Article

Approach for Evaluating the Suitability of Programming Environments for Learning and Developing Purposes

Aleksandr Igumenov*, Jevgenijus Marinuškinas
Department of IT Technologies, Vilnius Business College,
Kalvarijų 129-401 Vilnius LT-08126, Lithuania
* Corresponding author, e-mail: *aleksandr.igumenov@kolegija.lt*

Received: 13 June 2023
Accepted: 28 June 2023
Online: 30 June 2023

Abstract. Currently, the ability to create various programs has become particularly relevant. The example of this trend is the programs created for smartphones. In addition, the smartphone is becoming one of the tools for work and is increasingly used in many activities, such as work activities, entertainment, education, and IoT.
As a result, there is an increasing need for modern, versatile, and efficient tools for creating such types of programs. Also, development of such programs requires special programming skills. According to these facts we can see that such situation evolves a big request for specialists in this field of activity.
One possible solution to this problem is the use of alternative development tools that do not require deep programming knowledge or skills, such as visual programming tools. These tools have the potential to create a new generation of programmers who do not need to be experts in low-level programming. And these tools need to be properly selected based on the requirements of the program.
This article aims to provide a systematic approach to evaluating the possibilities of any programming environment software for learning and development purposes.

JEL: C61.

Keywords: visual programming; block programming; traditional programming; imperative paradigm; logical paradigm; functional paradigm; object-oriented paradigm; Thunkable; Xamarin; comparative analysis approach.
Citation: Aleksandr Igumenov, Jevgenijus Marinuškinas (2023) Approach for Evaluating the Suitability of Programming Environments for Learning and Developing Purposes. – *Applied Business: Issues & Solutions* 1(2023)35–39 – ISSN 2783-6967.
https://doi.org/10.57005/ab.2023.1.5

Introduction

The process of software development is the complex one that involves various tasks which must be completed to create a high-quality and useful software product [1]. This process comprises several steps, each of which has specific objectives and tasks. During the analysis step, relevant issues are examined, and requirements are defined that are essential for the planned product development. During the design step, the system architecture is created, technical and programming requirements are formulated, and components and their interaction are described. In the coding step, programming tasks are carried out, and during the testing step, testing of program software is performed. The installation step involves tasks related to the installation process of the developed software, such as setting up a personal computer or server features, integrating operating system capabilities into the project, and creating new libraries for additional installation. Finally, the support step is carried out with technical assistance and consulting services providing.

This article focuses on the program codding step, which is one of the most critical steps during software development. In this step, it is important to choose an appropriate programming language and tools. In modern programming, there are two main categories of program code development tools: text-based programming tools, where the programmer writes language-specific program text in a programming environment's text editor, and visual programming tools which allow programs to be created using drag-and-drop interfaces.

This article examines tools for block programming, which allow the creation of programs using pre-made visual blocks and their combinations within a programming environment. It is important to note that block programming is a specific type of visual programming that has its own specifics and should not be confused with other forms of visual programming. As Alexander Repenning notes [2], using the term block programming as a synonym for visual programming is not equivalent, which is why the specific term of block programming is used in this work to avoid ambiguity and specifically refer to the type of programming language being used.

This work is devoted to:
1) propose an approach for comparative analysis of program development solutions,
2) propose a specific program architecture and design for verification of any of program development solutions.

1. Literature review

When speaking about the *traditional programming*, it should be noted that such activity requires specific technical knowledge and skills. One of the main skills worth mentioning is understanding the principles and syntax of programming environments and programming languages (C++, Java, C#, etc.). To create a program using a *textual programming* language, in most cases, a programming environment and a set of supporting tools are required. The programming environment is often made from many different elements, and its main workspace is a type of specific text editor. Program text is written in it, which is later compiled or interpreted depending on the programming language and used environment. Modern programming environments provide users with a wide range of possibilities for writing program text and correcting errors. Some of them have functionality for visual elements designed for the screen of the program.

Traditional programming can be classified into several widely used programming paradigms. One possible classification was made by Kurt Nørmark [3], which includes: the *imperative paradigm*, *logical paradigm*, *functional paradigm*, and *object-oriented paradigm*. Currently, the most widely used paradigms are the object-oriented and imperative paradigms (specifically, a variation of the imperative paradigm called *procedural programming*). Object-oriented programming uses the concepts of classes and objects to create models that are based on the real world [4].

The essential difference of the procedural programming paradigm is that, unlike object-oriented programming, it uses a sequence of instructions. Procedural programming is dependent on procedures executed sequentially [4]. As a result, it is not always suitable for some modern more complex programs.

Regarding both paradigms mentioned, it should be noted that the dominant paradigm currently is object-oriented. It can be said that the object-oriented paradigm provides the programmer with more flexibility for software creation. The program code created based on this paradigm can be reused without additional changes (reusage of public or standard libraries) - where it may be necessary to be used for a programming task, and it is also significantly safer because it gives control for data access outside of the object [4].

One possible solution to this problem is the use of alternative development tools that do not require deep programming knowledge or skills, such as visual programming tools. These tools have the potential to create a new generation of programmers who do not need to be experts in low-level programming [5].

Block programming is also influenced by many different factors. One of these factors was mentioned earlier, the need of knowledge of the programming languages for developing the applications using any traditional programming languages. Block programming tools solve this problem by providing a more understandable and simpler programming environment interface. Such types of platforms require minimal programming knowledge. When working with this block programming tool, the user operates with visual blocks that can be easily composed into one not always complicated program. The programming tool has special mechanisms, such as animating cursors or block coloring, which indicate the compatibility of blocks, according to Alexander Repenning [2], and at the same time simplify the process of creating a program.

Block programming provides opportunities to apply both previously mentioned programming paradigms - object-oriented and procedural or certain combinations thereof, depending on the specific tool and programming environment used. Most often, objects and actions performed with them are manipulated using the drag-and-drop principle of the user interface. When programming in a Block programming environment, the ability to manipulate different program blocks or prepared objects by combining them into complex combinations is provided. A typical example of such type of a programming environment could be *Thunkable* [6] - a modern programming environment designed to create programs for mobile devices with Android OS.

It is also important not to forget another important feature of such type of programming. Usually, programming tools based on the block principle are classified as SaaS (Software as a service) products. They operate on the principle of cloud computing, and unlike traditional programming tools, do not require the user to perform an installation process on their computer, operating through a web browser instead. As a result, the preparation process is most simplified and is only a request of creating a user account. Moreover, it should be emphasized that traditional programming tools may require considerable computing power and technical resources, while tools created based on cloud computing place lower demands on user hardware and can theoretically be accessed from any type of computer (personal computer, nettop, ultratop, tablet, mobile etc.) with a browser and access to internet. Additionally, as shown by the statistics in Fig. 1 [7], professional tools are losing popularity, and other tools (including block programming) are taking an increasing share of the market, from 0% in 2019 to 10% in 2021.

3. Approach Procedure

To determine the effectiveness of any types of programming solutions, it was necessary to describe guidelines for a comparative analysis approach that would be applied during the implementation of the practical part of the analysis. It should be noted that we propose this approach which applies to mobile program developed using both a traditional programming and a block programming software's. To analyse the suitability of a specific solution, the following characteristics were compared (all values were estimated using a

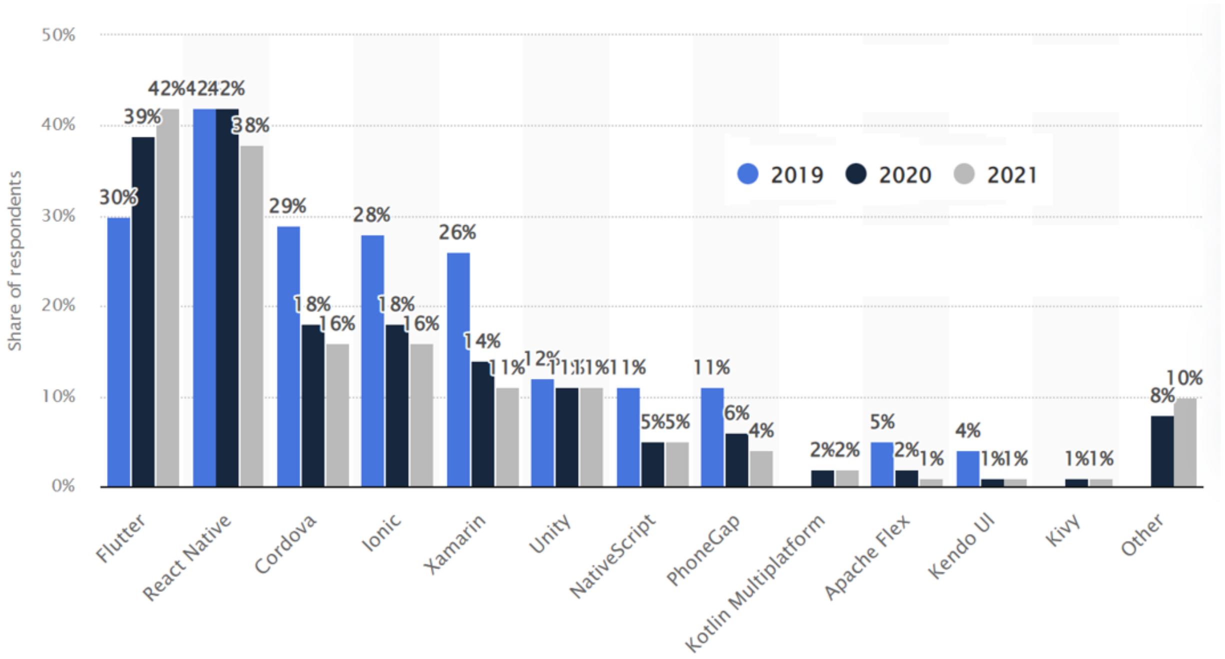

Fig. 1. The change in popularity of cross-platform mobile app development tools between 2019 and 2021. Adapted according to Ref. [7].

timer).

1. The time needed to develop methods (functions, procedures).
2. Time spent creating loops (e.g., while, for).
3. Time spent describing variables.
4. Time taken to write control statements.
5. Time taken to write text management/processing statements.
6. Time taken to write logical statements.
7. Time taken to add/edit various visual elements.

In addition to the usual creation of logical functional elements, the programming of visual user interface elements is also evaluated in terms of development time: i) Screen, ii) Button, iii) Label, iv) TextBox, v) DropDownList.

To ensure objectivity in the evaluation results, we propose to assess each visual interface or programmable element using three different time measurement factors.

1. Minimum time: The minimum time achieved, measured after creating the mobile program.
2. Maximum time: The maximum time achieved, measured at the initial stages of learning to use the solution.
3. Typical time: The average time it usually takes to act on the workflow. This result is obtained by adding and averaging the three randomly measured times.

The implementation of these functions and the development of the mobile program code are guided by methodologies aimed at writing efficient and fast code [8]. Furthermore, it is intended to validate the applicability of best practices and methodologies described by professional software engineer V. Podkamennyi [9].

1. The DRY methodology (*Don't Repeat Yourself*), which is all about avoiding repeated code development, i.e., using as many components as possible that are versatile and reusable.
2. The KISS methodology (*Keep It Simple, Stupid*), which is based on simplicity and code readability, avoids the creation of large and complex code constructs.

Since the development solutions being evaluated are different, it has been decided to assess the applicability of the methodologies (DRY and KISS) based on their levels of utilization.

1. Full: The development solution enables the comprehensive application of the principles and best practices of the respective methodology.
2. Partial: The development solution allows for the partial application of the principles and best practices, but there may be certain limitations or constraints that hinder full implementation.
3. None: The development solution does not provide the necessary features or capabilities to support the application of the principles and best practices associated with the methodology.

The whole program creation process was divided into several steps for further investigation.

1. Preparation (program development solution installation or account creation).
2. Designing the user interface.
3. Programming the elements or blocks of the mobile program.
4. Installing and testing of the mobile program.

To evaluate the effectiveness of this approach for evaluating of development solutions, a mobile program called *SmartHouseBudget* will be developed based on the client-server architecture. This mobile program aims to provide a platform for recording housing costs, catering to users who own one or more properties. However, the approach does not include the server programming part because it is only an API tool.

The mobile program will store client data, including property details and expenses, on the server. This allows the installation of the program on multiple devices, ensuring real-time synchronization of data. Customer account information, such as email address and password, will also be stored on the server, enabling individual customer logins.

The program will enable users to manage property-related data, including payments, as well as add and delete information about different apartments. It will also provide visual representations, such as graphs, to facilitate the analysis of the entered data.

By implementing this mobile program, it will be possible to test and compare the performance of the any program development solutions in developing software of medium complexity.

4. Program Requirements and Database Model

The development of the *SmartHouseBudget* mobile program for the approach requires the formulation of requirements for the program. Clearly defined and exhaustive requirements are essential for the program of the waterfall approach, contribute to the quality of the final result, and facilitate the future development of the program's functionality. The two main types of requirements that are the focus of this paper are functional and non-functional requirements.

4.1 Functional requirements

The program will work with three main types of objects: User, Property, and Cost. Each of these objects has the following functional requirements. Requirements for the User object are presented as follows.

1. The user registration process requires the user to enter their first name, last name, password, email address and phone number.
2. All fields have a maximum length of 30 characters and are mandatory. If any of these rules are violated, an error message will be displayed when the registration button is pressed.
3. The login process requires the user to enter their email address and password. An error message will be displayed if there is an error or if the data is not entered.
4. Upon successful login, the user will be redirected to the Property List window.

Requirements for the Property object are presented as follows.

1. Adding a Property requires the entry of an address (up to 100 characters), which is mandatory, and an optional name (up to 200 characters). An error message will be displayed when the rules are not followed.
2. Deleting a Property object can be done by a long click, which will display a confirmation message.
3. Deleting a Property object also deletes all associated spend records and refreshes the list.
4. Clicking on a Property object in the list will display the associated Cost records.

Requirements for the Cost object are presented as follows.

1. Adding a new Cost object requires entering its Name (up to 100 characters), Comments (optional field, up to 200 characters), Amount, selecting the Date and Time from the drop-down list, and selecting the Category and Subcategory from the drop-down lists. An error message is displayed if any of the rules are violated.
2. A Cost object can be deleted by a long click, which displays a confirmation window.

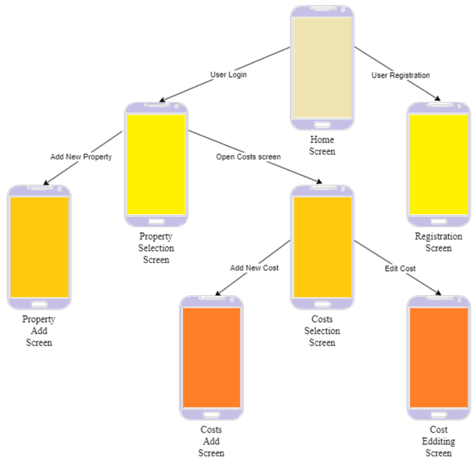

Fig. 2. Screens - logic diagram of the program.

3. The list is updated when an object is deleted.
4. Updating an item allows you to change the title, comments, amount, date and time. The same rules apply as in the Add Item window.

General functional requirements are presented as follows.

1. The 'Back' system button allows to return to the previous screen.
2. The Real Property objects screen displays a graph with the total amounts calculated for each month since the beginning of the current year.
3. The Costs records screen displays a graph controlled by two drop down lists - Period and Category. By default, the period selected is 'Month' and no category ('-'). The graph will then show the expenses for the current month by category. If the period is changed (another option is the current year), the graph will be updated. If a category is selected, the graph will be displayed by sub-category. The drop down lists also affect the list of expense records.
4. The Costs record screen also includes a section called Costs Summary, which displays the total amount for the selected period.

4.2 Non-functional requirements

In addition to the aforementioned functional requirements, the program is subject to the following non-functional requirements.

1. Responsive design (adapted to the screen size of the mobile device).
2. Modern and visually appealing user interface.
3. Intuitive program control.
4. High contrast colors.
5. Compatibility of the program with both new and old devices (minimum Android OS version 8.0).
6. Graphical user interface in Lithuanian language.
7. Data access via the Internet.
8. Data storage on the cloud or common web server.
9. Secure and encrypted data transmission with the server.

4.3. The structure of the program

Initially, we need to determine the functionality of the program, and since it is designed for a mobile phone, we should divide the functionality into relevant screens as presented in Fig. 2.

4.4. Database structure

Fig. 3 represents the inner structure of database designed for supporting the solution and storing the data. Direct relations were organized between tables *Users* and *Estate Objects*, *Expences* and *Estate Objects*, *Expences* and *SubCategories*, *Categories* and *SubCategories*. According to the best database traditions, each record in the table starts from unique item *Id*.

5. Main results

The article proposes an approach for evaluating the effectiveness of programming development solutions, focusing specifically on mobile programs created by any of presented programming software.

1. The approach involves a comparative analysis of various characteristics such as development time for methods, loops, variables, control statements, text management/processing statements, logical statements and visual elements such as screens, buttons, labels, textboxes and dropdown lists.

2. To ensure objectivity, each visual interface or programmable element is evaluated using three different time measurement factors: minimum time, maximum time and typical time.

3. Function implementation and code development follow approach aimed at writing efficient and fast code, specifically the DRY and KISS methodologies. These methodologies emphasise reusability, simplicity and code readability.

4. The applicability of the DRY and KISS methodologies is assessed based on their level of use: full, partial or none, depending on the ability of the development solution to support the principles and best practices associated with each methodology.

5. The program development process is divided into preparation, design of the user interface, programming of the elements or blocks, and installation and testing of the mobile program.

6. A mobile program called *SmartHouseBudget*, based on a client-server architecture, is being developed to evaluate the effectiveness of the approach. This program focuses on the recording of housing costs and allows the installation of several devices with real-time data synchronisation. It allows users to manage property-related data, make payments and provide visual representations for data analysis.

By implementing the *SmartHouseBudget* program and applying the proposed approach, the performance and suitability of different program development solutions can be compared and evaluated in the context of medium complexity software development.

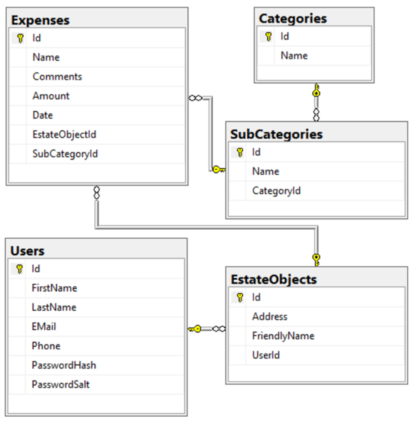

Fig. 3. Database structure.

Conclusions

This article presents an approach for evaluating and comparing mobile program development solutions. The approach includes analysis of development time, adherence to best practices, and practical implementation. By assessing coding task time and considering the application of DRY and KISS methodologies, developers can gain insight into the efficiency and industry compliance. The approach usability proposed to apply through the development of a mobile program called *SmartHouseBudget*, which provides a real-world usage context for evaluating different solutions.

Abbreviations

DRY	-	Don't Repeat Yourself
IoT	-	Internet of Things
KISS	-	Keep It Simple, Stupid
SaaS	-	Software as a Service
STEAM	-	Science, Technology, Engineering, Arts, Math. The STEAM Open Access Centre

Acknowledgements

Authors would thank to Vilnius University STEAM Center. Part of this article was prepared using the material of Final thesis which was successfully defended by Jevgenijus Marinuškinas at Vilnius Business College in June 2021.

Authors' contributions

Aleksandr Igumenov and Jevgenijus Marinuškinas initiated research concept and design, Jevgenijus Marinuškinas collected and analysed data. Both authors interpreted data and prepared the manuscript. Jevgenijus Marinuškinas prepared the theoretical overview, approach and requirements, whilst both authors prepared other parts of the manuscript. Aleksandr Igumenov prepared and Jevgenijus Marinuškinas reviewed the initial draft of the manuscript. Both authors reviewed and approved the final manuscript.

Conflicts of interest

All authors declared at they have no conflicts of interest.

References

1. Shylesh, S. (2017) A study of software development life cycle process models. - *SSRN Electronic Journal* - http://dx.doi.org/10.2139/ssrn.2988291.
2. Repenning, A. (2017) Moving beyond syntax: Lessons from 20 years of blocks programing in agentsheets - *Journal of Visual Languages and Sentient Systems* 3(1)(2017) 68-91 - https://doi.org/10.18293/vlss2017-010.
3. Nørmark, K. (2013) Functional Programming in Scheme With Web Programming Examples - Functional Programming in Scheme. - Department of Computer Science, Aalborg University, Denmark, 2013. - https://homes.cs.aau.dk/ normark/prog3-03/html/notes/top-level-title-page.html, retrieved May 1, 2023.
4. Adhkari, B. (2016) Object Oriented Programming Vs Procedural Programming. - https://doi.org/10.13140/RG.2.2.33443.45604.
5. Bernard, A. (2020) Is low-code/no-code the future of application development? - *TechRepublic. TechnologyAdvice* - https://www.techrepublic.com/article/is-low-codeno-code-the-future-of-application-development/, retrieved May 1, 2023.
6. Friedman, M. et al (2010) Best no code app builder: No code app creation, Thunkable. - Thunkable, Inc. - https://thunkable.com/, retrieved May 1, 2023.
7. Vailshery, L.S. (2022) Cross-platform mobile frameworks used by global developers 2021. - Statista. - https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/, retrieved May 1, 2023.
8. Kartik ; Shantnu (2020) 9 techniques to write your code efficiently - https://blogspot.com. - https://patataeater.blogspot.com/2020/08/how-to-write-efficient-and-faster-code.html, retrieved May 1, 2023.
9. Podkamennyi, V. (2020) Principles of Software Engineering. - In: Podkamennyi V. Software development. - https://vpodk.com/principles-of-software-engineering/, retrieved May 1, 2023.