Analyses on Characteristics and Efficiency in
Opensource Programming Languages of
Undergraduates

Pardha Saradhi Byra, K. Gowri Raghavendra Narayan, Seshu Chakravarthy Thota

Abstract – A basic open source programming language (OPSL) may speak to the principal subject introduction for some
understudies. Before taking this course, understudies may have framed discernments in regards to the trouble of OPSLs from
books, online courses and other media anticipating the
innovation. Through this we breaking down what impacts
understudy adequacy of their capacity to prevail in an early on
OPSLS course, it gives Synopsys of the affiliation that exists
between understudy qualities and sentiments of viability in an
early on OPSL.

Index Terms— open source, student efficiency, characteristics

1. INTRODUCTION

Writing computer programs involves giving directions PC
to play out specific undertakings. Code, is the dialect which
can be justifiable by PC, comes in numerous varieties. The
wide decision of coding dialects can be exceptionally
confounding to a beginner. Today, programming or coding
isn’t as large an arrangement as it used to be a couple of
years back. Prior, this was an approach to offer answer for
basic issues dwelling in labs or research focuses. Without
breaking a sweat of adapting, even a school youngster can
code today, and we have seen extensive excitement from all
age bunches in figuring out how to program or code.

With the rise of OPSL, one isn’t constrained or bound
to take in a solitary dialect however one can look over a ton of
alternatives and can pick something that appropriate to one
best. Open source is a PC program or source code accessible
to end clients free of expense and they are allowed to
transform it in like manner to make it increasingly valuable
and mistake free. Instead of a solitary restrictive possession,


Table 1: Pros and Cons of OPSLs.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development and implementation cost is low</td>
<td>Support and maintenance cost may be hidden in the initial package</td>
</tr>
<tr>
<td>More easy data transferability</td>
<td>Being open to all, code may be more vulnerable to the hacker community</td>
</tr>
<tr>
<td>Potential for fast cycle time of release and bug fixes</td>
<td>Open source solutions may require additional development to enable integration with an existing proprietary environment</td>
</tr>
<tr>
<td>No limit on usage or target audience</td>
<td>Introduction of new programs/software may require staff retraining to enable them to use open source solutions</td>
</tr>
<tr>
<td>Suitable for rapid prototyping and experimentation</td>
<td>Most considering using and developing open source 'in-house' must ensure that they have the right level of expertise to manage it effectively</td>
</tr>
<tr>
<td>Opportunities for customization and innovation</td>
<td>Open source solutions may require additional development to enable integration with an existing proprietary environment</td>
</tr>
</tbody>
</table>

2. EASE OF USE

Talaga and Oh (2009), for instance, take a gander at how
Lego Mindstorm robots can be used related to Russell and
Norvig's Artificial Intelligence: A Modern Approach course
reading. LEGO handyboard robots' utilization in AI
guidance was exhibited by Greenwald and Artz (2004) (who
utilized them to show both neural and Bayesian systems)
and Imberman (2003; 2004) (who utilized them to show
neural systems). The more current LEGO Mindstorm robots
were utilized by Kumar (2004), Klassner (2002), (Schafer,
them to show hunts and master frameworks. (Klassner,
2002) utilized the robots to educate about the contrasts
among accepted and real state and additionally seek state
and slope climbing operator structures. A few critical
thinking ventures were additionally utilized without an
explicitly expressed calculation use necessity. Schafer
(2004) secured slope climbing, recreated tempering, looking
and arranging, notwithstanding fundamental mechanical

Revised Manuscript Received on December 22, 2018.

Pardha Saradhi Byra, Assistant Professor, Department of CSE,
Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India
K. Gowri Raghavendra Narayan, Assistant Professor, Department
of CSE, Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India
Seshu Chakravarthy Thota, Assistant Professor, Department of
CSE, Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India

Retrieved Number: B10360182S219/19©BEIESP
autonomy acclimation and operator idea works out. A last task (proposed by understudy colleagues) was additionally joined. Fermé and Gaspar (2007) use an infrared transmitter to enable the Mindstorm robots to be controlled continuously by a personal computer. They use the robots in an issue based learning condition for man-made brainpower office guidance.

Klassner (2006) proffers that mechanical technology and the utilization of the LISP dialect can be "simulated intelligence's October Sky" (an occasion that incites understudy intrigue and sends them on to encourage investigation all alone. Fermé and Gaspar (2007) use PROLOG and Kumar, Kumar and Russell (2006) built up a Python-based automated stage, while yet others use C and C++. Dodds, et al. (2006) demonstrate that there are unmistakably a larger number of choices than just LEGO items. They distinguish in excess of ten distinctive automated (and reproduced mechanical) stages that length from free programming to over USD $2,000. Martin (2007), then again, proffers that robots give the comprehension of the job of input (which moves toward becoming repetition to people in their regular daily existence) to understudies and enable them to comprehend the unpredictable conduct that true tasks can deliver. Various others (e.g., (Blank, Meeden, and Kumar, 2003; Dunn and Wardhani, 2003; Klassner and Anderson, 2003; Klassner and Continanza, 2007; Marín, Sanz, and Del Pobil, 2003; Weiss and Overcast, 2008)) have exhibited the adequacy of the utilization of apply autonomy outside of AI. Fagin and Markle (2003; 2002) and Duan (2003) have made structures to gauge the quantitative and subjective impact of the utilization of mechanical technology in software engineering instruction. Klassner (2006) proffers that mechanical technology and the utilization of the LISP dialect can be "simulated intelligence's October Sky" (an occasion that incites understudy intrigue and sends them on to encourage investigation all alone. Fermé and Gaspar (2007) use PROLOG and Kumar, Kumar and Russell (2006) built up a Python-based automated stage, while yet others use C and C++.

3. APPLICATION AREAS OF OPEN SOURCE PROGRAMMING LANGUAGES

Creating Web applications with PHP

On the off chance that you wish to make progressively responsive and gorgeous Web applications, PHP is the name of the amusement for you. PHP has been in the image throughout the previous 20 years and is viewed as the best Web improvement system even at this point. JavaScript doesn't have any similarity issues with any program, which implies you, can run it on any program. Additionally, it might be used for both front-end and back-end systems, filling in as an aggregate spine for your structure. There is much talk over the comparable prevalence of PHP and Java, yet PHP is more straightforward to learn and code than Java. Diverse names in the summary fuse Python, Ruby, Action Script and JavaScript, which are altogether also exceptionally astonishing yet PHP scores over them.

Use Ruby and Python when fast prototyping is required

Organizers who require some back-end programming to test their applications can rely upon Ruby and Python.
Both are dynamic and question arranged programming vernaculars with basic learning conditions. The main role behind these two being so conspicuous is their strong architect organize. A significant customer base and their synergistic undertakings have made them extraordinarily renowned. On the Internet, you can find such a substantial number of attracting Web portals and talks to guide you to take in programming from the basics to front line levels.

Python is favored when you need to learn machine programming or computerized reasoning. For all apply autonomy and human association innovation, use Python.

Build Android applications with Java

With the ascent of Android, the term ‘open source’ has turned out to be all the more generally acknowledged. At this moment, Java is the main simple to-learn and successful programming dialect for building Android applications. Regardless of whether you have a Windows or Mac machine, you can prepare your working application inside minutes and it tends to be tried on your neighborhood cell phone. To learn Java, you don’t need to know about programming terms. Indeed, even an apprentice can, with training, turn into a bad to-the-bone programming devotee.

Build iOS applications with Objective-C

On your Mac machine you can develop IOS applications impeccable to continue running on the iPhone, iPad or iPod contraptions. It uses Objective-C as the programming vernacular and this is so healthy, it enables you to manufacture irrefutable applications. At the point when appeared differently in relation to the Android organize, IOS gadgets have exceptionally restricted gadget similarity and screen varieties, so it is less demanding to manufacture an application which is good enough to keep running on a few gadgets.

Build desktop applications with Visual Basic

For speaking with the work region condition, Visual Basic and VBScript are what people have been using for a significant long time, anyway their repression is that they require unrestrained UIs. Starting late, Python and Ruby have progressed as astounding yet great programming lingo for work region application enhancement.

Learn basic coding with Scratch

On the off chance that you would prefer not to be wasted time with genuine programming terms and ideas before making something important, at that point you should begin with the Scratch apparatus. This was first presented in the US and, in the classroom, kids were locked in to instruct code. The utilization of zombies, the Angry Birds and other animation characters has made it all the more engaging and propelling to code. The apparatus is a basic fitting and-play utility with predefined conduct, and you have to gather it to get the conduct required.

Figure gives the statistics of the popularity of various programming languages among the user community.

| Table 2: popularity of various programming languages among the user community |
|--------------------------|--------------------------|
| Language | Popular Compilers |
| PHP | Wamp, Zamp, Lamp |
| Java | NetBeans, Eclipse |
| Python | PyCharm |
| Ruby | ColdRuby |

4. EXPERIMENTAL METHODS & RESULTS

A study was given to understudies in a starting python programming dialect toward the start of the semester. this study, which depended on earlier work by Ramalingam and Wiedenbeck (1998) [2] and Wilson(2002) [3], got some information about their own attributes (e.g., sex, age, significant, scholarly year, semesters enlisted, math courses taken and GPA). It at that point got some information about their confidence in their productivity to perform in the course. the initial six inquiries, from Wilson( Wilson, 2002) [3], requested that understudy describe:

Expected trouble of this class
Scale of 5 - very easy to 1 - very difficult
Composing PC programs
Scale of 5- very easy to 1 - very difficult
Working at the PC on a programming task
Scale of 5 - very relaxed to 1 - very anxious
My dimension of comprehension in this class
Scale of 5 - higher than others to 1 - lower than others
Interest in class
Scale of 5 - comfortable to 1 - afraid
Utilizing available time for help with assignments or
Course content
Scale of 5 - comfortable to 1 - uncomfortable
the next 5 questions, from ramalingam and wiedenbeck (1998) [2], asked students about particular programming constructs. These questions asked students to respond on a seven - level scale ranging from 7 - absolutely confident to 1 - not at all confident. The questions asked were:
1. Complete a programming project if someone showed me how to solve the problem first.
2. Understand the object – oriented paradigm.
3. Build my own libraries.
4. Write a program that displays a greeting message.
5. Write logically correct blocks of code.

These surveys were keyed and segmented based on four variables to detect correlations between these characteristics and survey responses. This data is presented in the following section.

5. DATA AND ANALYSIS

- The first variable for which correlations were assessed for was age, the course was separated into two groups: 18–22 (the prototypical age of college students who proceed directly from high school to college and complete college within four years) and 22–30. Additional age groups were presented to respondents, but either had no one indicate membership or an insufficient membership to facilitate analysis. Table 1 presents the correlation between age and the seven questions from Wilson (Wilson, 2002) [3] (listed fully in Section 3). Tables 2 present the correlation between age group and the statements from Ramalingam and Wiedenbeck (1998) [2]. Note that columns in the table have been highlighted to facilitate the identification of statistically significant results. Green highlighting indicates significance at $p<0.05$, yellow indicates significance at $p<0.10$ and orange indicates significance at $p<0.15$. Significance determinations have been made from values prior to rounding for presentation.

- The results presented in Table 3 indicated two statistically significant findings [1]. It appears that the younger students are comfortable, working at the computer and feel more confident in reaching a high level of understanding in the class.

- In table 4, five statistically significant correlations are presented [1]. Interestingly, in all cases the younger group of students exhibits a superior level of confidence.

- In Table 5, only one statistically significant correlation is demonstrated [1]. Senior-year students demonstrate a greater belief that they will participate in class.

- In Table 6, five statistically significant correlations [1] are shown, including four at $p<0.05$.

- In Table 7, four statistically significant correlations [1] are presented, including three at $p<0.05$. Interestingly, the lower-GPA students evidence a greater belief level in their ability to write computer programs and that they will participate in both the class and the lab. The higher-GPA students indicate an expectation of having a higher level of understanding. Notably, GPA is the only variable that shows a statistically significant correlation with the confidence in writing computer programs and lab participation.

- In Table 8, one statistically significant correlations [1] are presented. This includes question 9 (understand object-oriented) Notably, GPA is the only variable shown to have a correlation.

- In Table 9, two statistically significant correlations [1] are presented. Students who self-learned programming expect the class to be more difficult, but they also expect to gain a higher level of understanding of the materials presented. Notably, self-learning status is the only variable showing a statistically significant correlation with expected difficulty and the only one shown a statistically significant correlation at $p<0.05$ for the level of understanding.

- In question 3, they indicate a greater belief in their understanding of the object-oriented programming approach. Question 5 have younger students exhibiting their confidence in writing logically - correct code, understanding language structure and writing a program in an "only vaguely familiar" area.

- In question 2, the senior-year students indicate greater confidence in their efficiency to write a program using only the help functionality, complete a program using only a language manual and build a library.

- In question 1, the senior-year students indicate greater confidence in their efficiency to complete the programming project if someone showed how to solve the problem first.

- In question 4, the both senior and younger group students write the program with high level of confidence. Efficiency.

6. CONCLUSIONS AND FUTURE WORK

This paper has displayed starting work on distinguishing proof of understudy attributes which relate with apparent adequacy/investment in class exercises, viability and expected execution. It is additionally impractical to, from the information gathered; disambiguate the choice factor of understudies undertaking the upper dimension OPSL course from other upper dimension understudies. Indeed, a significant number of these relationships may appropriate to any (or most) upper dimension courses.

Future work will incorporate gathered extra information with respect to the qualities that make understudy sentiments of adequacy and capacity to perform. This will be gathered over numerous contributions of the course and at various organizations. A comparative information gathering component will be used for extra upper dimension courses to survey what connections are explicit to OPSL and which are all the more for the most part material. At last, this investigation will proceed to asses’ understudy execution and thoroughly analyze it with understudy desires and feeling of productivity.

REFERENCES


24. X. Daan. Outcomes assessment of student learning for an educational robotics program.


Analsys on Characteristics and Efficiency in Opensource Programming Languages of Undergraduates

### TABLE 3. CORRELATION BETWEEN AGE GROUP AND EXPECTED CLASS ACTIVITIES.

<table>
<thead>
<tr>
<th>Age</th>
<th>Writing Computer Difficulty</th>
<th>Writing Computer Programs</th>
<th>Working at Computer</th>
<th>Level of Understanding</th>
<th>Participation in Class</th>
<th>Participation in Lab</th>
<th>Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-22</td>
<td>3.01</td>
<td>3.51</td>
<td>4.23</td>
<td>3.52</td>
<td>2.76</td>
<td>3.26</td>
<td>3.02</td>
</tr>
<tr>
<td>22-30</td>
<td>3.01</td>
<td>3.61</td>
<td>3.61</td>
<td>3.03</td>
<td>3.21</td>
<td>3.41</td>
<td>3.45</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.51</td>
<td>0.41</td>
<td>0.15</td>
<td>0.01</td>
<td>0.19</td>
<td>0.39</td>
<td>0.25</td>
</tr>
</tbody>
</table>

### TABLE 4. CORRELATION BETWEEN AGE GROUP AND METRICS OF EFFICACY / EXPECTED PERFORMANCE.

<table>
<thead>
<tr>
<th>Age</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-22</td>
<td>6.81</td>
<td>6.51</td>
<td>5.10</td>
<td>7.10</td>
<td>7.01</td>
</tr>
<tr>
<td>22-30</td>
<td>6.80</td>
<td>6.00</td>
<td>5.50</td>
<td>7.10</td>
<td>6.61</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.25</td>
<td>0.15</td>
<td>0.23</td>
<td>N/A</td>
<td>0.10</td>
</tr>
</tbody>
</table>

### TABLE 5 PRESENTS THE CORRELATION BETWEEN THE ACADEMIC YEAR AND THE QUESTIONS FROM WILSON (2002)[3].

<table>
<thead>
<tr>
<th>Year</th>
<th>Expected Difficulty</th>
<th>Writing Computer Programs</th>
<th>Working at Computer</th>
<th>Level of Understanding</th>
<th>Participation in Class</th>
<th>Participation in Lab</th>
<th>Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior</td>
<td>3.10</td>
<td>3.61</td>
<td>4.01</td>
<td>3.41</td>
<td>2.81</td>
<td>3.41</td>
<td>3.01</td>
</tr>
<tr>
<td>Senior</td>
<td>3.10</td>
<td>4.10</td>
<td>4.01</td>
<td>3.26</td>
<td>3.51</td>
<td>3.51</td>
<td>3.26</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.51</td>
<td>0.23</td>
<td>0.511</td>
<td>0.35</td>
<td>0.06</td>
<td>0.41</td>
<td>0.38</td>
</tr>
</tbody>
</table>

### TABLE 6 CORRELATIONS BETWEEN ACADEMIC YEAR AND METRICS OF EFFICACY / EXPECTED PERFORMANCE.

<table>
<thead>
<tr>
<th>Year</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior</td>
<td>6.61</td>
<td>6.41</td>
<td>5.00</td>
<td>7.01</td>
<td>7.10</td>
</tr>
<tr>
<td>Senior</td>
<td>6.76</td>
<td>5.76</td>
<td>5.75</td>
<td>7.01</td>
<td>6.50</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.35</td>
<td>0.06</td>
<td>0.13</td>
<td>N/A</td>
<td>0.09</td>
</tr>
</tbody>
</table>

### TABLE 7. CORRELATION BETWEEN GPA AND EXPECTED CLASS ACTIVITIES.

<table>
<thead>
<tr>
<th>GPA</th>
<th>Expected Difficulty</th>
<th>Writing Computer Programs</th>
<th>Working at Computer</th>
<th>Level of Understanding</th>
<th>Participation in Class</th>
<th>Participation in Lab</th>
<th>Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0-2.99</td>
<td>3.01</td>
<td>4.1</td>
<td>4.101</td>
<td>3.02</td>
<td>4.11</td>
<td>4.01</td>
<td>3.50</td>
</tr>
<tr>
<td>3.0-3.99</td>
<td>3.02</td>
<td>3.51</td>
<td>4.11</td>
<td>3.35</td>
<td>2.69</td>
<td>3.18</td>
<td>3.17</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.51</td>
<td>0.05</td>
<td>0.61</td>
<td>0.11</td>
<td>0.001</td>
<td>0.03</td>
<td>0.43</td>
</tr>
</tbody>
</table>

### TABLE 8. CORRELATIONS BETWEEN GPA AND METRICS OF EFFICACY / EXPECTED PERFORMANCE.

<table>
<thead>
<tr>
<th>GPA</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0-2.99</td>
<td>7.52</td>
<td>5.51</td>
<td>5.01</td>
<td>7.02</td>
<td>6.61</td>
</tr>
<tr>
<td>3.0-3.99</td>
<td>6.69</td>
<td>6.51</td>
<td>5.18</td>
<td>7.21</td>
<td>7.11</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.41</td>
<td>0.14</td>
<td>0.32</td>
<td>N/A</td>
<td>0.25</td>
</tr>
</tbody>
</table>

### TABLE 9. CORRELATION BETWEEN SELF-LEARNING PROGRAMMING AND EXPECTED CLASS ACTIVITIES.

<table>
<thead>
<tr>
<th>Self-learned programming</th>
<th>Expected Difficulty</th>
<th>Writing Computer Programs</th>
<th>Working at Computer</th>
<th>Level of Understanding</th>
<th>Participation in Class</th>
<th>Participation in Lab</th>
<th>Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>2.76</td>
<td>3.50</td>
<td>4.00</td>
<td>3.01</td>
<td>3.01</td>
<td>3.26</td>
<td>3.01</td>
</tr>
<tr>
<td>Y</td>
<td>3.18</td>
<td>3.83</td>
<td>4.00</td>
<td>3.51</td>
<td>3.01</td>
<td>3.34</td>
<td>3.34</td>
</tr>
<tr>
<td>P-Val</td>
<td>0.11</td>
<td>0.23</td>
<td>0.50</td>
<td>0.05</td>
<td>0.51</td>
<td>0.45</td>
<td>0.28</td>
</tr>
</tbody>
</table>