Heliyon

Received:
4 July 2018
Revised:
12 January 2019
Accepted:
11 February 2019

Cite as: Manar Abu Talib, Omar Einea, Qassim Nasir, Mohamad Fouzi Mowakeh, Mohamed Eltawil. Enhancing computing studies in high schools: A systematic literature review & UAE case study.
Heliyon 5 (2019) e01235.
doi: 10.1016/j.heliyon.2019.e01235

Enhancing computing studies in high schools: A systematic literature review & UAE case study

Manar Abu Talib [a], **Omar Einea** [a], **Qassim Nasir** [b], **Mohamad Fouzi Mowakeh** [b], **Mohamed Eltawil** [b,*]

[a] *Department of Computer Science, University of Sharjah, Sharjah, United Arab Emirates*

[b] *Department of Electrical and Computer Engineering, University of Sharjah, Sharjah, United Arab Emirates*

[*] Corresponding author.

E-mail address: mtalib@sharjah.ac.ae (M. Eltawil).

Abstract

Open source software (OSS) is increasingly being integrated into educational institutions, and many countries require the use of OSS in government departments. However, not much focus is placed on integrating it into the educational sector in a strategic and productive manner. This paper examines the existing literature on the use of OSS in terms of the potential enhancements it can provide for computer science studies in high schools in general, and those in the UAE more specifically. It also details a survey conducted among 400 high school teachers after teaching them about multiple types of OSS that might enhance their teaching experience. After examining more than 69 different research papers and taking the survey findings into account, we drafted a roadmap that can be used by any educational institute—especially high schools—to strategically integrate OSS into the educational system.

Keywords: Computer science, Education

https://doi.org/10.1016/j.heliyon.2019.e01235

1. Introduction

Increasing the enrollment of students in computer science related fields has been the focus of many research projects in the past. The integration of open source software (OSS) in educational institutions can be useful in enhancing computing studies and motivating students to enter the computer science field. However, this has not been extensively discussed in the literature, nor has a clear roadmap of these potentials been developed. Students, teachers and faculty often have little knowledge of Open Source Software and are unaware of its benefits and potential uses. As a result, instructors typically have no experience dealing with or developing OSS. Moreover, switching from traditional programs into new programs is a real challenge for educators. On the other hand, using open source software in schools and universities should be encouraged. Many educators find them to be very useful for education and learning. In addition to the educational benefits, OSS is available at no cost to educators. "There is no reason why colleges and universities cannot teach production programming in the classroom". Students who are just beginning to use Linux systems have reported satisfaction and a relatively smooth learning curve. Students also showed remarkable patience and tended to accept the technical difficulties introduced by the open source software. OSS gives students a better appreciation for the usefulness of computing, and allows them to understand how their projects can impact the world (Daloukas et al., 2008; Dias et al., 2012; Dorodchi and Dehbozorgi, 2016; Hislop et al., 2009; Izbicki and Mike, 2016; Morelli and de Lanerolle, 2009; Smith et al., 2014; Terbuc, 2006; van Gumster and Jason, 2007).

The UAE has begun the process of overhauling their educational system, and especially their Computer Science curriculum. Our research takes advantage of this moment of change in the UAE to demonstrate the potential of Open Source Software (OSS) to improve the curriculum.

We will begin by taking a close look at OSS usage in academia. Fig. 1 shows how we conducted the research presented in this paper, including an overview of our procedures for performing systematic reviews (Kitchenham, 2004; Jesson et al., 2011; Booth et al., 2016).

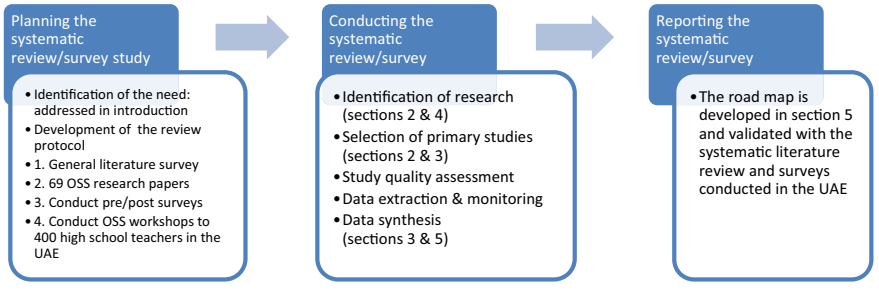


Fig. 1. Methodical planning.

https://doi.org/10.1016/j.heliyon.2019.e01235

2. Related work

2.1. Systematic literature review

Increasing the enrollment of students and enhancing their education in computer science-related fields has been the focus of much research in the past. This issue persists in many universities all over the world, according to Abu Talib (Talib and Elnagar, 2015). However, while literature exists on this topic, it is fragmented and has not been synthesized and analyzed to find overarching results.

The focus of this paper is the use of open source software to enhance computer science education in high schools in general and in the UAE specifically. As there is currently no early systematic review of the mentioned subject, this paper provides one.

In order to conduct this review, more than 69 research papers and articles have been collected from different resources. Over the past 10 years, many organizations have shown interest in developing ways to implement open source software in the education process. The steps in Fig. 2 show the search and selection process of the research papers used in this systematic literature survey.

The following digital libraries were searched for the required articles:

- IEEE Xplore
- ACM Digital Library
- Springer
- SIGSE
- University Researches
  - Miami University
  - University of Freiborg
  - The Higher Education Academy
  - Nova University
  - University of Oulu
- Other Sources
  - Futurelab
  - System Sciences
  - Educause
  - UNESCO
- Dwheeler
- Researchgate

https://doi.org/10.1016/j.heliyon.2019.e01235

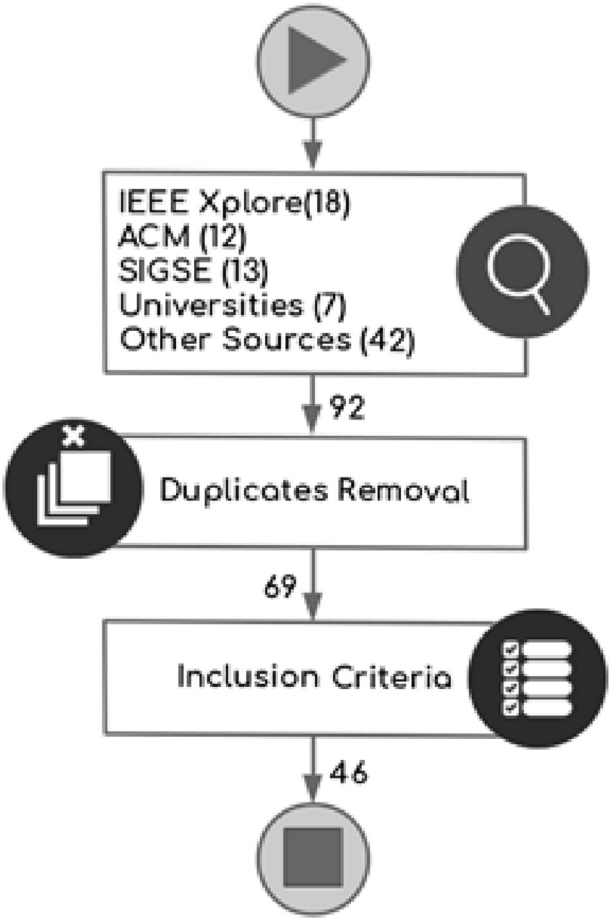


Fig. 2. Search and selection process.

Overall, there is a satisfactory amount of interest in discussing many successful cases of adopting open source software in academia; however, OSS implementation has occurred exclusively in universities rather than high schools. Moreover, these papers focused on the software side of open source usage rather than the hardware side. Fig. 3 highlights general findings in the published research, listing these findings by year of publication and also by whether the paper explored the usage of Linux & FOSS in the education system. We have also analyzed the data to show the number of papers that discussed OSS adoption in universities and schools, and the number of papers that discussed the role of open source software and hardware in enhancing the education experience.

This review begins with the general question "What is the current status of OSS in education?" It addresses whether OSS is effective and motivating, and also discusses the drawbacks to OSS use in schools. Then it narrows down the question to "What are the benefits of using OSS in computer science education?" It details possible enhancements OSS could bring, whether it is affordable and whether it could increase

https://doi.org/10.1016/j.heliyon.2019.e01235

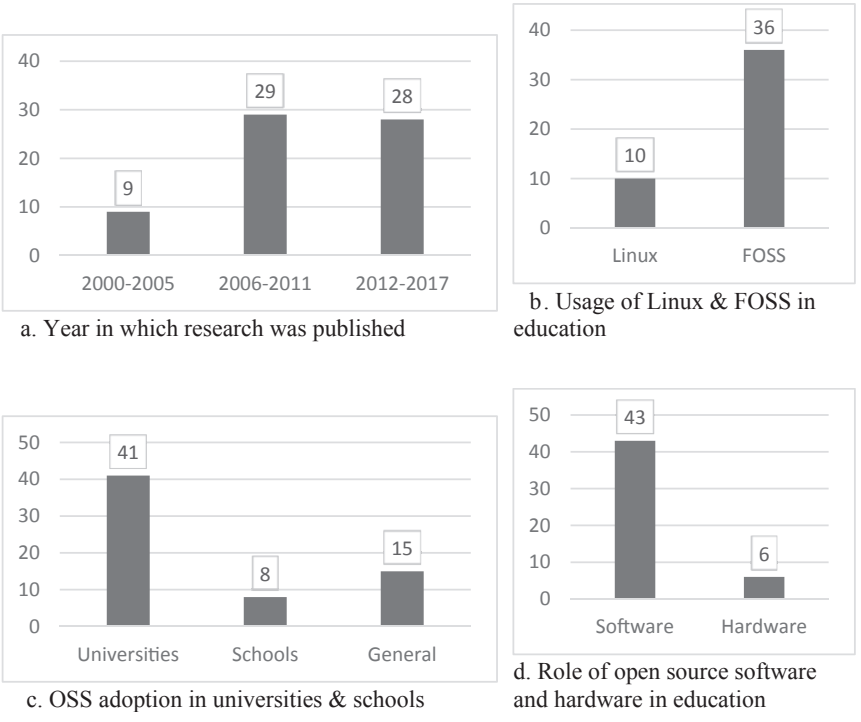


Fig. 3. Paper frequency by year, license, education level and medium.

student motivation. Finally, it ends with a question that is specific to this study: "Can OSS be used in UAE high schools for computer science education?" The response to this question is compartmentalized into challenges, results and possible educational uses in the UAE. Table 1 highlights the main results of these research questions. In the following section, a comprehensive analysis on these outcomes is discussed in greater detail.

2.2. Discussion of systematic literature review

In this section, we discuss in detail the outcomes of the systematic literature review. The current status of OSS in education is explored in terms of its effectiveness, how it affects motivational factors and barriers. The benefits of using OSS in computer science education are measured by OSS enhancements, the level of student enthusiasm as well as OSS affordability. We also introduce the challenges of using OSS in education, the expected end result of OSS usage and how it can be used. Using these findings, we aim to answer our last research question: Can OSS be used in UAE high school for computer science education?

Due to the lack of research papers that address the aforementioned issues in schools specifically, we included the current research studies that are conducted on universities, as well as on non-computer science related curriculums. Although these papers do not tackle the exact problem that we address, they have useful suggestions and

https://doi.org/10.1016/j.heliyon.2019.e01235

Table 1. Overall research questions & results.

Systematic Literature Review		
RQ1: What is the current status of OSS in education?		
Is it effective?	**Is it motivating?**	**What are the drawbacks?**
- Ten papers - open source hardware: Raspberry Pi and Arduino along with BeagleBone - Linux Operating System - Free and Open Source Software (FOSS) - Software Engineering	- Nine papers - Teamwork: collaborative programming, documentation & collaborative skills - Women's Education - Real FOSS development - Software Engineering and Operating Systems	- Four papers - Lack of awareness - Lack of knowledge - Acceptance - Compatibility - Training and expertise - Variety of alternatives - Unavailability of some functionalities - Poor documentation - Lack of support
RQ2: What are the benefits of using OSS in computer science education?		
What are the enhancements?	**Is it exciting?**	**Is it affordable?**
- Thirteen papers - Project grading system - Provide code bases - Free/low cost - Project management tools	- Two papers - Increase the challenge - Determination - Gaining popularity - Personal needs - Improve programming skills - Ideology - Collaboration - Free	- Six papers - Lack of awareness - Lack of knowledge - Acceptance - Compatibility - Training and expertise - Variety of alternatives - Unavailability of some functionalities - Poor documentation - Lack of support
RQ3: Can OSS be used in UAE high schools for computer science education?		
What are the challenges?	**What is the end result?**	**How to use it?**
- Eight papers - Lack of resources - Lack of programming skills - Lack of documentation - Lack of expertise	- Nine papers - Encouraging - Useful despite the difficulty - Student satisfaction - Smooth learning curve - Acceptance of technical difficulties	- Eight papers - Incorporating with courses - Team based collaboration - Reading and documenting - Enriching activities

insightful findings that are helpful in our case study and they facilitate the formation of the roadmap that we developed at the end of this research paper.

2.2.1. What is the current status of OSS in education?

We examined ten papers that explore the effectiveness of OSS. Three of them discussed open source hardware such as Raspberry Pi, Arduino and BeagleBone. The rest detailed open source software in general.

https://doi.org/10.1016/j.heliyon.2019.e01235

According to Jamieson (2011; Jamieson and Herdtner, 2015), using Arduino exposes students to sufficient complexity and challenges for an embedded system course. They concluded that integrating similar devices into a course's flow is of a huge benefit to both the curriculum and the students.

According to (Dias et al., 2012; Pezer et al., 2017), students could effectively use Linux to develop projects and complete assignments instead using Windows. Marmorstein (2011) and Morelli and de Lanerolle (2009) talked about Free and Open Source Software (FOSS), where students undertake projects and make them free and openly sourced for everyone on the internet. This method was used in a software engineering class, and the associated paper found that many students liked the idea, while some of them didn't.

Several findings in the available literature highlight the motivational impact of OSS on education. These are listed in Table 2. Students were enthusiastic to use Open Source Software (OSS) in classrooms and in CS education in general. It allows students to program code collaboratively with their peers, which increases their understanding of programming. Applying OSS in schools and universities can benefit students in many different fields. For example, students will come to understand how OSS developers communicate and collaborate by getting familiar with existing OSS code. Using OSS can also be a part of their assignments and projects, especially within CS courses such as Software Engineering and Operating Systems. Moreover, introducing OSS into high school curricula will eventually attract women to computing courses, because women are mainly interested in computing for the social value it provides. Women can also be attracted to the field of computer science when it involves merging computing and other specialties such as medicine, the arts and education. The OSS community and FOSS can meet students' needs by allowing them to participate in coding and sharing ideas with other students and experts online. In addition, OSS can bring students to realize that, with some CS education, they can start making a real contribution to software development projects that they care about.

Many papers discuss the drawbacks of using OSS in the classroom (Barry, 2009; Lakhan and Jhunjhunwala, 2008; Morgan and Finnegan, 2007; Shaame et al.,

Table 2. OSS motivational factors.

Motivational Factors	Published Research Papers
Teamwork: Collaborative programming that improves student documentation and communication skills, using tools such as DrJava. **Skills Improvement:** Involving students in developing major software by fixing bugs or adding new features, ensuring that they graduate with marketable skills. **Women's Education:** Women are more interested than men in applications that benefit the society. **Real FOSS development:** Letting students contribute to a real Foss project is a valuable way to make a connection between programming and what it accomplishes.	Allen et al. (2003); Bacon and Dillon (2006); Bishop et al. (2016); Buffardi (2015); Ellis et al. (2007); Hars and Ou (2002); Pavlina and Petrovic (2013); Pedroni et al. (2007); Roberts et al. (2006)

https://doi.org/10.1016/j.heliyon.2019.e01235

2013). One of the main barriers found was that people are not used to open source software and are therefore unfamiliar with how to use it. Some of the drawbacks have been listed in Fig. 4 and explained as following:

Several papers in our literature review mentioned that students, teachers and faculty members have a lack of awareness of Open Source Software, including its benefits and applications. This lack of awareness has led to a lack of knowledge as well, as instructors have no knowledge of or dealings with OSS or how to develop it. Moreover, switching from traditional programs into new programs created problems at first, because instructors were unfamiliar with these programs. Hence, accepting such software might take time. Another drawback is the potential for incompatibility between OSS and other software, such as Adobe Photoshop or Microsoft office, for example. Instructors had a hard time finding a software alternative. Different OSS programs doing the same thing creates a problem, because the instructors need to find a program that meets their needs, but without having features that need to be explained or programs that are missing some of the basic features. Finding an OSS program that is appropriate for a given class requires training and expertise in the OSS domain. In addition, many OSS programs have overlapping functionality, so the teacher needs to be familiar with each one in order to select one that is suitable for the educational context.

Lack of functionality in an OSS program, especially if such functionality is available in commercial software, will limit the instructor's ability to teach the course. Another main problem with Open Source Software is poor documentation. Usually, OSS developers are interested in developing the program, but they don't necessarily spend a lot of time on documentation. Support can be both an advantage and disadvantage. Even if the support is free and everyone can help, sometimes you might need support from the original developer in order to solve a specific problem that the community does not have a solution for. The developer might have stopped developing the tool, or may take a long time to respond.

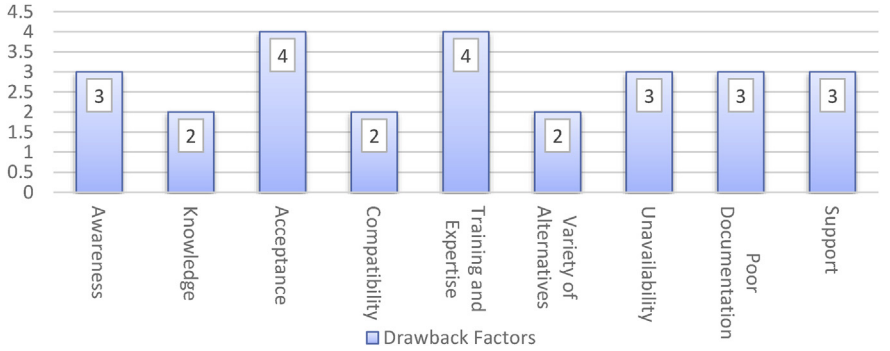


Fig. 4. Summary of OSS drawbacks & paper numbers.

https://doi.org/10.1016/j.heliyon.2019.e01235

2.2.2. What are the benefits of using OSS in computer science education?

Introducing OSS in education has demonstrated potential benefits to strategic learning. Based on a variety of data sources, the main benefits found according to the published papers in our literature review are described in Table 3.

According to (Lin and Zini, 2006; Raj and Kazemian, 2006), people are excited to study, work in and use open source programs. Some of things that drive this excitement include:

1. **The Challenge:** The challenge of using OSS—including the opportunity for people to test their abilities within the community—was one of the main reasons why people decided to implement, develop and improve the software.
2. **Determination:** Some people were determined to improve the open source program to make it more functional and more optimized.
3. **Gaining Popularity:** Students are also interested in participating in OSS projects and programs to improve their chances of getting a job and to gain skills for the e-job market.
4. **Personal Needs:** A person might need a specific function that is not implemented in the OSS, so they develop the function, add it to the software and make it available to everyone.
5. **Improve programming skills:** New learners can look up real examples and methods, improving their skills and making learning more fun.
6. **Ideology:** The belief that programs must be open source and that people can see them and learn from them.

Table 3. OSS enhancement factors.

Enhancement Factors	Published Research Papers
Project Grading System: OSS can enhance the current grading system in courses like software engineering projects based on different criteria using software such as GROW. **Provide Code Bases:** Since OSS is open to everyone, a huge code base can be found, which will increase the number of resources that students (and teachers) can access to learn or see different models of codes. **Free/Low Cost**: OSS can provide us with free or low-cost technology in classrooms, reducing the expenses used on software licenses. **Project Management Tools:** "Open-source PM tool facilitating the teaching of the usage of PM tools for project initiating and planning in alignment with the PMBOK".	Allbritton (2003); Brannock and Napier (2012); Chang (2005); Dias et al. (2012); Dorodchi and Dehbozorgi (2016); Goncalves and von Wangenheim (2017); Marmorstein (2011); Mohamed et al. (2009); Mondada et al. (2017); K. J. O'Hara and Kay (2003); Pedroni et al. (2007); Pezer et al. (2017); Saini et al. (2014),

https://doi.org/10.1016/j.heliyon.2019.e01235

7. **Collaboration:** OSS helps teachers create large projects that require a large group of students, where all students work together and collaborate to finish the project.
8. **Free:** Many OSS programs are free of charge, which allows developing countries and institutions to save money to do other projects or to develop other things.

OSS is very affordable for several reasons summarized in Table 4.

2.2.3. RQ3: Can OSS be used in UAE high schools for computer science education?

The challenges faced when trying to incorporate OSS in computer science programs are explained in Table 5. The first of these is a lack of resources and efficiency in distributing resources, since OSS has been recently introduced to the education field. In addition, one major problem is that open source software is rarely mentioned in elementary and high schools, which means that students are unfamiliar with it when they reach the university level. Moreover, not all OSS comes with well-organized and easy-to-read documentation. Lack of expertise is another challenge. An open source project started by a university can fail if there are only a few teachers and students in the community doing tasks related to their studies.

Many authors of the research papers included in our literature review encourage the use of open source software in high schools and universities (Daloukas et al., 2008; Dias et al., 2012; Dorodchi and Dehbozorgi, 2016; Hislop et al., 2009; Izbicki and

Table 4. OSS affordable factors.

Affordability Factors	Published Research Papers
Financing other Projects: Most OSS programs are free of charge, which can save institutions a lot of money. That money can then be used to support other projects or to develop the institution. **Reduce Piracy:** Users don't have to download a pirated version of the program, because the programs are free and available to all users. **Low-Cost Setup:** OSS doesn't require an expensive Hardware Setup; they can work on a middle hardware specification. **Community Support:** OSS has its own community that can help users solve problems without waiting for the main developer to solve it, which will save time. **Free Distribution:** Users have the right to modify the software to meet their needs, as well as the right to redistribute the software with their own modifications without getting into legal trouble with the developers. **Lower Education Fees:** Educational institutions can save a lot of money if they don't need to buy licenses for programs. If they use a free OSS, they can reduce the cost of studying at the institutions. **Government Economy:** Switching all governmental systems and programs to free and OSS will save governments a lot of money that can be used to boost their economy.	Barry (2009); Meera and Bandaru (2012); Pankaja and Raj (2013); Pezer et al. (2017); Smrithi Rekha et al. (2009); Terbuc (2006)

https://doi.org/10.1016/j.heliyon.2019.e01235

Table 5. OSS challenges.

Challenges	Published Research Papers
Lack of Resources: Resources include content (such as lectures or textbooks), materials (such as multimedia) and tools (such as IDEs or test environments). **Lack of Programming Skills:** Participation in OSS development is effectively limited to people with programming skills. Most students are not able to make such a contribution until after they have completed their degree. **Lack of familiarity:** The move from proprietary software to OSS can be difficult for the majority of students. **Lack of Documentation:** The OSS used should be sufficiently developed that students can find someone that can help troubleshoot their software problems, but with much OSS, this is not guaranteed. **Lack of Expertise:** It is necessary to train teachers and lecturers on the use of Linux and FOSS.	Alasbali and Benatallah (2015); Barry (2009); Hepburn and Buley (2006); Lazić et al. (2011); Saini et al. (2014); Shaame et al. (2013); Terbuc (2006); Wolf et al. (2002),

Mike, 2016; Morelli and de Lanerolle, 2009; Smith et al., 2014; Terbuc, 2006; van Gumster and Jason, 2007). Despite the warnings about how difficult open source software development can be, it was found to be very useful for education and learning. In addition to the experience gained, OSS is provided at no cost. "There is no reason why colleges and universities cannot teach production programming in the classroom". Students learning to use Linux have reported satisfaction and a relatively smooth learning curve. Students also showed remarkable patience and tended to accept the technical difficulties introduced by the open source software. OSS gives students a better appreciation for the usefulness of computing, and for how their projects can impact the world. Summary of research outcomes are illustrated in Fig. 5.

Many OSS applications are proposed in the literature (Brannock and Napier, 2012; Buffardi, 2015; Li et al., 2009; Marmorstein, 2011; K. J. . J. S. K. O'Hara, 2003; Pedroni et al., 2007; Raj and Kazemian, 2006; Smrithi Rekha et al., 2009), including the following.

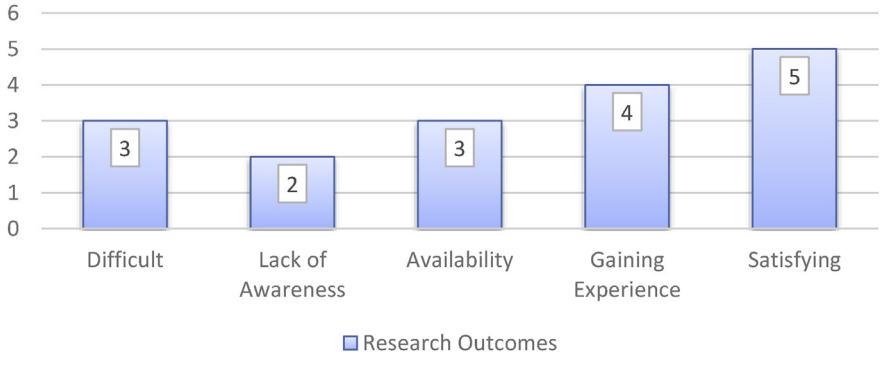


Fig. 5. Summary of research outcomes.

https://doi.org/10.1016/j.heliyon.2019.e01235

9. **Courses:** Implementing and teaching a specific program or operating system in a course will help students understand the open source concept more. Students can also get help from their teachers. The courses where OSS can be taught is illustrated in Fig. 6.
10. **Collaboration:** This works with courses, such as Software Engineering, that require projects where a large group of students work together to create a program to solve a specific problem using open source tools and libraries.
11. **Reading and Documenting:** Open source software often lacks documentation, so one idea is to give the students an open source project where they can read the code and document it to help the community to understand the program better.
12. **Activities:** Encouraging students to participate in activities that are outside the university or school, such as competitions and hackathons, can help them to better understand and work with open source programs.

2.3. Results - UAE survey & case study

We conducted many surveys and a workshop on OSS in order to understand the problem that we're dealing with and attempt to address it, and to test our solution to see whether or not it has positive results. University of Sharjah and Ministry of Education approved our surveys to be conducted. We started by asking a small group of high school teachers about their teaching experiences, in order to understand the problems, they face while teaching. Then we decided to run a workshop for them on the particular areas they were having trouble with and other topics that might help them elsewhere. To determine which areas to focus on, we conducted another survey to get the teachers' suggestions for what subjects they felt were most important to address and how we could make the workshop successful. After the workshop, we conducted two surveys: one to test teacher satisfaction and get their feedback

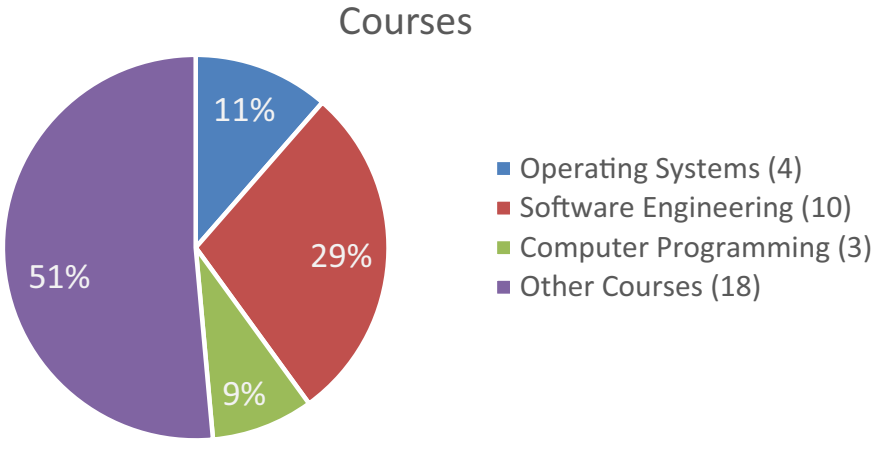


Fig. 6. Courses where OSS can be taught & number of papers.

https://doi.org/10.1016/j.heliyon.2019.e01235

about the workshop, and the other for the workshop instructors themselves, to evaluate the teachers who attended the workshop. Below, we detail the analysis of those four surveys in order. Our work is summarized in Fig. 7.

1. *Problem statement survey*

We gathered a small group of school teachers with the help of the Ministry of Education in UAE. We delivered a short presentation about open-source software, ideology and computer science in general. Then we asked them to complete a ten-question survey about what they know from their teaching experience, feedback about their teaching life, knowledge of open-source and suggestions for improving the curriculum (possibly through OSS). There were 23 teacher participants in our OSS survey, ranging from 29 to 46 years old and teaching grades 10, 11 and 12. About 20 of them had more than 7 years of experience in teaching. More than 15 of them had a great or extraordinary experience in teaching. Based on the survey answers regarding the obstacles that teachers face while teaching CS, we split them into two groups in Table 6.

Teachers' rating of student interest in school curriculum CS subjects (*on average, from best to worst):*

- Animation: students are very interested in it
- Web Design: students are moderately interested in it
- Information Security: students are somewhat interested in it
- Networking: students are neutral about it
- Java Programming: students find it boring

The following is an overview of the participating teachers' remarks:

To motivate their students, teachers often try to get them to realize how dependent people are on programs and show them inspiring apps and games made by programmers. Moreover, they sometimes mention the vast number of jobs that are open to CS

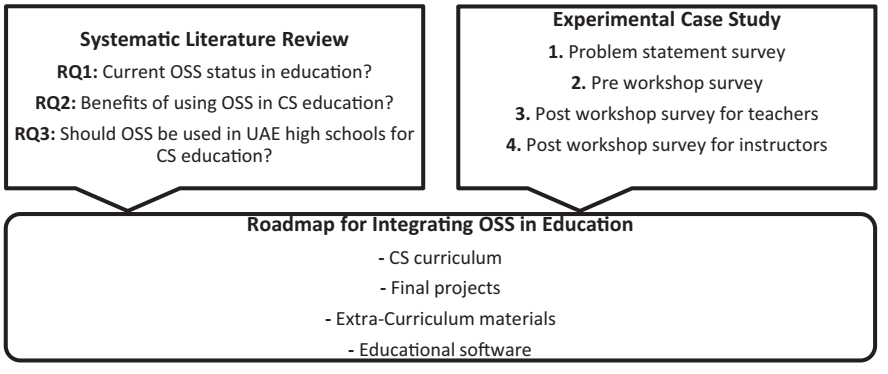


Fig. 7. Summary of our research work in this paper.

https://doi.org/10.1016/j.heliyon.2019.e01235

Table 6. Obstacles based on focused group survey.

Obstacles which can be overcome by OSS	*Obstacles which can NOT be overcome by OSS*
■ Students aren't excited or interested in CS (4) ■ Students have poor English skills (4) ■ Programming language is hard (2) ■ Boring teaching and boring content of CS (2) ■ Lack of licensed versions of software (1) ■ Subjects are sometimes taught poorly by unfit or boring teachers (1)	■ Lack of proper Internet connection and devices (6) ■ Shortage of time given for teaching CS (6) ■ The curriculum isn't appropriate for the students' level (2) ■ The frequency of changing the curriculum (2) ■ Parents and society don't give CS much importance [social stigma] (2) ■ CS courses don't contribute to student's marks (2) ■ Good students are fine with CS, but average students find it difficult (2) ■ The large number of tasks given to the teacher (1)

*The number on the right indicates how many teachers mentioned that point.

graduates and describe how cool ethical hackers are. All teachers let students write code and run it by themselves to practice programming. They also run code in front of the students while explaining it, and some even explain code on the board. Teachers give students homework of different kinds: some give group work to design a project, others make them solve programming questions and code small, fun games, a few leave the door open for students to code something related to the discussed topic.

Teachers are interested in workshops about OSS, mobile applications, networking and security, and about programming in general. They found both lectures useful, with more votes going towards the OSS lecture compared to the CS one. Teachers agreed that providing students with an open source platform for communication, collaboration and competition would increase their interest. Some would like to see an open source platform featuring collaborative coding; others would like to be able to use OSS as a portal to share documents, files and announcements. In addition, some would like to use OSS with group chatting and studying functionality. Teachers agreed that it's a good idea to let students collaborate to create an open source app/game as their final project.

When we asked about the best way to use OSS in the curriculum, teachers voted for using it as a tool in the subjects taught and using it to share educational materials remotely. Only few voted for using it as a training tool for teachers and using it in extracurricular activities. Some teachers gave additional OSS ideas to increase students' engagement and productivity:

- Activities and games over the internet for students
- Storing students' projects to be able to see them in coming years
- Creating an e-library as a reference for students

https://doi.org/10.1016/j.heliyon.2019.e01235

- Providing content for other courses or curriculums (like math, physics...)
- Gamifying the platform by awarding students who solve a problem
- Providing professional tutorial videos for different subjects
- Focusing on an open source programming language
- Getting professionals to post helpful content for students

2. *Pre-workshop survey*

We asked a small group of teachers for their suggestions before conducting the workshop to make sure that we were on the right track. There were 8 pilot participants in our OSS survey about the teachers workshop. Many thought "School name" means where they studied, so we'll add "(in which you're a teacher)". A few did not know what "Teaching Grade" means, so we'll change it to "Grades you're teaching". They also showed interest in the following workshop topics: Python & Arduino, 2D/3D tools, IoT and Linux.

Most participants went for the advanced level subjects more than the basic ones. The largest number that wanted to learn about IoT chose smart home functions, followed by smart garbage systems and smart plant watering. In 2D/3D tools, most teachers wanted to learn about Blender; few wanted to learn about the other listed options. Most teachers wanted to learn how to teach a complete Arduino project with a fan and a temperature sensor. As for Linux, all listed topics had a generally equal level of interest, with commands and scripting being at the top. The suggested topics were as follows:

- Building web applications in Python
- Game & Functional programming in Python
- IoT monitoring of information
- IoT backbones: Thread, Weave, Brillo
- Setup and manage Linux web server
- Extra: A workshop about GitHub

To make the workshop successful, some suggested taking the emphasis away from theory and focusing on the practical part. A suggestion was also made to provide printed manuals/code labs, and not to have to wait for others to finish.

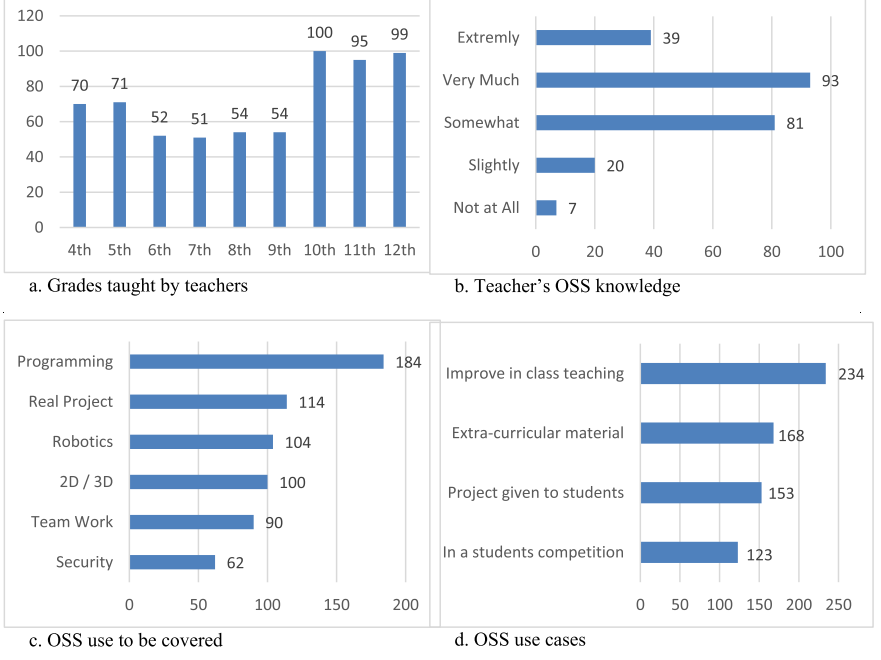
3. *Post-workshop teachers survey*

Based on the teachers' feedback from the above-mentioned surveys, we planned the workshop content. We designed it to have ten simultaneous sessions each with a different topic ranging from subjects that are currently being taught in the curriculum (e.g. Python, Android) to subjects that might be helpful for the teachers' self-development or in their teaching experience (e.g. Arduino, Linux, IoT Home

https://doi.org/10.1016/j.heliyon.2019.e01235

Automation). These ten sessions were run twice in the same day, allowing each teacher to attend any two sessions of their choice. After finishing each session, we asked the teachers to fill out a survey to rate the teacher, the assistants, the session overall, and whether they found it beneficial or not. The survey analysis and final results are summarized in Fig. 8.

A total of around 350 teachers filled out the surveys: 240 teachers filled out the first survey, while 165 teachers filled out the second survey. The teachers came from schools throughout around the UAE. Most teachers had more than five years of teaching experience, and more than half had been teaching for more than ten years. Most of the teachers taught high school students, while the rest taught grades four to nine. Regarding the teachers' knowledge of the subject, most of the teachers self-reported that they had good knowledge about Open Source Software. Only a small percentage of the teachers self-reported slight knowledge about it, or didn't know anything about OSS. As for the applications of open source to improve students' skills, most teachers responded that they supported using it to improve programming skills. A few teachers suggested using it in Data Analysis, Health and Creativity. More than 50 teachers answered with "No" or left the question empty. When questioned about how they wanted to use the skills learned in the workshops sessions, the highest votes were for using them to improve their class teaching. 14 teachers answered with "None" or left the question empty. Some teachers mentioned other uses, including personal development or creating IT clubs in schools and using their new skills for these clubs.



a. Grades taught by teachers

b. Teacher's OSS knowledge

c. OSS use to be covered

d. OSS use cases

Fig. 8. Summary of teacher feedback on post workshop survey.

https://doi.org/10.1016/j.heliyon.2019.e01235

As for the topics the teachers wished to learn about in the workshops, the most frequently requested topics were: Robotics, 3D Printing, Networking, Security, Design, AR & Ethical Hacking. Other requested topics included: Games & their Engines, Animation, Video Editing, Other Programming Languages, AI, Edu Apps, Electronics, Advanced Python & Android, Motivational & Expo Topics. Many requested topics that were already being covered (which might indicate either poor publicity or a misunderstanding): Python, Arduino, 2D/3D. As for teacher interest in attending future workshops, most teachers said they would like to attend future workshops like this one. Two thirds of the teachers thought that this session would enhance computing studies and add value to the curriculum (Fig. 9). Only a few thought otherwise, and the rest weren't sure whether it would or would not.

In the overall session analysis as shown in Fig. 10, the second sessions had better ratings than the first ones. The first sessions were mostly rated as average or above average. The second sessions were mostly rated as outstanding or above average. Overall, the number of unsatisfied teachers seems to be very low. Most ratings were between average and outstanding.

4. *Post-workshop instructor survey*

After the workshop, we asked the workshop instructors and teaching assistants to complete a survey to evaluate the teachers' participation and give us feedback about the workshop. Thirty participants filled out the survey. Half of them were instructors

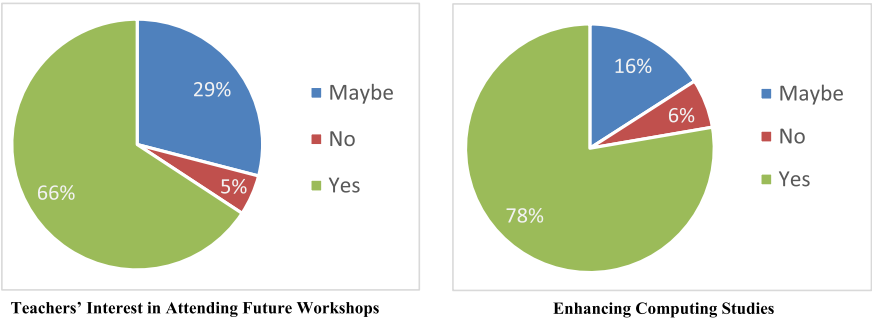


Fig. 9. Overall teacher feedback.

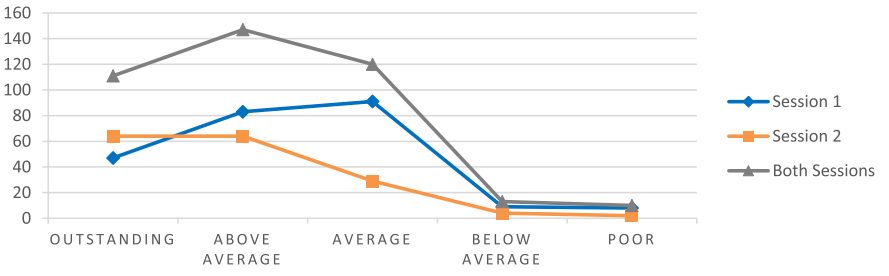


Fig. 10. Overall OSS session rating – Teachers' perspective.

https://doi.org/10.1016/j.heliyon.2019.e01235

and the other half were teaching assistants. One third of them were well aware of Open Source Software, while the other two thirds had good knowledge about OSS. Three to five people contributed to the instruction of each workshop. Most voters said they would be willing to contribute to workshops like this one in the future. Most voters thought that this session would enhance computing studies and add value to the curriculum, while some of them weren't sure whether it would or not. The highest number of votes regarding the use of the skills taught in the workshop was for using the materials to improve in-class teaching and as extra-curricular material. Overall rating as shown in Fig. 11:

- Most voters rated school teachers as below average, average or above average (half of them chose average).
- All voters rated their overall experience as average, above average or outstanding (half of them choose above average).
- The session and the manuals got a better rating than the rating for the workshop overall.

To that end, the instructors proposed the following suggestions:

- Add a web design and development session.
- Add a session to teach how to contribute to Open Source projects and use GitHub.
- Include videos for illustration in the manuals.
- Reduce the use of paper during the sessions.
- Organize more workshops; become more active in developing projects together.

Our summary of this experimental case study is presented in Table 7.

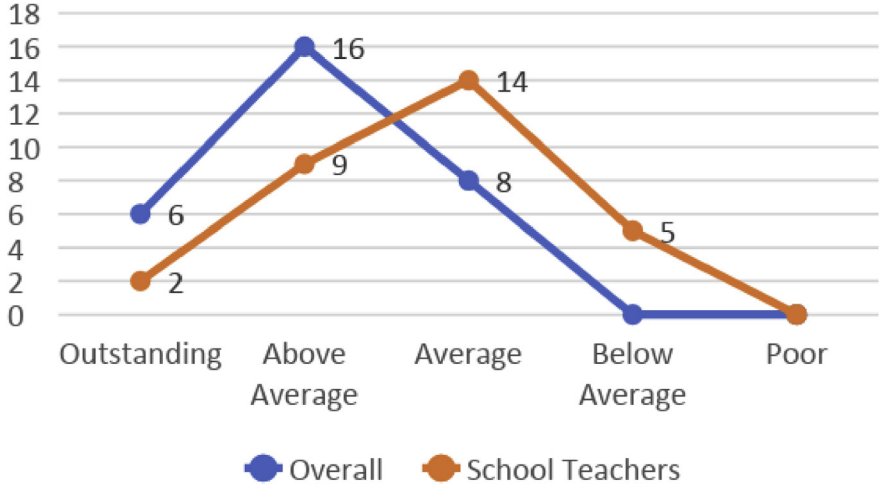


Fig. 11. Overall OSS session rating—Instructor perspective.

https://doi.org/10.1016/j.heliyon.2019.e01235

Table 7. Experimental case study summary.

Experimental Case Study			
Research Outcome	**Benefits**	**Challenges**	**Future Work & Implications**
Problem statement survey	Obstacles which can be overcome by OSS - Students aren't excited or interested in CS - Students have poor English skills - Programming language is hard - Boring teaching and boring content of CS - Lack of licensed versions of software - Subjects are sometimes taught poorly by unfit or boring teachers	- Lack of proper Internet connection and devices - Shortage of time given for teaching CS - The frequency of changing the curriculum - Parents and society don't give CS much importance - CS courses don't contribute to student's marks - Good students are fine with CS, but average students who find it difficult - The large number of tasks given to the teacher	- Activities and games over the internet for students - Storing students' projects to be able to see them in coming years - Creating an e-library as a reference for students - Providing content for other courses or curriculums (like math, physics...) - Gamifying the platform by awarding students who solve a problem - Providing more professional tutorial videos for different subjects - Focusing on an open source programming language - Getting professionals to post helpful content for students - Other requested topics include: Games & their Engines, Animation, Video Editing, Other Programming Languages, AI, Edu Apps, Electronics, Advanced Python & Android, Motivational & Expo Topics. - Adding a Web design and development session. - Adding a session to teach how to contribute to Open Source projects and use GitHub. - Including videos for illustration in the manuals. - Reducing the usage of papers during the sessions. - Organizing more workshops, and become more active and developing projects together. - Applying the roadmap that is proposed in this research paper.
Pre workshop survey	The suggested topics were as follows: - Building web applications in Python - Game & Functional programming in Python - IoT monitoring of information - IoT backbones: Thread, Weave, Brillo - Setup and manage Linux web server - A workshop about GitHub	The challenge to coordinate with the Ministry of Education and organize ten parallel sessions, each with a different topic, ranging from subjects that are currently being taught in the curriculum (e.g. Python, Android) to subjects that might be helpful for the teachers' self-development or even in their teaching experience (e.g. Arduino, Linux, IoT Home Automation).	
Post workshop teachers' survey	- Two thirds of the teachers thought that this session would enhance computing studies and add value to the curriculum. - Most teachers said they would like to attend future workshops like this one. - The most frequently requested topics were: Robotics, 3D Printing, Networking, Security, Design, AR & Ethical Hacking.	- Challenges were faced in the first sessions. The second sessions had better ratings than the first ones. The first sessions were mostly rated as average or above average. The second sessions were mostly rated as outstanding or above average.	
Post workshop instructors' survey	- All voters rated their overall experience as average, above average or outstanding (half of them chose above average). - The session and the manuals got a better rating than the rating for the workshop overall.	- Most voters rated school teachers as below average, average or above average (half of them chose average).	

https://doi.org/10.1016/j.heliyon.2019.e01235

3. Discussion

3.1. Roadmap

So far, we have discussed several suggested uses of OSS in computer science education, including using OSS as tools for enhancing the educational environment and reducing software expenses, as well as using OSS as educational material to be taught and added to the curriculum itself. Both use cases were explored in the systematic literature review section, which found that OSS is highly effective in improving the educational infrastructure. We are therefore including both potential uses in our roadmap as illustrated in Fig. 12.

Based on the information we gathered, we came up with a draft of a roadmap that might be helpful in improving the current educational system. We investigated all solutions proposed in the literature, and then analyzed the survey results and selected the solutions that were applicable and suitable for our case study. We then arranged the selected solutions into three main targets: (1) the curriculum itself throughout the semester, (2) a final project at the end of the semester, and (3) extra-curricular

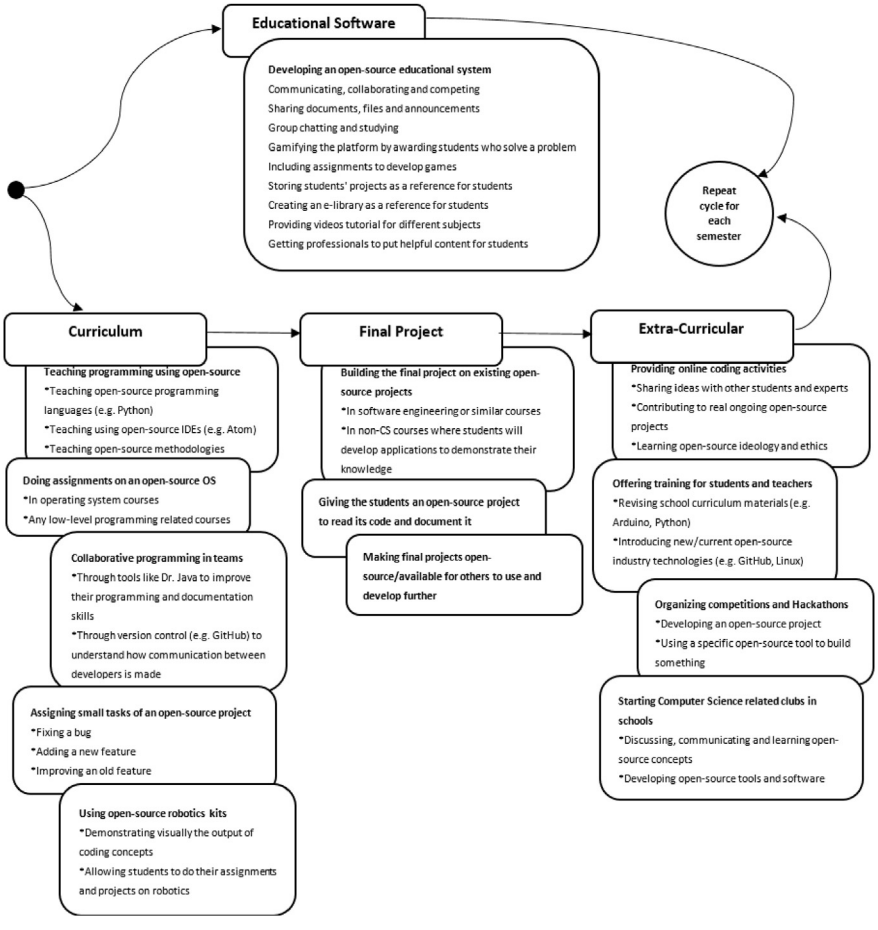


Fig. 12. Roadmap of using OSS in education system.

https://doi.org/10.1016/j.heliyon.2019.e01235

material that could be used during vacations or holidays. Furthermore, this roadmap proposes educational system software that could be developed to facilitate and accommodate the former three targets. These three targets, combined with the educational system (illustrated in Fig. 12), form a cycle that could be repeated every academic school semester.

This roadmap could be generally implemented in the educational system as a whole, or it could be used selectively according to each school's capabilities, needs or priorities.

The bottom line of the roadmap demonstrates the three targets and their corresponding list of steps for improving the educational experience in UAE schools specifically, and in high schools in general. They are as follows:

3.2. Curriculum (during the semester)

This section proposes possible steps to improve the current academic curriculum. Each school's administration would be responsible for providing the necessary tools to carry out these enhancements, and the teachers would initiate these steps throughout the semester.

1. Teaching programming using open-source

There are several ways this could be done, either by using open-source programming languages like Go or PHP. A superior option would be Python, as it has wider uses for both students and teachers. Another method would be to use open-source editors that would facilitate teaching and learning programming languages like VS Code, Atom or even Notepad++. Aside from open-source tools and languages, Open-source methodologies could be used to introduce students to the idea of open-source software programming and its development lifecycle.

2. Conducting assignments using open-source operating systems

In some courses that relate to the operating system architecture, it's advisable to do the assignments on an open-source operating system, as every single component of the OS is openly sourced and thus accessible for exploration by the students. This may open the students' minds and provide them with practical experience, compared to the purely theoretical learning methods used on proprietary operating systems.

3. Collaborative programming in teams

Programming in a team is an essential skill in the computing industry. It's very important to introduce students to the concept and to have them use it throughout their semester on tasks and assignments. Teachers can give their students this opportunity using tools like DrJava or GitHub (Version Control) by asking them to do

https://doi.org/10.1016/j.heliyon.2019.e01235

small coding or documentation tasks. Many companies facilitate this sort of team-based programming in their software, including most IDEs and especially Unity 3D Engine for making games.

4. Assigning students small tasks of an open-source project

An enormous number of open source projects out there could be used in the classroom. The teacher could pick one public project, share its link with their students, and ask them to do small tasks as course work throughout their semester. Such tasks could be to add a minor feature, make an improvement or fix a small bug that exists in the project, or better yet a bug that the teacher made himself to test the student's understanding of the lecture.

5. Using open-source robotics kits

With the school administration's support, teachers could buy a set of low-cost, open-source, educational robotics kits like Scribbler Robot or Sparki, and then use these kits to visually demonstrate the lecture's theoretical concepts and have students do various assignments on these kits. If the subject is more advanced, then it's better to use open-source circuit boards like Arduino or Raspberry Pi to demonstrate such subjects.

6. Providing open-source examples of key concepts

Students always ask: "Why are they teaching us this?" Teachers should be able to provide an answer! They could answer that question by demonstrating to students an open-source project that implements the concepts taught in class, thereby showing students the relationship between academics and industry. The teacher could assign a snippet of code from that OSS project for students to read and write about, in order to expand and test their understanding of the subject.

7. Using open-source tools instead of commercial ones

As a means of reducing the costs spent on purchasing licenses for commercial software, schools could replace proprietary software with OSS. This could be done with the software in the current curriculum (i.e.: replacing Fusion 360 with Blender 3D), or with software used for common tasks (i.e.: replacing Microsoft Office with LibreOffice) or even for the whole operating system (i.e.: replacing Microsoft Windows with a user-friendly distribution of Linux). This would free up funds that could be spend improving other sectors of the school.

3.3. Final project (end of semester)

This section proposes steps that could be taken at the end of the academic semester to ensure a better outcome of the courses taught. This section requires no extra cost and minimal effort from the teachers.

https://doi.org/10.1016/j.heliyon.2019.e01235

1. A teacher could ask their students to build a final project on existing open-source projects, allowing students to sharpen their collaborative programming skills, understand each other's code (an important skill in the computing industry), and expand the knowledge they gained throughout the course. Such an approach is proven to be effective in courses like Software Engineering and Operating Systems. Now it can be tested on other Computer Science courses, as open-source projects have been developed to perform a wide variety of tasks nowadays. It can also be used in non-Computer Science courses, such as Physics or Mathematics, to develop better applications in the related field.
2. Another suggestion of a final project is to give the students an open-source project where they would read its code, understand it and document it accordingly. This would be an easier project than in the first suggestion, as it doesn't require the students to develop or write any code. Instead, it is a test of their ability to understand code, demonstrate their documentation abilities and prepare for real-life projects.
3. Once they have finished the project, students should make their code open-source and thus available to the public, allowing others to contribute and develop it. Interesting projects could gain visibility, becoming an excellent addition to the students' resumes. These projects can also become resources for students in subsequent years to use to develop their own projects.

3.4. Extra-curricular (after the semester or during holidays)

This section of the roadmap addresses the steps that could be taken outside of the academic curriculum: after the semester, during the school holidays, or during school days but outside the classroom. Some points in this section require a lot of effort from the Ministry of Education, as they would work best on a large scale and therefore may incur considerable expenses.

1. Online coding activities during the holidays

As there are a large number of online resources about various open-source tools, languages and projects, teachers could share these resources with their students, encouraging them to read about OSS during the holidays or whenever they have time. Teachers could also motivate their students to work on projects of their choice or to contribute to real ongoing open-source projects, and to share ideas and knowledge among themselves and with experts about project development.

2. Offer training workshops for students and teachers

Training workshops could be offered by the Ministry of Education on a large scale, or by groups of enthusiasts on a smaller scale. These workshops could cater to the teachers or the students and aim to revise school curriculum materials like

https://doi.org/10.1016/j.heliyon.2019.e01235

Arduino or Python, or to introduce new topics about open-source industry technologies like GitHub or Linux.

3. Organize competitions and Hackathons

If a large-scale Hackathon is organized, this step may require the collaboration of an organization, or if a more localized competition is organized, the school administration may have to get involved. Such competitions and Hackathons could involve developing an open-source project or using a specific open-source tool to build something. These kinds of events are always a great experience, as they allow students to explore new ideas, communicate and socialize, learn new things, work as part of a team and have fun.

4. Start Computer Science clubs in schools

These clubs could open the doors for students to discuss, communicate and learn about open-source concepts, and to develop open-source tools and software under the supervision of a teacher. In addition, students could help each other with curriculum-related material, study together and organize fun events, making the club into a welcome break from the classroom.

3.5. Developing an open-source educational system

The top line of the roadmap diagram describes the proposed educational software that could be used in school systems to facilitate the three steps detailed above (from the bottom line of Fig. 12). This software could be used throughout the semester, during the holidays, as students work on their final projects and between semesters. It should be developed and maintained by expert developers assigned by the Ministry of Education, as the idea is for all schools in the UAE to use this system. This software would be open-source, and would have the following features:

1. Communicating, collaborating and competing

As with any educational system, it would allow students to communicate and send messages and emails to one another. It would also have multi-user functionalities, allowing students to collaborate in teams on coding platforms such as GitHub or DrJava. Also, it should facilitate the organization of competitions, from creating polls to submitting and presenting final code.

2. Sharing documents, files and announcements

This is also a very basic feature. Teachers and authorized parties would be able to use this software to share documents, files and announcements with the student. It could also be used by club organizers to share online resources among club members.

https://doi.org/10.1016/j.heliyon.2019.e01235

3. Group chatting and studying

Students should be able to communicate in groups through group chats and virtual classrooms, allowing them to study together or discuss a team project.

4. Gamifying the platform by awarding students who solve a problem

This is an important feature, especially for high school students, as teens like games and rewards. If the software rewards them with badges and level-ups for task completion and assignment submission, students would be more motivated to use it. If this aspect is implemented successfully, students would not see using the platform as a mandatory task, but rather as an exciting game that they find enjoyable.

5. Include game-like coding assignments

As mentioned above, students like games. Why not make their tasks more like games, rather than boring academic assignments? This software would allow teachers to set up enjoyable and puzzling game-like assignments, like showing a timer of a bomb that would explode as an indication of a time limit, or requiring students to build a bridge with lines of code so that vehicles could pass over.

6. Storing students' projects to retrieve them in the future

This software should also provide a public archive of school projects accumulated throughout the years so that students from later years can use previous code for reference and learning, allowing them to build on previous work instead of starting from scratch.

7. Creating an e-library for student reference

The software would include a library of e-books included in the curriculum, as well as external books suggested by teachers or ordered by students. It would also include all course materials and references uploaded to the software over the years.

8. Providing video tutorials for different subjects

The software would have a video management and editing section so teachers and experts could upload videos explaining different subjects, as well as a video viewer for the students to watch those videos in a structured playlist.

9. Getting professionals to contribute content

The software would facilitate contribution by outside professionals to school resources. This could include video management, an interactive drag-and-drop tutorial maker and a game maker, offering students a wide array of options to choose from.

https://doi.org/10.1016/j.heliyon.2019.e01235

4. Conclusion

This research paper has conducted a systematic literature review on the use of OSS to enhance computer science studies in high schools in general and in the UAE specifically. More than 69 different research papers and articles have been collected from different resources in the past 10 years. There is a satisfactory amount of interest in discussing many successful cases of adopting open source in academia; however, OSS implementation has taken place exclusively in universities rather than high schools. Moreover, we launched an experimental case study in which 400 high school teachers were invited to take multiple OSS workshops in order to enhance their teaching experience using OSS. We conducted two pre-workshop surveys as well as two post-workshop surveys. According to the survey results, two-thirds of the teachers thought that these sessions would enhance computing studies at their school and add value to the curriculum. After examining more than 69 different research papers and taking the survey findings into account, we constructed a roadmap that can be used by any educational institute—especially high schools—to strategically integrate OSS into the educational system. The roadmap is designed to target three main aspects of implementation: using OSS in the curriculum itself throughout the semester, using OSS in final projects at the end of the semester, and using OSS in extra-curricular activities. The future direction of this research is to implement this roadmap and measure its effectiveness. The roadmap can be applied via three phases: changes to curriculum, implementation in final course project and establishment of extra-curricular activities.

Declarations

Author contribution statement

Manar Abu Talib: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Omar Einea: Performed the experiments; Analyzed and interpreted the data; Wrote the paper.

Qassim Nasir: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data.

Mohamad Fouzi Mowakeh, Mohamed Eltawil: Performed the experiments.

Funding statement

This work was supported by the Office of Research & Graduate Studies at the University of Sharjah, as well as by the OpenUAE Research and Development Group. It

https://doi.org/10.1016/j.heliyon.2019.e01235

is supported by the Ministry of Education in the UAE. Dr. Manar Abu Talib and Dr. Qassim Nasir have received research grants from University of Sharjah for this research project to be conducted with the Ministry of Education. Project #: 140447 (1602141139-P).

Competing interest statement

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

Alasbali, N., Benatallah, B., 2015. Open source as an innovative approach in computer science education A systematic review of advantages and challenges. In: 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE), pp. 278−283.

Allbritton, D.W., 2003. Using open-source solutions to teach computing skills for student research. Behav. Res. Methods Instrum. Comput. 35 (2), 251−254.

Allen, E., Cartwright, R., Reis, C., 2003. Production programming in the classroom. ACM SIGCSE Bulletin. 35 (1), 89.

Bacon, S., Dillon, T., 2006. The potential of open source approaches for education. Futurelab 44. Retrieved from. http://www.futurelab.org.uk/resources/documents/opening_education/Open_Source_report.pdf.

Barry, B.I.A., 2009. Using open source software in education in developing countries: the Sudan as an example. In: 2009 International Conference on Computational Intelligence and Software Engineering, pp. 1−4.

Bishop, J., Jensen, C., Scacchi, W., Smith, A., 2016. How to use open source software in education. In: SIGCSE 2016 - Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 321−322.

Booth, A., Sutton, A., Papaioannou, D., 2016. Systematic Approaches to a Successful Literature Review. Sage.

Brannock, E., Napier, N., 2012. Real-world testing: using FOSS for software development courses. In: Proceedings of the 13th Annual Conference on Information Technology Education - SIGITE '12. ACM Press, New York, New York, USA, p. 87.

https://doi.org/10.1016/j.heliyon.2019.e01235

Buffardi, K., 2015. Localized open source collaboration in software engineering education. In: Proceedings - Frontiers in Education Conference, 2014. FIE.

Chang, L., 2005. Enriching software engineering courses with service-learning projects and the open-source approach. In: Proceedings - International Conference on Software Engineering, 2005, pp. 613–614.

Daloukas, V., Dai, V., Alikanioti, E., Sirmakessis, S., 2008. The design of open source educational games for secondary schools. In: Proceedings of the 1st ACM International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '08. ACM Press, New York, New York, USA, p. 1.

Dias, J., Tavares, S., Carvas, A., Silva, P.S., 2012. Open source operating system for students: EOS project. In: Proceedings of the Workshop on Open Source and Design of Communication, pp. 79–83.

Dorodchi, M., Dehbozorgi, N., 2016. Utilizing open source software in teaching practice-based software engineering courses. In: Proceedings - Frontiers in Education Conference, 2016–Novem. FIE.

Ellis, H.J.C., Morelli, R.A., de Lanerolle, T.R., Damon, J., Raye, J., 2007. Can humanitarian open-source software development draw new students to CS? ACM SIGCSE Bull. 39 (1), 551.

Goncalves, R.Q., von Wangenheim, C.G., 2017. DotProject+: open-source software for project management education. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE, pp. 213–215.

Hars, A., Ou, S., 2002. Working for free? Motivations for participating in open-source projects. Int. J. Electron. Commer. 6 (3), 25–39.

Hepburn, G., Buley, J., 2006. Getting open source software into schools: strategies and challenges. Innovate J. Online Educ. 3 (1). Retrieved from. http://nsuworks.nova.edu/innovate.

Hislop, G.W., Ellis, H.J.C., Morelli, R.A., 2009. Evaluating student experiences in developing software for humanity. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education - ITiCSE '09, 41. ACM Press, New York, New York, USA, p. 263.

Izbicki, M., Mike, 2016. Open sourcing the classroom. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16. ACM Press, New York, New York, USA, 723–723.

Jamieson, P., 2011. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? Proc. FECS 289–294. Retrieved from. http://www.worldcomp-proceedings.com/proc/p2011/FEC3377.pdf.

https://doi.org/10.1016/j.heliyon.2019.e01235

Jamieson, P., Herdtner, J., 2015. More missing the Boat - Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them. In: Proceedings - Frontiers in Education Conference, 2014. FIE.

Jesson, J., Matheson, L., Lacey, F.M., 2011. Doing Your Literature Review: Traditional and Systematic Techniques. Sage, London.

Kitchenham, B., 2004. Procedures for Performing Systematic Reviews. Joint Technical Report between Keele University and NICTA. ISSN:1353-7776.

Lakhan, S.E., Jhunjhunwala, K., 2008. Open source software in education. Educ. Q. Mag. 31 (2), 32–40. Retrieved from. http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterlyMagazineVolum/OpenSourceSoftwareinEducation/162873.

Lazić , N., Banek Zorica , M., Klindžić, J., 2011. Open source software in education. In: MIPRO, 2011 Proceedings of the 34th International Convention. IEEE, Opatija, Croatia. Retrieved from. https://ieeexplore.ieee.org/document/5967252/.

Li, W., Zhang, S., Li, Z., 2009. Open source movement and computer science education innovation. In: 2009 International Conference on Information Engineering and Computer Science. IEEE, pp. 1–4.

Lin, Y., Zini, E., 2006. An empirical study on implementing free/libre open source software (FLOSS) in schools. In: Social Informatics: an Information Society for All? in Remembrance of Rob Kling. Springer, Boston, MA, pp. 123–132.

Marmorstein, R., 2011. Open source contribution as an effective software engineering class project. In: ITiCSE '11 Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, pp. 268–272. Retrieved from. https://dl.acm.org/citation.cfm?id=1999823.

Meera, L., Bandaru, D., 2012. A feasible rural education system. Int. J. Adv. Comput. Sci. Appl. 3 (1).

Mohamed, N., Seman, M.S.A., Hussein, R., 2009. Open source software in information technology education. In: 2009 International Conference on Information Management and Engineering. IEEE, pp. 99–102.

Mondada, F., Bonani, M., Riedo, F., Briod, M., Pereyre, L., Retornaz, P., Magnenat, S., 2017. Bringing robotics to formal education: the thymio open-source hardware Robot. IEEE Robot. Autom. Mag. 24 (1), 77–85.

Morelli, R., de Lanerolle, T., 2009. Foss 101: engaging introductory students in the open source movement. In: SIGCSE '09: Proceedings of the 40th ACM Technical Symposium on Computer Science Education.

 https://doi.org/10.1016/j.heliyon.2019.e01235

Morgan, L., Finnegan, P., 2007. Benefits and drawbacks of open source software: an exploratory study of secondary software firms. IFIP Int. Fed. Inf Process 234, 307–312.

O'Hara, K.J.J.S.K., 2003. Investigating open source software and educational robotics. J. Comput. Sci. Coll. 18 (February 2003), 8–16. Retrieved from. https://dl.acm.org/citation.cfm?id=771717.

O'Hara, K.J., Kay, J.S., 2003. Open source software and computer science education. J. Comp. Sci. Coll. 18 (3), 1–7. Retrieved from. http://dl.acm.org/citation.cfm?id=771712.771716.

Pankaja, N., Raj, M., 2013. Proprietary software versus open source software for education. Am. J. Eng. Res. 02, 2320–2847. Retrieved from. www.ajer.org.

Pavlina, K., Petrovic, B., 2013. Students' attitudes to the use of Open Source technology in education. In: 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2013 - Proceedings, pp. 597–599.

Pedroni, M., Bay, T., Oriol, M., Pedroni, A., 2007. Open source projects in programming courses. In: Proceedinds of the 38th SIGCSE Technical Symposium on Computer Science Education SIGCSE, 07, p. 454.

Pezer, M., Lazić, N., Odak, M., 2017. Free and open source software in the secondary education in Bosnia and Herzegovina. In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 882–886.

Raj, R., Kazemian, F., 2006. Using open source software in computer science courses. In: Proceedings. Frontiers in Education. 36th Annual Conference. IEEE, pp. 21–26.

Roberts, J.A., Hann, I.-H., Slaughter, S.A., 2006. Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the Apache projects. Manag. Sci. 52 (7), 984–999.

Saini, S.K., S, S.A., D, A., C N, K., 2014. A web-based degree program in open source education. In: Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion, pp. 1083–1086.

Shaame, Abdalla, Shanmugam, Kamalanathan, Dehghantanha, A., 2013. An educational framework for free and open source software. Int. J. Innovat. 4. Retrieved from. http://ijimt.org/papers/348-D0433.pdf.

Smith, T.M., McCartney, R., Gokhale, S.S., Kaczmarczyk, L.C., 2014. Selecting open source software projects to teach software engineering. In: Proceedings of

 https://doi.org/10.1016/j.heliyon.2019.e01235

the 45th ACM Technical Symposium on Computer Science Education - SIGCSE '14. ACM Press, New York, New York, USA, pp. 397–402.

Smrithi Rekha, V., Adinarayanan, V., Maherchandani, A., Aswani, S., 2009. Bridging the computer science skill gap with free and open source software. In: 2009 International Conference on Engineering Education (ICEED). IEEE, pp. 77–82.

Talib, M.A., Elnagar, A., 2015. A new computer science student recruitment strategy university of sharjah (UOS) case study. J. Comput. Sci. 11 (1), 145–152.

Terbuc, M., 2006. Use of Free/Open Source Software in e-education. In: 2006 12th International Power Electronics and Motion Control Conference. IEEE, pp. 1737–1742.

van Gumster, J., Jason, 2007. Red hat high. In: ACM SIGGRAPH 2007 Educators Program on - SIGGRAPH '07. ACM Press, New York, New York, USA, p. 34.

Wolf, M.J., Bowyer, K., Gotterbarn, D., Miller, K., 2002. Open source software. In: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education - SIGCSE '02, 34. ACM Press, New York, New York, USA, p. 317.

 https://doi.org/10.1016/j.heliyon.2019.e01235