Customization of open-source solutions on the example of the LMS Moodle distance learning platform

P A Ukhov, D A Borshchenko, D D Kabanov, M E Bergen and A V Ryapukhin

Moscow Aviation Institute (National Research University), Volokolamskoe highway 4, 125993, Moscow, Russia

E-mail: ryapukhin.a.v@mail.ru

Abstract. Open-source software has been a vast topic of thought and discussion since its inception. Many companies and developers did not understand for what purpose open-source software was developed, and some even opposed the existence of such solutions. Others, on the contrary, supported and still support the creation and distribution of this kind of programs, despite their unprofitableness. One of these programs is Moodle. It is a free web application that provides the ability to create websites for online learning, created in the early 2000s. Anyone can easily create or modify their own learning platform for free and without problems with copyright and licenses. The platform supports a huge number of languages, is constantly updated, and is also easy and convenient to customize. The user can add and customize his plugins and addons. The system has found wide application in connection with the emergence of COVID-19, since most schools and universities have switched to distance learning. Moodle functionality allows you to collect more than 50 users in one session without losing productivity. The platform has a built-in mechanism for authorizing users, including a login and password, which will not allow intruders to carry out unauthorized access to personal data of users. Teachers can leave the necessary files and recordings of classes on the platform for learners to access them. This article discusses the goals and issues of creating open-source software, its customization and application for solving problems of various kinds of activities, including the organization of multi-user educational platforms using the Moodle platform.

1. Introduction

As you know, a personal computer is a universal device for information processing. Personal computers can perform any information processing action. To do this, it is necessary to compose an exact and detailed sequence of instructions for a computer in a language it understands - software on how to process information [1]. Ada Lovelace, daughter of the famous poet George Byron, is considered to be the very first creator of software. The first programmable personal computer is considered to be the z3 Computer, the software was written using machine codes and numerical instructions understandable by the processor for which the program is written. The writing of software as we know it today dates back to the 1950s. It is not known for certain at what point or even the year the first open-source software appeared, however, some consider the first exchange of programs as such. The most famous bands whose members shared "source" with each other were PACT, SHARE and DECUS. Software is one of the most important and high-speed components of modern information technologies, world information technologies, automated systems, world information resources. Computer software can be divided into operating system and application software. An open-source program can belong to both groups. The...
operating system ensures the functioning and interconnection of all components of the computer and provides the user with access to its hardware capabilities. Application software, in turn, is divided into two groups: development tools and applications. Development tools are commonly understood as programmer tools. Traditional development tools are programming environments that use algorithmic programming languages. Applications are software products or packages of application programs, focused mainly on a non-programming user and implementing a certain group of functions or information technologies - work with documents, multimedia materials, communications and other functionality demanded by an ordinary user. The overwhelming part of the software produced by Open-Source can be attributed to the group of applications [2]. Most of the applications, as a rule, are written according to the principle of maximum convenience for the user and have a friendly interface, however, there are exceptions when the user is required to have certain knowledge related to operating the system or any syntax of the programming language, which makes the use of this software practically impossible for the average user. The reader is invited to familiarize himself with this problem a little later in this article.

The appearance of a huge amount of software has provoked the emergence of interest among ordinary users and specialists in the principle of the operation of a particular program. The demand for such information has spawned Open-Source software, in other words, open-source software.

An open-source product is software that is available to everyone, including the availability of the source code together with the ownership rights that allow you to use, study, modify or distribute the software product. Open-Source programs have a number of advantages over their commercial counterparts. One of the important advantages is the openness of the software: the user himself can view the source code of the program, for example, in order to make sure that no third-party modules are embedded in the code that can violate data confidentiality or interfere with the work of other programs installed by the user [3, 4].

Other advantages include the possibility of modifying the product, its ability to fix errors that interfere with the stable operation of the program, free products in most cases, as well as its versatility, because a certified Open-Source product is developed on the basis of international standards, which allows you to freely use the program when working with different browsers, operating systems and devices. It is also worth highlighting the presence of feedback from users who can send bug reports for further correction by a team of technical specialists. Despite its advantages, this software also has disadvantages. In some cases, the software does not have technical support, as a result of which the user has to deal with the problem on his own. It should also be noted that there are some malicious programs that can impersonate an Open-Source product.

Despite the presence of such significant shortcomings, Open-Source products have become widespread on the web. Most of these software products are integrated into applications that are used by a huge number of people every day: browsers, social networks and other resources.

2. Problem statement

Commercial companies set aside finance, labor, and time to create a product that can be distributed free of charge. There are several reasons for this. One of them is the widespread dissemination of their program. Free products, for obvious reasons, are far better distributed than commercial ones. This can serve as an advertisement for the company and its other products. Another reason may be partial monetization of the product. A standard copy of the program can be supplied free of charge, while additional functions and utilities are available for a fee. The third reason is the previously mentioned “feedback” from users, because they can also affect the development process and fix product bugs.

In his book, Carl Vogel gives an example from Jim Blandy, a free software hacker: «In 1993 I was working for the Free Software Foundation, we were beta testing GNU Emacs 19. We made a beta release about once a week, and users tried to work on it and sent bug reports. There was one guy among them that none of us personally met, but he did a great job: his reports were always clear and pointed to the problem, and when he sent in the fix himself, it almost always worked. He was perfect» [6].
This example only illustrates that “feedback” is important both for the user, so that the problem is fixed, and for the developer himself, in order to understand the essence of the problem and prevent its possible reoccurrence. Independent developers create open-source products in order to attract the attention of investors and large companies, as with the help of their code they show what they are capable of and what they can do. This is the reason why Open-Source software still exists today.

It is also worth noting again that free software solutions, which are the majority of Open-Source products, are distributed much more willingly, which means that more people, including specialists, will be able to customize this software for themselves [6].

Customization is considered to be the change and personalization of a product for the needs of a certain circle of users. By changing the programs for the computer, you can turn it into the workplace of an accountant or constructor, designer or scientist, writer or agronomist [7]. Customization can occur both at the code level and at the level of the graphical user interface (GUI), which is usually used for ease of use of the product by the user. Some users change the GUI, adapting the programs for the visually impaired, or simply for users who could not understand the GUI of the original program. More often than not, it is the GUI change that gets the attention of the developers, since it is not as time-consuming work as changes at the code level. Changes at the code level imply the introduction, modification and removal of functional modules from the program code that ensure the presence and performance of a particular function. Some users believe that the program may lack functionality that is necessary in connection with their work. Therefore, users consider it necessary to build these capabilities into the standard functionality of the program, which allows users to achieve the most effective results from the product. Quite often, users do not build in any new modules, but only change the existing ones, for example, optimizing them for the most efficient operation of the program in terms of performance and consumed resources.

In order for a program to be recognized as an Open-Source product, it must be developed in accordance with the requirements of the Open-Source Initiative. In total, it is customary to distinguish ten requirements:

- Free distribution;
- Availability of source code and the possibility of changing it;
- Permission to modify the program distributed under the same license as the program itself;
- Integrity of the author's code, which prohibits its distribution, but only on condition that the modified author's code can be distributed in the form of patches or builds (new versions of the product);
- No discrimination against individuals or groups;
- Non-discrimination of spheres of activity;
- Distribution of the license for the program;
- Generality of the license, implying the absence of specialization of the license for a specific software product;
- The license must not prohibit the use of other products;
- The license must be technology neutral, which means that the license must not specialize in a particular interface or software solution.

Failure to comply with the above requirements means the product does not have an Open-Source Initiative license

3. Research questions
In this paper, we will consider an Open-Source product using Moodle as an example. The Moodle Course Management System is a platform built specifically to provide a personalized, secure learning experience. The convenience of this platform lies in the ability to create courses, fill them with educational material and integrate functional modules such as chat, database and others. In addition to
the above, Moodle also allows the student to get acquainted with the educational material at any time convenient for him from any place where there is Internet access [8]. This Open-Source product is used in many educational portals, websites and other resources that imply an educational process. Moodle is a web-based framework that requires a PHP-enabled web server and a database server.

In this work, we will integrate Moodle with third-party modules, self-developed plugins (including integration ones) and themes using gitlab.mai.ru, which is a modern solution for collective software development, which works by analogy with GitHub - the largest platform for collaborative software development. These platforms use the following development techniques: “Continuous Integration” (CI) / “Continuous Delivery” (CD) and “Development and Operations” (DevOps). CI/CD involves development, deployment, and command, speeding up the build, test, and deployment process of an application. The term itself consists of two other CI (Continuous Integration) and CD (Continuous Delivery). Continuous Integration is about constantly merging working copies into a common main development branch and performing frequent automated builds of a project to quickly identify potential defects and solve integration problems. This increases the efficiency of the work performed and allows you to notice and fix errors before the complete program build to minimize the time spent on fixing them. Continuous Delivery means the production of software in short iterations, ensuring that the software is stable and can be delivered to production at any time. CI/CD in its combination can greatly minimize the time required to fix problems when they arise, since the final product is released with a certain frequency in small portions. It is quite obvious that fixing a small part of the program is much easier than going through the whole product [9]. DevOps involves the active interaction of the development team with the IT service specialists and the mutual integration of their workflows with each other to ensure a high-quality product. In simple terms, DevOps means the interaction of developers working on different parts of a software product. In simple terms, DevOps means the interaction of developers working on different parts of a software product. Developers in different fields can discuss ways to stitch together parts of a program into a final product, which greatly increases the overall quality of the final product. The combination of CI/CD and DevOps is one of the most effective, since in parallel with the release of a new build, the developers also have a general view of the final product and have an idea of the methods of its compilation, namely the combination of separately developed parts. Gitlab uses the same syntax as GitHub, so new users familiar with GitHub will feel comfortable in this development environment. The reader is offered the code of the file “.gitlab-ci.yml”:

```yaml
image: docker:stable

stages:
  - build
  - plugins
  - test
  - deploy

before_script:
  - apk add git
  - apk add lftp

download_moodle:
  stage: build
  script:
    - mkdir moodle
    - git clone git://git.moodle.org/moodle.git moodle
  artifacts:
    name: "moodle_with_plugins"
    paths:
      - moodle
```
expire_in: hour

install_plugins:
  stage: plugins
  script:
    - cd moodle/mod
    - git clone https://github.com/moodleulm/moodle-mod_bigbluebuttonbn bbb
  dependencies:
    - download_moodle

test:
  stage: test
  script:
    - echo 'Test complete'

deploy:
  stage: deploy
  script:
    - lftp -c "set ftp:ssl-allow no; open -u $FTP_USERNAME,$FTP_PASSWORD $FTP_HOST; mirror -Rv moodle/ . --delete --ignore-time --parallel=10 --exclude=.git* --exclude=.git/
    - echo 'Deploy complete'
  dependencies:
    - download_moodle
    - install_plugins
    - test
  timeout: 6h

This code is successfully going through the development process, otherwise called the pipeline. Thanks to the pipeline, you can connect programs and pull data through them. It is customary to distinguish the following model of the pipeline: build-test-send-deployment [10]. Let's consider the components of this code. The code has 4 stages: build, plugins, test, deploy. First, a docker image is added to the configuration file, with which all the tasks will be implemented. The block diagram of the code is shown in the figure 1.

The "build" stage is responsible for building the application. At this stage, the latest version of LMS MOODLE is cloned from the master branch on the github platform. Also at this stage, a "moodle" folder is created for newly received files. In case of successful completion of this stage of building the project, all existing directories and files must be saved for transfer to the next stages of Continuous Integration (CI).

The plugins stage creates a mod folder inside the moodle folder and downloads the BigBlueButton software, which is used by students and teachers of the Moscow Aviation Institute to conduct online classes. It is this module that will be the third-party module that we install within the framework of this project.

At the "test" stage, the code is tested and the test results are displayed on the screen. The main advantage of automated code testing is that it runs many times faster than manual testing - this makes it possible to run tests after each code change. In addition, automated testing in Gitlab allows some protection of the written code, since there is a built-in version history. Tests help locate and point to where the error occurred.
Figure 1. Block diagram of the code.
It becomes more convenient to make edits to the code, since there is no need to be afraid that during the editing any functionality of the application was broken. Testing is especially useful when developing complex applications in a large team.

During the “deploy” stage, the code is deployed to the test server.

The code deployment process is so well optimized that it takes an average of 10 minutes to complete. This allows you to perform this operation several dozen times a day. You can use non-engineering staff to deploy your code. Under the guidance of engineers, they can make small changes to the code. In addition to regular software deployment, testing and deployment processes can be applied for any purpose, for example, to build virtual machines or to test a monitoring program [11].

In the course of this work, a test server was created using Amazon Web Services. On this server, Gitlab Runner for Linux operating system was installed, which was also configured in Gitlab. In addition, variables have been added to the CI/CD Gitlab settings to apply to the project environment.

The code was deployed on a rented server from Amazon, a company that was one of the first to use the endless possibilities of the Internet, including providing users with servers on which they can deploy their programs and applications [12]. On this platform, the code works quite stably, which is why it was chosen by the authors.

4. Purpose of the study
As a result of this work, a process for continuous integration, continuous delivery and continuous deployment (CI/CD) of open-source software was developed using Gitlab. This process allows teams of developers to create customized software at a fairly high speed. The Moodle course management system was chosen as an OpenSource product, in which third-party modules were implemented, placed in third-party repositories. Autotesting of the received customized product was carried out on a test server. After making improvements to the code and testing it, the code was uploaded to the product server.

5. Research methods
As you know, it is impossible to write a program that meets the needs of all users at the same time. Customized code must be constantly supplemented with various modules. Therefore, the main problem is the ability to connect third-party modules or plugins. Each of these modules can be optional and developed by a separate team. It is important that for a specific project you can connect optional modules from other repositories and keep them up to date.

The OpenSource product is more secure because the community quickly finds holes in the software and reports them to the developers, who quickly fix them. The quality of open-source code is often much higher than that of closed code, since such projects are not worked on by a separate group of several people, but by tens and hundreds of developers from all over the world. Bugs in OpenSource programs are found very quickly, as the code is reviewed and tested by many developers and testers.

6. Findings
The main problem with custom software is the disregard for the user interface. An intuitive interface is achieved if the user starts using the application and almost immediately understands which buttons to press in order to get to a particular section. Each user has his own experience of working with programs and interfaces, and if one person finds the interface intuitive, this does not mean that other users will find it the same. It is necessary to take into account the purpose for which a customized product is created, what is its target audience.

7. Conclusion
In this article, the reader was invited to familiarize himself with the concept of open-source software, the goals and reasons, as well as the problems of its creation, while covering the history of its creation and development.
This article also showed software development as a continuous integration, continuous delivery, and continuous deployment (CI/CD) process of open-source software using Gitlab. Moodle, an open-source software designed to improve the learning process, was chosen as the Open-Source product. Third-party modules were introduced into the software, extending the functionality of the original product, placed in third-party repositories - file locations.

Autotesting of the received customized product was carried out. Based on the test results, a product server was deployed and this code was uploaded to this server.

On the grounds described in the article, we can safely say that open-source software not only has the right to exist, but even is needed in the modern market. The reason for this is the ability of users to customize this software according to their needs and desires, as well as the ability to view the source code of the program in order to make sure it is safe. However, a complete transition to free software is not only impossible, but also undesirable. Such a transition could severely hit the economies of most countries, bankrupting corporations, slowing down scientific and technological progress, and also creating the threat of widespread malware, since despite the increased security of free software, if a vulnerability or malicious code is found in it, many users will find themselves threatened by. In addition, free software tends to be rapidly and ubiquitous, as noted earlier in one of the previous paragraphs.

References
[1] Romanova Yu A 2005 Informatics (Moscow: Eksmo)
[2] Golitsyna O L, Partyka T L and Popov I I 2019 Software (Moscow: INFRA-M)
[3] Jammoul S M, Syuzev V V and Andreev A M 2019 Open Source Software Usage in Education and Research: Network Traffic Analysis as an Example IGI-Glob. 28 331-2
[4] Kaimin V A 2009 Computer science. Textbook for students (Moscow: INFRA-M)
[5] Fogel K 2005 Producing open source software: How to run a successful free software project (Sebastopol: O'Reilly Media, Inc.)
[6] Sirotkina I 2009 Product customization Un. entr. 5 2-3
[7] Mozharov R V, Mozharova N R, Evteev V V, Kuzmenko O A and Shevchenko M O 1999 Personal computer software. Textbook for universities (Moscow: Finstandartinform)
[8] Tikhomirova O G 2018 Study of the essence and process of customization of production in a digital environment of functioning Fund. Res. 9 93-7
[9] Humble D and Farley D 2019 Continuous software deployment: automation of build, testing and implementation of new software versions (Moscow: Litres)
[10] Kravchenko G V 2013 Using the Moodle distance environment in the educational process of full-time students Bull. of the Alt. State Un. 2(78) 23-5
[11] Davis J and Daniels R 2016 Effective DevOps: building a culture of collaboration, affinity, and tooling at scale (Sebastopol: O'Reilly Media, Inc.)
[12] Stone B 2013 The everything store: Jeff Bezos and the age of Amazon (New York: Random House)