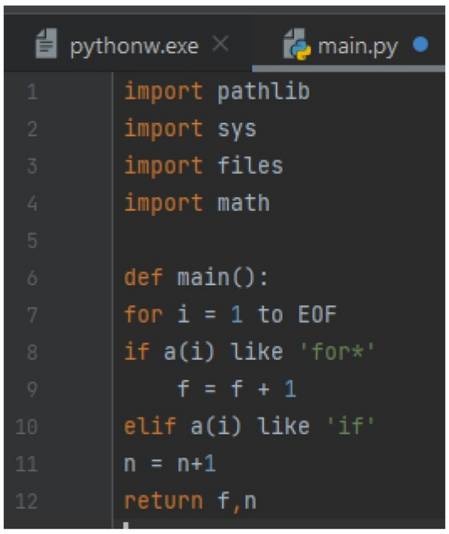

pythonw.exe main.py

```
import pathlib
import sys
import files
import math

def main():
  I = 0 , Top = 0
if a(i) like "function" or "procedure"
while a(i) = end
    J = i
    T = 1
    Stack(top) = a(j)
    a(j) = " "
if a(j) like "integer"
Count the number of variables
    Space = space + (t * 2)
    elif a(j) like "real"
    Space = space + (t * 4)
    Top = top + 1
    Loop
    I = 1
    while i = N
    if a(i) like "Type"
    I = i + 1
    read the variable that pleased before the assignment signal and store it in variable(k3).
    if the variable that pleased after the assignment signal like 'record': J = i
 while a(j) = "end"
if a(j) like 'integer': p=p+2
if a(j) like "real" :  p=p+4
if a(j) like "string" :p = p + 255
 J = j + 1
 Loop
 elif the variable pleased after the assignment signal like "array": Count the number of its elements(t) and the kind of it,
 and store this value in variable(P).
 if  a(i) like "var": J = i + 1
 while a(j) like "Begin"
 if a(j) like 'array': Count the number(t) and kind of its elements and added that to the total space
 if a(j) like 'integer': Space = space + (t * 2)
 elif a(j) like 'real': Space = space + (t * 4)
 elif a(j) like 'string': Space = space + (t * 255)
 J = j + 1
 Loop
 Print space
```

Figure 3:

The process:



```
import pathlib
import sys
import files
import math

def main():
for i = 1 to EOF
if a(i) like 'for*'
    f = f + 1
elif a(i) like 'if'
n = n+1
return f,n
```

Figure 4:

4 Algorithm of Reliability

It is one of the major standards that we add to our system since it detects situations of programmer failure due to mathematical assertions such as division on zero, and the algorithm alerts the programmer of these circumstances.

The procedure:

Input: A mathematical expression written in Python with the Infix state.

Output = A massage box that indicates whether or not the software is trustworthy.

The Method:

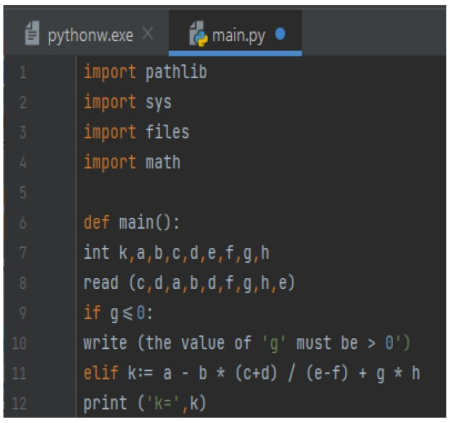
Step 1: Read the expiry and determine whether it contains the div symbol.

Step 2: Change the expression's status to postfix.

Step 3: Assign a value number to all variables in the equation except the first.

Step 4: To prevent division by zero, get the entire value of the expression.

Example 2:



```
1   import pathlib
2   import sys
3   import files
4   import math
5
6   def main():
7   int k,a,b,c,d,e,f,g,h
8   read (c,d,a,b,d,f,g,h,e)
9   if g≤0:
10  write (the value of 'g' must be > 0')
11  elif k:= a - b * (c+d) / (e-f) + g * h
12  print ('k=',k)
```

Figure 5:

Line 8 of the preceding programmer is where the division procedure is carried out. The intended program can determine whether or not the divisor expression $(e - f) + g * h$ is equivalent to zero and what the program is trying to prevent division by zero.

5 Discussion of the Findings

We obtained the findings in a table by implementing our software and testing a variety of Python as shown in table 1.

Table 1: programs characteristic

	Time complexity	**Space complexity**	**Document**	**Structure**	**Statistical analysis**
A1	$3 + 3n$	6 byte	0.16	Not structure	For s=1 if s=1
A2	$2 + 6n + 3n^2$	14 byte	$7.14 * 10^{-2}$	Not structure	For s = 3
A3	$3 + 4n + 5n^2$	50 byte	$6.6 * 10^{-2}$	Not structure	For s = 3 if s = 1
A4	$1 + n$	773 byte	0.27	Not structure	For s = 1
M1	$9 + 6n$	8 byte	0	More structure	For s = 3

We examine five distinct programs that were created to handle various challenges. As our programmer (system) succeeds in examining all of these programs one by one, their writing styles and programs tools vary. Additionally, to time. and store area, that are as in measuring the performance. of any program, we included some more requirements such as documentation, organized and statistical analysis to the table. All of this aids in the analysis and evaluation of program performance. and qualifications. As demonstrated in the table below, our software is successful in predicting time and saving space in order to perform all of these applications. Each program has a different executive time than the others. See the table 2.

Table 2: the programs characteristics

	Time complexity	**Space complexity**	**Rate of Document**	**Structure**	**Statistical analysis**
A-1	$2 + 3n^{-2}$	18	0	No functions	For s = 2
A-2	$3 + 6n + 2n^2$	18	$8.3 * 10^{-2}$	No functions	For s = 4
A-3	$2 + 2n$	18	0.333	No functions	For s = 1 ex.line -6

When the intended programmer was applied to three programmers designed to handle the same problem in various ways. The issue is determining the worth of the principal diagonal in a square matrix. For we experiment and evaluate these programs using our system, we receive various findings from the executive time, which is useful when comparing program performance see the table 3.

Table 3: the Reliability

	Time complexity	**Space complexity**	**Rate of Document**	**Structure**	**Statistical analysis**
R1	1	18	0.3333	No functions	Not reliable
R2	1	18	0.4	No functions	Reliable
R3	1	10	0.3333	No functions	Reliable

6 Conclusion

Due to the importance of the level of performance of programs, in this research we prepared a program that measures the performance and efficiency of application programs, where the level of performance depends on the time spent to implement that program and storage space. In this work, we relied on the laws of complexity adopted to estimate the time and storage space, in an automatic and rapid manner. The program has adopted other criteria in order to analyze performance, such as reliability, documentation and other statistics that help in making a decision on the efficiency of performance. In this research, the performance of samples of simple programs written in Python was tested, as it is an easy-to-understand and simple to install language, which provides a simple and obvious way to test the performance of programs.

References

[1] A.M. Alghamdi and F.E. Eassa, *Software testing techniques for parallel systems*, IJCSNS Int. J. Comput. Sci. Network Secur. **19** (2019), no. 4, 176–186.

[2] K. Gao, *Simulated software testing process and its optimization considering heterogeneous debuggers and release time*, IEEE Access **9** (2021), 38649–38659.

[3] T. Hamilton, *Software testing techniques with test case design*, https://www.guru99.com/software-testing-techniques.html, 2021.

[4] M. Ibrahim and U. Durak, S. Götz, L. Linsbauer, I. Schaefer and A. Wortmann, *State of the art in software tool qualification with DO-330: A survey*, Software Engineering (Satellite Events), Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, 2021.

[5] R.A. Khurma, H. Alsawalqah, I. Aljarah, M.A. Elaziz and R. Damaševičius, *An enhanced evolutionary software defect prediction method using island moth flame optimization*, Mathematics **9** (2021), no. 15, 1–20.

[6] F. Meng, W. Cheng and J. Wang, *Semi-supervised software defect prediction model based on tri-training*, KSII Trans. Internet Inf. Syst. **15** (2021), no. 11, 4028–4042.

[7] S. Natarajan and M. Chellam, *Study on different types of software in library and their evaluation*, Indian Highways **11** (2019), no. 4, 26–37.

[8] L. Raamesh, S. Jothi and S. Radhika, *Enhancing software reliability and fault detection using hybrid brainstorm optimization-based LSTM model*, IETE J. Res. (2022), 1–15. https://doi.org/10.1080/03772063.2022.2069603

[9] S. Shafiee, Y. Wautelet, S.C. Friis, L. Lis, U. Harlou and L. Hvam, *Evaluating the benefits of a computer-aided software engineering tool to develop and document product configuration systems*, Comput. Ind. **128** (2021), p. 103432.

[10] S. Singh Ghuman, *Software testing techniques*, IJCSMC **3** (2014), no. 10, 988–993.

[11] D.S. Taley and B. Pathak, *Comprehensive study of software testing techniques and strategies*, Int. J. Eng. Tech. Res. **9** (2020), no. 8, 817–822.