JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Advanced Technique For Enhancing Computer Programming Learning Based On Content-Based Recommendations

Dr. Dileepkumar Padidem,
Prof Dept. of CSE,
RSR Engineering College,
Kadanutala, Nellore AP,

Kandlagunta Ramesh
Assoc .Prof Dept of CSE,
RSR Engineering College,
Kadanutala, Nellore AP,

T.Lakshmi Prasanna,
Asst. Prof Dept of CSE,
Narayana Engineering College,
Nellore AP,

Abstract: *Learning computer programming is a challenging process. Visual Programming Language (VPL), like Scratch have shown very promising results for beginners to overcome this challenge. Interestingly, some higher-education institutions have started to use VPLs to in CS1 courses which is referred as introduction to programming courses. However, an important issue regarding Scratch's usage in higher education environments is that students may feel unmotivated being confronted by programming exercises that do not fulfill their individual expectations. In existing work, CARAMBA which is an extension of Scratch including exercise recommender system. This is based on the features like taste and complexity, CARAMBA is able to personalize student learning with Scratch by suitably suggesting exercises for students.The proposed work aims at Recommendation of controlled exercises which will study to build recommendations in which topics are taught to improve students learning based on Content-Based Recommendation systems. These systems recommend items with similar characteristics to those already evaluated by a user and to incorporate the statistical details and to improve user interface.*

Keywords: Visual Programming language, Laboratory Virtual Instrument Engineering Workbench

1. INTRODUCTION

Visual Programming Language (VPL) has shown prominent results for beginners to learn programming. Some institutions have started VPL in CS1 courses which is referred as Introduction to Programming Courses. Scratch is one of the VPL which is used in higher education environment and this scratch is not able to fulfill the individual expectations of the students. This paper presents the results of a random sample of the full data set, which indicates that LabVIEW programmers are confident that the advantages of the visual programming provided by LabVIEW outweigh its disadvantages. Each of these surveys is the first study to investigate the opinions about visual programming of their respective subject populations. Taken together, these three provide the most comprehensive analysis of current opinions toward visual programming to date.

In the first survey, an examination of the VP literature revealed a range of statements made by VP academics about the ways in which they think that visual programming assists a programmer's thought processes. These statements have been grouped into categories. In the second and third projects, the surveys administered to the programmers and LabVIEW programmers collected the same types of opinions by asking them open-format questions about how visual programming affects their thought processes. Additionally, these questionnaires asked respondents to rank the relative advantages of textual and visual programming languages along several dimensions. Thus, the questionnaires provided a larger range of opinion from each respondent, and more analysis is possible on the resulting data. These three surveys of opinions are valuable in two ways. First, the survey of the VP literature outlines the extent of the evidence problem for the VP community. Even more importantly, some of the opinions provide good directions for subsequent research. In general, people's perceptions of their cognitive processes are not necessarily accurate; thus, a person's claims as to how and why a VPL impacts the programming process cannot be accepted as fact without further empirical study. However, people's opinions do provide a starting point: the claims of the researchers are their predictions reached after contemplating the programming process; the programmers in the second survey provide the speculations of experienced programmers; and the LabVIEW programmers provide the insight of people whose opinions have been based on experience using VPLs. In brief, the collected opinions may provide hypotheses for future, targeted empirical studies.

2. LITRATURE SURVEY

As a field, visual programming encompasses a wide variety of suggested visual notations. The Project 1 and 2 surveys are general in that neither limited its focus to one VPL. Project 1 involved collecting all opinions in the VP literature about how visual representations might impact programming, and the Project 2 questionnaire described visual programming generally as the use of a language in which "the programmer does almost all programming by manipulating diagrams instead of typing text." In contrast, Project 3 limited its inquiry to LabVIEW. LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a programming environment that features a dataflow-based VPL (called G) which was designed to facilitate development of data acquisition, analysis, display and control applications. Moreover, one of LabVIEW's marketing claims is that LabVIEW is so usable that it is an effective tool not only for trained programmers, but also for certain types of end users. In particular, LabVIEW is described as usable by scientists and engineers who possess limited programming experience, yet who need software to interact with laboratory equipment. One of the few commercially-available VPLs, LabVIEW has enjoyed relatively wide success compared to other VPLs, the majority of which are research prototypes.

LabVIEW is an attractive choice for an empirical study of visual programming for several reasons. For example, the fact that LabVIEW has been commercially available for 10 years means that one can find a sizable population of people who have used. Thus, given the desire to study an expert population, LabVIEW becomes a natural candidate. Moreover, within the fairly small number of empirical studies that have investigated visual representations in programming, two prior studies of LabVIEW pose some interesting questions.

One of these studies is an industry-based, observational study reported by Baroth and Hartsough (1995) which describes their organization's experience using two VPLs (LabVIEW and VEE) to build test and measurement systems. Of particular interest is their case study which compared the progress of two development teams (one using LabVIEW and the other using the textual language C) who were, in parallel, each developing the same system. The goal was to discern how LabVIEW would compare to conventional, text-based programming. Both development teams received the same requirements, the same funding and the same amount of time (three months) to complete the project. At the end of the allotted time period, the C team had not achieved the original project requirements, yet the LabVIEW team had gone beyond the requirements. From this case study and from experience using LabVIEW and VEE on over 40 other projects over the course of three years, Baroth and Hartsough enthusiastically report performance benefits for both VPLs. They report that projects using LabVIEW and VEE require 4 to 10 times less time to complete than if attempted using a textual programming language. Baroth and Hartsough attribute the benefits of the two VPLs, in large part, to the visual representations of the VPLs. With respect to the case study, Baroth and Hartsough attributed the results in large part to LabVIEW's visual syntax. They claim that LabVIEW's visual notation is more readable for certain classes of end users than traditional textual programming languages. Because LabVIEW's visual syntax produces programs resembling wiring diagrams, Baroth and Hartsough claim that LabVIEW is relatively easily learned by engineers and scientists who are familiar with wiring diagram notations.

In contrast, a controlled study by Green, Petre and Bellamy revealed no benefits resulting from LabVIEW's visual notations for conditional logic (Green, Petre & Bellamy, 1991, Green & Petre, 1992). Their controlled experiment compared the comprehensibility of LabVIEW's conditional representations (LabVIEW provides two ways to represent conditional logic) to two different textual notations. Green et al. tested comprehension in terms of response time required to answer questions about code segments. Their subjects were 11 experienced programmers: Five had used LabVIEW in their work for at least six months, while the other six were advanced digital electronics designers and, thus, experienced in using electronics schematics. LabVIEW elicited worse performance (i.e., longer response times) on all of the comprehension questions. In fact, the text outperformed LabVIEW for each and every subject (each subjects' mean visual versus textual times were compared).

3.EXISTING WORK

In the existing work, CARAMBA is used and it is an extension of scratch that includes exercise recommendation system. It is able to personalize student learning with Scratch by suggesting suitable exercises for students. The results of the exercises confirm that recommender exercise has a positive effect on student learning programming languages.

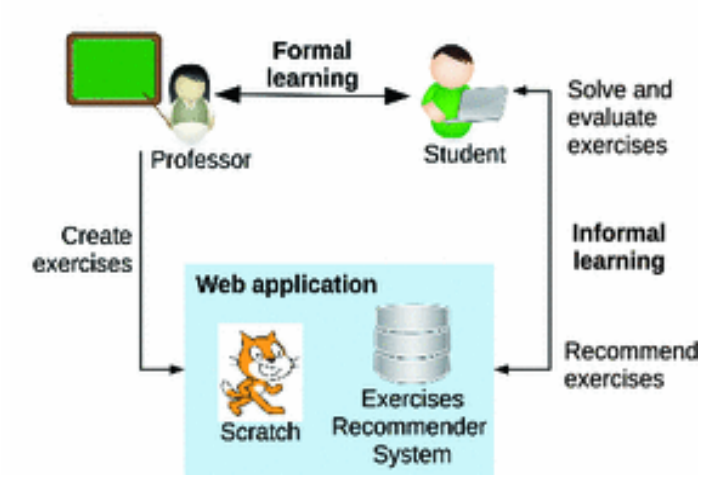


Figure 1: Existing work architecture

4. PROPOSED WORK

CollaborativeFilteringapproach was used in the existing work. The proposed work aims at recommendation of controlled exercises to improve students learning based on Content – Based Recommendation systems. It also aims to incorporate the statistical details and to improve user interface.

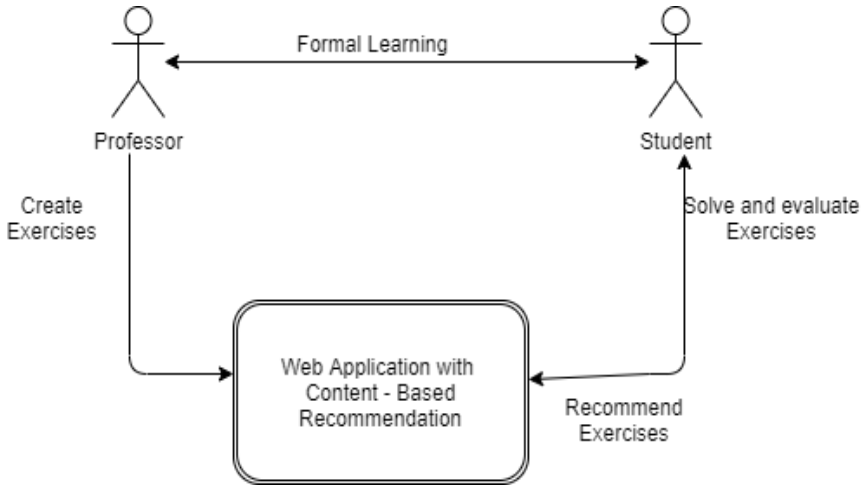


Figure 2: Proposed Work Architecture

1. User first registers with the simple data like name, address, email id, mobile no, standard etc.

2. Registered user login into the exercise to attempt the randomized exercises and then results are displayed according to the capability of the user.

3. Results are displayed as next which level is recommended.

4. Then lectures are recommended according to the result.

5. METHODOLOGY

Content-based filtering refers to recommend items based on contents available from user data.

a Content based filtering is applied on the skill and ability of the users to generate a list of recommended videos[fig(7)] and to recommend the next processing level[fig(6)] of the user.

b The questions in each level are randomized.

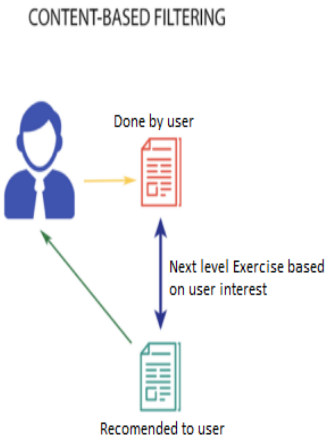
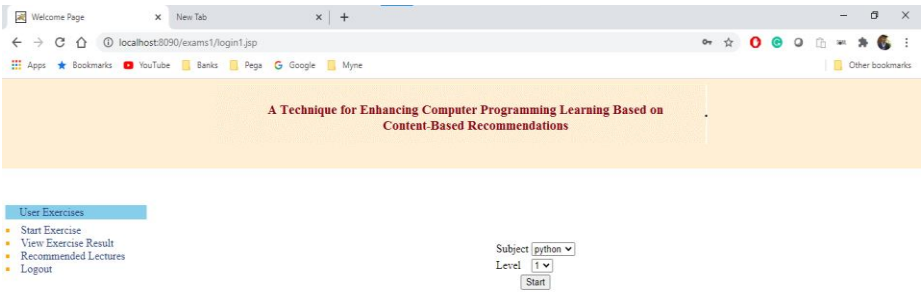


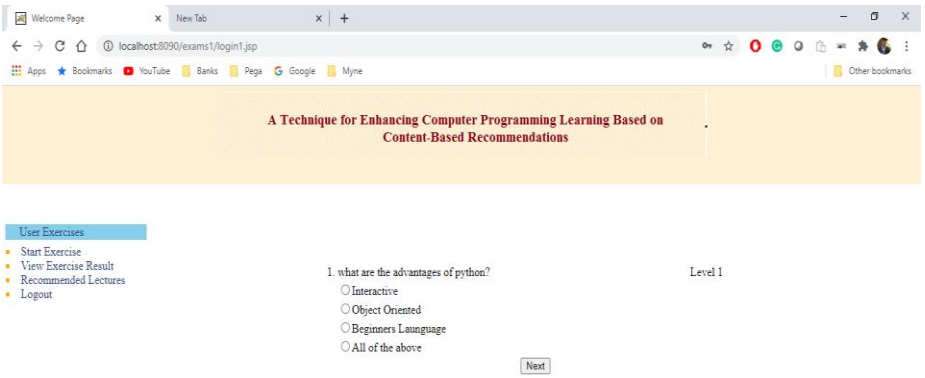
Figure (3):

Algorithm :

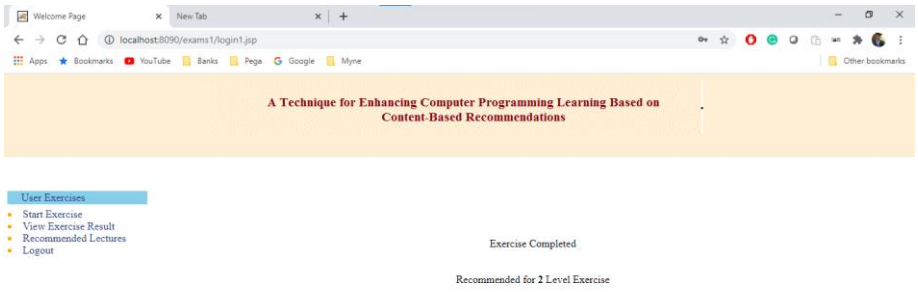
Step1: input users u=u1,u2,…..un
Step2: subjects s1,s2,…..sn
for each subject three levels l=l1.....
for each level, n number of questions q=q1,q2,…..qn.
Step3: for each user attempts exercise e1,…..en.
if new user, exercise from first level l1
else exercise from selected level
Step4: if user ui clears the level li,
ui will be eligible for level li+1 and user ui will be recommended with li+1 lecture
else ui will be recommend with li lectures



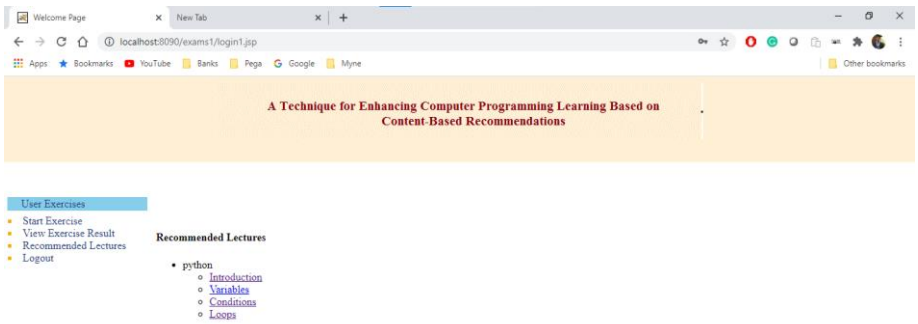
Fig(4): User Exercise



Fig(5): Exercise Questionnaire



Fig(6): Next Level Recommendation



Fig(7): Recommended Lectures

6. RESULT

From this experiment we can analyse the types of users and their ability to learn programming languages. We present a bar chart describing the set of increased number of users in content-based recommendation compared to the previous results in collaborative recommendation.

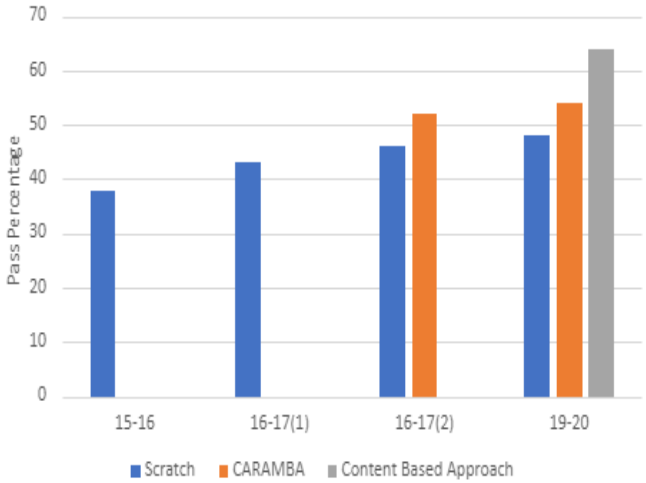


Fig: Graphical Analysis

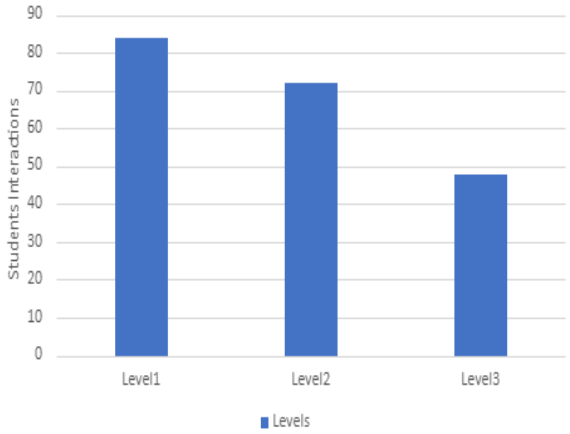


Fig: Level Analysis

7.CONCLUSION

The proposed system simplifies the learning process of programming languages by using exercises based on content-based approach.This helps in recommendation of lectures and other level exercises according to the interest and skill of the user and improved in user interface..

REFERENCES

[1] "Recommender Systems and Scratch : An Integrated Approach for enhancing Computer Programming Learning", Jesennia C´ardenas-Cobo, Amilkar Puris, Pavel Novoa-Hern´andez, Jos´e A. Galindo and David Benavides , IEEE Transactions on Learning Technologies,2019.

[2] Huet, O. Pacheco, J. Tavares, and G. Weir, "New challenges in teaching introductory programming courses: a case study," in 34th Annual Frontiers in Education, 2004. FIE 2004. IEEE, 2004, pp. 286– 290.

[3] H. C. Jiau, J. C. Chen, and K.-F. Ssu, "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," IEEE Transactions on Education, vol. 52, no. 4, pp. 555–562, nov 2009.

[4] T. M. Connolly, M. Stansfield, and T. Hainey, "An application of games-based learning within software engineering," British Journal of Educational Technology, vol. 38, no. 3, pp. 416–428, 2007.