https://doi.org/10.31891/csit-2024-3-4
UDC 004.412

Lidiia MAKAROVA, Maxim TATARENKO
Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine

COMPARATIVE ANALYSIS OF METHODS AND MODELS FOR ESTIMATING SIZE AND EFFORT IN DESIGNING MOBILE APPLICATIONS

Mobile applications have long become an integral part of everyday life and have gained popularity among ordinary users, small companies, and large enterprises. However, this is a completely separate area of software. We can say that this is a whole separate ecosystem. Creating functional software for this ecosystem comes with new requirements, constraints, and characteristics that were not used in previous software metrics evaluation methods such as desktop programs or web applications. Mobile applications have new features, limitations and capabilities that are different from other software.

Estimating the size and effort helps the software development team approximate the costs needed to more successfully develop a software product such as a mobile application. An early estimate of size and effort is necessary to identify resources that can help in the early stages of implementation a software project within budget and on time. The reliability of the assessment is also of great importance. Low reliability can lead to overestimation or underestimation, which can have bad consequences for any software development company, especially mobile applications. However, the suitability of methods and models for measuring size and efforts that are used for traditional software (PC applications and web applications) do not always provide sufficient reliability for mobile applications.

The main goal of this article is to analyze the usage of various methods and models for estimating size and effort the development of mobile applications, including for the Android OS at the initial stages of their design. There is a brief description of the main and widespread methods and models that are used to estimate the size and effort, such as: Lines of Code (LOC), Function Points (FP), COCOMO and COCOMO 2, COSMIC, FiSMA, application of regression analysis for construction of mathematical models. The methods and models proposed by various authors are directly analyzed.

Keywords: software metrics, measuring software metrics, size estimation, effort estimation, mobile application design.

Лідія МАКАРОВА, Максим ТАТАРЕНКО
Національний університет імені адмірала Макарова, Миколаїв, Україна

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ТА ТРУДОВИТРАТ ПРИ ПРОЕКТУВАННІ МОБІЛЬНИХ ЗАСТОСУНКІВ

Мобільні застосунки вже давно стали невід'ємною частиною повсякденного життя та завоювали популярність серед звичайних користувачів, дрібних компаній, великих підприємств. Системи, до яких раніше можна було отримати доступ через окремі програми або веб-інтерфейси, що працюють на десктопних (настільних) ПК, тепер пропонують альтернативні методи доступу через мобільні пристрої. Разом з тим, це зовсім окрема сфера програмного забезпечення. Можна сказати, що це ціла окрема екосистема. Створення функціонального програмного забезпечення для цієї екосистеми пов'язане з новими вимогами, обмеженнями та характеристиками, які не використовувалися у попередніх методах оцінювання метрик програмного забезпечення, таких як програми для персональних комп'ютерів або веб-застосунків. Мобільні застосунки мають нові функції, обмеження та можливості, які відрізняються від іншого програмного забезпечення.

Оцінювання розміру та трудовитрат допомагає команді розробників програмного забезпечення приблизно оцінити витрати, необхідні для більш успішної розробки такого програмного продукту, як мобільний застосунок. Рання оцінка розміру та трудовитрат необхідна для виявлення ресурсів, які можуть допомогти на початкових етапах реалізації проекту програмного забезпечення в рамках бюджету та у встановлені терміни. Достовірність оцінювання також має велике значення. Низька достовірність може призвести до переоцінки або недооцінки, що може мати погані наслідки для будь-якої компанії-розробника програмного забезпечення, зокрема мобільних застосунків. Однак придатність методів та моделей для вимірювання розміру та трудовитрат, які застосовуються для традиційного програмного забезпечення (програми для персональних комп'ютерів та веб-застосунки) не завжди забезпечує потрібну достовірність для мобільних застосунків.

Основна мета статті - проаналізувати використання різних методів та моделей для оцінювання розміру та трудовитрат при розробці мобільних застосунків, у тому числі і для ОС Android на початкових етапах їхнього проектування. Наводиться короткий опис основних та широко поширених методів і моделей, які застосовуються для оцінювання розміру та трудовитрат таких як: Lines of Code (LOC), Function Points (FP), COCOMO and COCOMO 2, COSMIC, FiSMA, застосування регресійного аналізу для побудови математичних моделей. Виконується безпосередньо аналіз методів та моделей запропонованих різними авторами.

Ключові слова: метрики програмного забезпечення, вимірювання метрик програмного забезпечення, оцінювання розміру, оцінювання трудовитрат, проектування мобільних застосунків.

Introduction

Systems that previously could be accessed through standalone programs or web interfaces running on desktop PCs now offer alternative access methods through mobile devices. Mobile devices have different requirements and functionality compared to PCs. The increase in the number of developers and users of mobile applications emphasizes the need for an efficient and standardized design and development process, which also includes estimating their size and effort to development (cost, time, required efforts, etc.).

In a competitive environment, companies specializing in software development are trying to maintain their reputation in the market. Including by estimating the size of the software at the early stages of its creation. This

allows better predict the timing of software development and its delivery to the end user or customer company [1]. Mobile applications often have quite complex functionality. This circumstance requires the software analyst to correctly predict the size of the future mobile application being developed, which relatively affects the cost and duration of development of such software.

With the increasing complexity of mobile applications, problems have arisen related to more efficient development. One such critical challenge is estimating the effort faced by the software project and mobile application management teams. Effort estimation helps the development team estimate the costs required to successfully develop a software product such as a mobile application.

Mobile application development requires a reevaluation of current knowledge about the processes of planning and building software systems. Mobile devices based on the Android OS (and applications for them) have different characteristics compared to traditional computer systems and programs for them [2]. Estimation processes are typically based on the characteristics of such systems in order to quantify the complexity of implementing programs for them. For this reason, it is important to review the methods and models currently proposed for estimating such software projects and analyze their applicability to estimating the size and/or effort of mobile applications, particularly for the Android OS, in the initial stages of their design.

This study is an analysis of various methods and models proposed by different authors when performing size and/or effort estimation in the initial stages of mobile application design.

Analysis of existing methods and models

There are various metrics, methods and models for estimating software size and effort. Let's look at some of them. Metrics are quantitative indicators for measuring software projects, products and resources [3]. In software engineering, the term software metrics is directly related to measurement. The right selected metrics can help management, and developers focus on their goals.

Common code size metrics include Lines of Code (LOC) and Function Points (FP). LOC is used to measure quantitative characteristics of a program's source code. FP is also often used to determine the size of programs early in the development process. The approach here is to calculate the total value of FP for the project, which depends on the number and complexity of the following five elements: external inputs and outputs (EI and EO), external inquiry (EQ), internal logical files (ILF), external interface files (EIF).

COCOMO and COCOMO 2 are development cost model. In the late 1970s, a prototype of the COCOMO model was first built. The model was described by Barry Boehm in the book "The Economics of Software Engineering" [4]. This is the most popular model for estimating the cost of software development, which has become a de facto standard, and based on the number of lines of source text (LOC). It was created based on the analysis of statistical data of 63 projects of various types. Can be applied to three types of software projects depending on team size and project complexity: Organic, Semi-detached, Embedded. The COCOMO model calculates the time, effort, cost and quality of a software project. Simply put, this model predicts the performance of a software project. The development of this model can be considered key, since from this moment not only experienced specialists, but also developers or analysts can be deal effort estimation, since to work with formal methods, very extensive development experience and knowledge in the subject field are not required.

In 1997, the COCOMO model has been improved, resulting in the COCOMO 2 model, first introduced in 2000. This model is the inheritor to the original COCOMO model and is most suitable for evaluating modern software projects. It provides fuller support for modern software development processes and is built on an updated project base. COCOMO 2 differentiate between two stages of project evaluation: a preliminary evaluation, which is carried out at the initial stage of development, and a detailed evaluation, which is carried out after the architecture has been developed.

The COSMIC method is a standard for measuring the functional size of software [5]. COSMIC is an acronym for the COMmon Software Measurement International Consortium, a voluntary organization that developed this method and is still expanding its use to more software areas. It defines the principles, rules and process for determining the functional size of software, which is a measure of the amount of functionality expressed in terms that the user can understand. Functional size is independent of the technology used to create the software. COSMIC can be used to size any type of software, including business applications, real-time software, and infrastructure software.

This method can be applied at any level of software decomposition, at any level of a layered architecture, and at any point in the software life cycle. COSMIC can be used to size software that is primarily data-serving functionality rather than software that is primarily data-manipulating. This method separate the user's functional requirements for software into combinations of four types of data movement: Enter, Exit, Read, Write.

The COSMIC method is based on a software model that analyzes user functional requirements into functional processes that consist of data movement and manipulation. It is also based on function point analysis (FPA) and is designed based on a fundamental principle of software engineering. The size of a piece of software is defined as the total number of data movements that are considered COSMIC function points (CFPs). The smallest functional process size is 2 CFP; there is no upper limit to the functional process size.

By measuring size, this method can be used to set control metrics (and subsequent estimates) for software development effort, cost, quality, and duration.

FiSMA - method for measuring the functional size of software [6]. FiSMA - Finnish Software Measurement Association is a non-profit, independent organization whose goal is to improve the quality and measurability of software development and software systems. This organization presents its method, which is called FiSMA.

FiSMA is designed based on Functional User Requirements (FUR). A portion of the functional size is measured from the user's perspective. User requirements can be considered as: functional - what the software does, and non-functional - how the software should work. Functional Size Measurement (FSM) is a common sizing method for all types of software. It is used to develop and analyze estimations. The FiSMA characteristics are service-oriented rather than process-oriented, which means that all services are identified to calculate the functional size of the software.

Measuring the functional size of an FSM works best when there is a complete list of the user's functional requirements, making content management and change management efficient, reliable, and relatively easy to understand for the end user. The correctness of the counting parameters and therefore the usefulness of the FSM method can be assessed based on the correlation between functional size and efforts.

Regression analysis is a method of modeling data that is measured and studying their properties. It allows to calculate the assumed relationship between one or more independent variables and interested of the dependent variable. Uses a given estimation method, a dependent variable, and one or more independent variables to create an equation that estimates the values of the dependent variable [7]. The independent variable (predictor) is usually denoted *X*. The dependent (estimated) variable is usually denoted *Y*. A regression model is a function of an independent variable and coefficients with included random variables. Among the regression models, the following are distinguished:

- One-parameter models (dependence on one variable);
- Multi-parameter models (dependency on several variables);
- Linear models of relatively independent variables;
- Non-linear models by variables and non-linear by parameters.

Regression analysis is used for predicting, testing hypotheses, identifying hidden relationships in data. It is used to solve the following types of problems [7]:

- Find out which independent variable is associated with the dependent variable;
- Understand the relationship between dependent and independent variables;
- Predict unknown values of the dependent variable.

Regression analysis is related to correlation analysis. Correlation analysis examines the direction and strength of the relationship between independent variables. In regression analysis, the form of dependence between independent variables is investigated. These two methods study the same relationship, but from different sides, complementing each other.

Simple linear regression takes into account the relationship between one independent variable and one interesting dependent variable. Multiple linear regression, in a generaly sense, is an analysis of the relationship between more than one independent variable and one interesting dependent variable [8].

Nonlinear regression is a special case of regression analysis in which the considered regression model is a nonlinear function that depends on parameters and one or more free variables. The difference from linear regression is only in the form of the relationship and methods of parameter estimation.

These models are not linear in nature, because they cannot be represented in the form of a simple regression model with some transformations of independent variables. Non-linear models are evaluated using explanatory variables, but linear models are evaluated using a parameter that is not particularly difficult to evaluate. In this case, substitution of variables is usually used to reduce the model to a linear one and estimate the parameters using the ordinary least squares method, which is applied to the model with substituted variables.

Linear models are easily estimated and their results are clearly interpreted. However, real data are rarely completely linear, so for their adequate description, it is necessary to resort to the construction of non-linear models.

When predicting estimating size and efforts of developing software projects, nonlinear regression equations also often used depending on certain software metrics, for example, such as Lines of Code (LOC), Function Points (FP), pages of project documentation, etc. Empirical data on the size and effort of software projects often have a non-normal distribution, since normalization of the empirical input data is usually performed to build such models. The construction of nonlinear regression models based on normalizing transformations is as follows. Empirical data are first normalized using a normalizing transformation. After that, a linear regression model is built based on the normalized data. And then a nonlinear regression model is built using the inverse of the normalizing transformation [9]. The most widespread normalizing transformations: logarithmic, Box-Cox, Johnson.

The use of standard regression analysis to derive predictive equations for software development at latest time has been supplemented by less common methods such as: neural networks, fuzzy logic models, and regression trees [8].

Literature review

Now let's look at the methods and models proposed by different authors and analyze their suitability for estimating the size and/or efforts of designing mobile applications at the initial stages, including mobile applications for the Android OS.

This article, [10] presents a method for detecting transactions and their influential complexity attributes that impact the functional size of software in Android projects. This method involves the use of both static and dynamic analysis of the source code. By analyzing transactions within the source code, the research aims to provide a more accurate basis for estimating project effort in open source Android projects.

The study emphasizes the importance of understanding transactional information to improve software functional size measurement and effort estimation in software mobile Android software.

The authors of the following research work, [11], pay the main attention to the direct relationship between effort estimation for the development of mobile applications and their size. The paper highlights the potential consequences of inaccuracies in effort estimating and discusses the importance of standardizing the approach to predicting the size of a mobile applications.

The purpose of this research is to quantify the functionality provided by software to end users, and it emphasizes the need to know the size of application to achieve this goal. This paper presents a survey-based research in which questionnaires were sent to more than 140 practitioners, researchers, and industry professionals to understand the similarity of selected parameters with current trends in mobile application development. The publication also discusses the process of collecting parameters and preparing such a questionnaire, their significance for assessing mobile efforts. In a literature review, the authors identified 104 parameters used by various researchers that potentially affect the development of mobile applications. In the end, a final list of 40 parameters was created. Out of these 40 parameters, 9 parameters were defined as Mobile Application Complexity Factor (MCF) to calculate the functional size of a mobile application. The MCF are used as the main input for the proposed model with an emphasis on obtaining the adjusted size of the mobile application, represented as adjusted mobile application COSMIC function points (AMCFP).

Finally, the paper describes future work, including modifying and revising the uncorrected COSMIC FP and MCF, revising the weighting factors, and getting more accurate results. The paper also emphasizes the importance of standardizing size measurement methods for mobile application development, as existing approaches do not provide satisfactory results for accurate estimation. It also emphasizes the importance of standardizing the approach for clearly predicting the size of a mobile application and the possible consequences of inaccuracies in effort estimating.

A rather large research [12] analyzes COSMIC and machine learning methods for size and efforts estimation on mobile application development. A hybrid approach based on machine learning and the COSMIC Plus Effort Estimation Model (CPEEM) is proposed for estimation.

16 mobile application projects were taken from the IFPUG repository for statistical interpretation and comparison of the results of the proposed model with actual values. The model uses mobile functional factors (MFF) as input parameters.

At the beginning, the analysis of requirements for mobile applications was carried out. The user selects these requirements, commonly referred as feature user requirements (FURs). FUR is functionality provided to the user through mobile applications. After the requirements were identified, data collection was performed. Both primary and secondary data were collected. Primary data were collected in the form of a questionnaire from experts in the mobile software development industry . For secondary data, 16 mobile applications from the IFPUG repository were used for statistical estimation. These input factors act as input parameters for the proposed approach.

Various parameters were analyzed, listed and sent to experts in the mobile software development industry in order to find out which of them are the most relevant at the moment for mobile applications. A list of 21 parameters was compiled. These parameters are part of the mobile function factors (MFF). MFFs are used as input parameters to calculate the overall efforts of developing mobile applications using a hybrid custom approach. The whole process includes the concepts of CPEEM effort estimation model and machine learning algorithm.

The result is an estimate of the total effort estimation of developing various mobile applications. The obtained results were verified using common statistical methods.

Thus, the study attempts to propose a model to help reduce the gap between actual and estimated effort in mobile application development. The authors note that the next work will be to develop a model based on the structure presented in reviewed article.

This research highlights the importance of analyzing mobile application development requirements and getting inputs factors such as MFF. The authors also emphasize that mobile applications differ in many aspects from traditional software and web applications. For a more accurate estimation of the size and effort estimation, it is necessary to use methodologies and models focused on mobile applications. One of these models is proposed by the authors of reviewed article. Despite all the advantages of reviewed research work, the collected number of projects most likely will not allow us to build a high-quality mathematical model.

In works [13; 14] the authors built a three-factor mathematical model based on the nonlinear regression equation using the four-dimensional Johnson transformation of the S_B family. The model was developed in order to

estimate the efforts of mobile application development at the planning phase. The following independent variables were taken as a basis: actual complexity of the development, number of screens, number of functions, number of files. A comparison of the built model with a linear regression model and nonlinear regression models based on one-dimensional normalizing transformations was also performed.

Compared to other regression models, the model proposed by the authors has better quality (larger multiple coefficient of determination R^2, smaller value of the average value of the relative error *MMRE*, larger value of the prediction percentage *PRED(0.25)* and smaller widths of the confidence and prediction intervals. But the authors do not specify for which mobile platform the source applications were taken.

This paper, [15] compares the construction of a nonlinear multiple regression model with other regression models in the context of estimating efforts for mobile application development at the planning phase. The paper also discusses the use of multidimensional normalizing transformation and outlier detection. To compare the main developed model, additional linear and non-linear regression models were built, based on one-dimensional transformations, such as the decimal logarithm, the Box-Cox transformation, and the Johnson transformation.

The authors built a main three-factor nonlinear regression model based on the four-dimensional Johnson transformation for the SB family. The model is designed to estimate the effort of mobile application development at the planning phase based on a four-dimensional data set and documentary metrics for analyzing the requirements of a mobile application: number of screens, number of functions, and number of files. Initially, 38 mobile applications were taken as input data. After outlier detection, 29 applications remained.

To check the accuracy of the prediction of the main model, standard statistical methods were applied: multiple coefficient of determination (R^2), mean magnitude of relative error (*MMRE*), percentage of prediction at the level of magnitude of relative error (*MRE*), which equalled 0.25, *PRED(0.25)*. Outlier detection was performed using a method based on the squared Mahalanobis distance (*SMD*).

The authors note that the proposed main model can be applied to evaluate the efforts of developers of mobile applications with a development period of up to three months. The authors also recommend using the constructed model for basic (simple) mobile applications, for example, which do not have network connection modules.

In terms of predicting accuracy, the main proposed model outperforms the additional built regression models. In conclusions, the authors note that further research can be directed to using another applications data sets, focused on a specific platform (framework), with the help of which mobile applications are created.

In the research work [2] the authors in detail approve that mobile applications differ from standalone desktop or any other programs. Reviewing evaluation methods such as: FPA, Data Points, COCOMO, UCP, Object Points, COSMIC, FiSMA FSM, and others mainly based on “points”, the authors applied the FiSMA method to a real mobile application - the Management of Academic Activities Integrated System (Sigaa) in its Mobile version, developed by the Superintendence of Computing (SINFO) of the Federal University of Rio Grande do Norte (UFRN).

After an in-depth systematic review of the literature, the researchers identified 29 characteristics of mobile applications, such as limited power consumption, small screen size, limited performance, variable bandwidth, changing context, limited memory, varied network connectivity, interactivity requirements, need for portability especially for native mobile applications, etc. A survey of mobile development experts confirmed that these characteristics increase the complexity and effort required to develop mobile applications compared to traditional systems. After interviewing experts and additional analysis, 13 characteristics were left as a result.

The calculations were carried out using the Experience Service tool provided by FiSMA itself. After applying FiSMA, the functional size of the software was determined, from which the efforts are found using a quite simple formula. The authors conclude that only one of the factors takes into account the characteristics of the mobile application. The authors of this work propose the creation of new performance factors that will be specific characteristics of mobile applications. An adaptation of the existing FiSMA estimation method is proposed to take into account the characteristics of mobile applications. The adapted model includes these new characteristics. These are used to adjust the effort estimate getting from the FiSMA core functional size measurement.

Considering the results presented in this reviewed publication, the final conclusion is that using any existing estimation method in mobile application development projects is not reliable enough, since some existing widely common models based on function point analysis, for example Mark II and COSMIC-FFP, turn out to be ineffective because of to insufficient consideration of the features of mobile applications. The proposed adapted FiSMA model represents a step towards eliminate this gap and providing more accurate effort estimation for mobile projects.

This research work [16] examines the suitability of the COSMIC method for measuring the functional size of mobile applications. The paper identifies pros and cons of using this method, and offers ideas to start a discussion about more accessible sizing methods for mobile software development.

The COSMIC method is considered suitable for measuring the size of any software because of its flexibility and level of abstraction, but convincing mobile developers of its benefits is difficult due to the unique characteristics of mobile application development projects. Empirical data is needed to shows the usefulness of COSMIC in mobile development, especially for small teams and projects.

In the publication, the author emphasizes to need for efficient and standardized development processes, including size and effort estimation. The author then introduced the functional size measurement method (FSM) and described the COSMIC function point method.

The author conclude that the COSMIC method is a good approach for measuring the size of mobile applications given its flexibility and level of abstraction that are suitable for a wide range of applications. However, there are challenges in convincing mobile app developers of the benefits of COSMIC due to the unique features of mobile application development such as small teams, importance of non-functional requirements, etc.

The author proposes alternative ways to improve size estimation in mobile app development, including approaches for one-person teams, approximating functional size based on natural language requirements using text mining, and using project data from the ISBSG database.

In research work [17] the authors focused on efforts estimation required for the development of Android mobile applications and the number of graphic components they contain. Comparing predictions based on measures of technical requirements and software metrics getting from source code helps evaluate the accuracy and reliability of the efforts estimation models developed in the study. That is, in other words, models created for a software project and a software product are compared.

Authors use information from requirements specification documents, such as the number of actors, use cases, and classes, to build models. These models are then compared to models created using software metrics getting from source code, such as number of classes, files, and lines of code. A dataset of 23 Android applications was used as input data. The models are based on stepwise linear regression.

The results extracted from requirements development process are no worse than the results extracted from the source code. This indicates that the requirements development process could be used to efforts estimation when designing a mobile Android product. Highlighted of potential value on the basis of using requirements development process for efforts estimation and other project management tasks in the context of the development of Android mobile applications.

The authors of the following publication [18] propose a parametric model that can help project managers efforts estimation needed to develop mobile applications. It is noted that, despite the fact that the mobile application market is quite developed, there is no calibrated and tested model for effort estimating required to develop a new mobile application.

The authors propose a parametric effort estimation model calibrated using data from more than 160 mobile applications developed by various software development companies and freelancers. First, 20 cost drivers were identified, and then forward stepwise regression was used to obtain a effort estimation model. The resulting model uses 7 (of the original 20) cost drivers and has an R^2 value of 0.949. The following cost drivers are used in the model: NoOfScrns (number of screens), ScrnCmp (screen complexity), AppCmp (application complexity), MemOptCmp (memory optimization complexity), Flex3rdPrty (flexibility of third-party components), SrvrCnfgFlex (flexibility of server configuration).

The results of the comparison with the COCOMO II model show that the proposed model gives more accurate estimates. On a validation data set consisting of 44 projects, the *PRED(30)* value was 84% for the proposed model compared to 4% for the general purpose COCOMO II. Furthermore, the *MMRE* in the proposed model is 0.20 compared to 4.67 for COCOMO II.

From these results, it can be concluded that even well-known models such as COCOMO II give very low results compared to models developed specifically for mobile applications. The authors also suppose further work to calibrate the model for companies of different sizes and different types of mobile applications.

The publication [19] suggests using Use Case Point (UCP) for early estimation size and effort in designing mobile applications. UCP is a scoring method, based of use cases diagrams. It is used to predict the size of the software during its development. It is based on the fact that the requirements for the software system are written in the form of use cases, which are part of the UML modeling methods. The size of software is calculated based on the elements of the use cases of the software system.

This work presents a practical example of the application of the UCP method to five mobile Android applications and provides an estimate of the difference between the actual and estimated efforts. A modified UCP method called M-UCP is also proposed, which includes the characteristics of mobile applications to improve the estimation results. The authors studied the characteristics of all five Android applications. Actor weights and use case weights are estimated based on the use case diagrams of these applications. A detailed description of predicting the size of one mobile application is given, starting with a use case diagram and ending with the steps of calculating the UCP.

From the results of this reviewed research, it can be concluded that the efforts estimation for all five mobile applications is downgraded in comparison with the actually spent efforts. In publication emphasizes that mobile applications have specific characteristics that differ from traditional software. Thus, traditional software estimations models need to be improved to adapt them to mobile applications, taking into account their specific characteristics, which cannot be ignored when predicting estimations. In future, the authors intend to implement the proposed M-UCP model taking into account the features of mobile applications. The M-UCP method is expected to provide a more accurate effort estimation for mobile applications compared to the original UCP method.

The work [20] published a linear regression model for estimating size of multimedia Android applications written in the Java programming language. The authors take as a basis the model of evaluation of information systems written in Java. Since the development of mobile applications for the Android OS is often conducted in the Java programming language, the choice of a similar model for comparison is quite logical.

In this reviewed work, the authors note that Android mobile applications have their own development specifics and, taking into account new of the programming area, the selected metrics of the conceptual data model in the form of a class diagram may not be sufficient for accurate evaluation, which leads to need to create an own model. The mathematical model is built according to the multiple regression equation.

Comparing the obtained results with the model that was taken as a basis for improvement, the authors obtain better values of the coefficient of determination $R^{2,}$ the average value of the relative error *MMRE*, the percentage of prediction *PRED(0.25)*, which means a higher quality of the proposed model. Therefore, the developed model is more suitable for estimating the size of multimedia Android applications.

Work [21] is devoted to the construction of a nonlinear regression model for estimating the size of mobile applications such as personal organizers created using the Kotlin programming language. This work takes into account both the type of application, namely organizing applications, and the programming language. The programs were selected and written only in the Kotlin programming language. The work also focuses on the importance of developing mathematical models for a certain type of project.

This work collected an acceptable number of projects for analysis and metrics collection, namely 49 open sources projects. The disadvantage of the work is that it has a fairly simple set of metrics to build the model. For each project, code metrics such as number of program lines of code (KLOC) and total number of classes (NC) were collected. As a consequence, the construction of a one-factor nonlinear regression model. In addition, the authors note that the model requires improvement and it is possible to apply, for example, another type of normalizing transformation or build a multifactor model.

However, after checking the quality of the resulting mathematical model, the results turned out to be quite good.

The work [22] published the main findings on the construction of a non-linear regression model for estimating software size of mobile applications created for the Android platform. A decimal logarithm is used as a normalizing transformation. A comparison of the obtained model with a linear model is also given.

The lines of code (LOC) was chosen as the independent variable, and the number of classes (NC) as the dependent variable.

The authors emphasize that today mathematical models that allow estimating software size of mobile applications created on the Android platform often do not take into account the nonlinearity of the source data and do not provide the necessary reliability.

A comparison of the obtained nonlinear model with the linear model was also performed. The resulting nonlinear model showed greater adequacy.

Conclusions

Mobile application development requires a re-evaluation of current knowledge about the processes of planning and building software systems. Mobile devices based on the Android OS (and applications for them) have different characteristics compared to traditional computer systems and programs for them. Estimation processes are typically based on the characteristics of such systems in order to quantify the complexity of implementing programs for them.

In this work, methods and models presented by different authors were analyzed. Having examined the presented works, we can conclude that:

1. General-purpose models provide are not accurate enough predictive results in relation to mobile applications.

2. Models developed specifically for mobile applications provide better results. Even though they can be quite simple (namely one factors, the decimal logarithm is used as a normalizing transformation).

3. More highly specialized models that take into account certain features, such as the target mobile platform, programming language, project type, etc., provide even more accurate predictive results.

The conclusion can be drawn that using any existing well-known estimation method in mobile application development projects is not reliable enough, since some existing widely common models, such as those based on feature point analysis, turn out to be ineffective because to insufficient consideration of the features of mobile applications. Even very well-known models such as COCOMO II give low performance compared to models developed specifically for mobile applications.

There are no universal methods or models that work for all types of software projects. Therefore, it is necessary to improve existing and develop new models when creating a project of the selected type. Taking into account additional features when building mathematical models for estimating the size of mobile applications and estimating effort for their creation for the Android OS can increase accuracy of the mathematical model in the end.

Further construction of models to improve their accuracy and quality, taking into account the characteristics of mobile applications (target mobile platform, programming language, taking into account the type of mobile application) is necessary.

References

1. Software Delivery Explained. URL: https://cpl.thalesgroup.com/software-monetization/software-delivery-explained (Last accessed: 16.07.2024).

2. Laudson Silva De Souza, Gibeon Soares De Aquino Jr Estimating the Effort of Mobile Application Development. Computer Science & Information Technology (CS & IT) Second International Conference on Computational Science and Engineering. Academy & Industry Research Collaboration Center (AIRCC), 2014. DOI:10.5121/csit.2014.4405. Pp. 45–63.

3. ISO/IEC 9126-1:2000 - Software engineering - Product quality - Part 1: Quality model. URL: https://www.iso.org/standard/22749.html (Last accessed: 19.07.2024).

4. Barry W. Boehm Software Engineering Economics. Prentice-Hall, 1981. 806 p. ISBN 978-0-13-822122-5.

5. Frank Vogelezang This is a short introduction to the COSMIC sizing method - the open source, ISO standard for software sizing. Introduction to the COSMIC Software Sizing Methodology. Cosmic Sizing. This is a short introduction to the COSMIC sizing method - the open source, ISO standard for software sizing. 2024. URL: https://cosmic-sizing.org/cosmic-sizing/intro/ (Last accessed: 16.06.2024).

6. ISO-IEC. FiSMA 1.1, Functional Size Measurement Method FSO, ISO-IEC-29881-2008. (2008). ISO-IEC, 2008. URL: https://cdn.standards.iteh.ai/samples/45724/b02bf123eed74e519c5cd72477bd21da/ISO-IEC-29881-2008.pdf (Last accessed: 26.05.2024).

7. ArcGIS Insights. Regression analysis. The mapping and data analysis technology. 2024. URL: https://doc.arcgis.com/en/insights/latest/analyze/regression-analysis.htm (Last accessed: 29.08.2024).

8. Jim Frost. Linear Regression Explained with Examples. 2021. URL: https://statisticsbyjim.com/regression/linear-regression/ (Last accessed: 16.05.2024).

9. Prykhodko S.B. The method of constructing non-linear regression equations based on normalizing transformations. Abstracts of reports of the interstate scientific and methodological conference "Problems of mathematical modeling". June 13-15, 2012, Dniprodzerzhinsk, DDTU. Pp. 31-33.

10. Kan Qi, Barry Boehm. Effort estimation of open source Android projects via transaction analysis. Journal of Software: Evolution and Process. Vol. 33, 21.02.2020. P. e2253. DOI:10.1002/smr.2253.

11. Ziema Mushtaq, Abdul Wahid. Mobile Complex Factors: An Approach For The Prediction Of Mobile Size Parameters. Recent Patents on Computer Science. Vol. 12, 18.02.2019. DOI:10.2174/2213275912666190218152109.

12. Ziema Mushtaq, Sami Alshmrany, Fahad Alturise et al. Early Size and Effort Estimation of Mobile Application Development. EAI Endorsed Transactions on Energy Web. Vol. 9, 31.05.2021. P. 8. DOI:10.4108/eai.31-5-2021.170010.

13. Prykhodko S., Prykhodko N., Knyrik K. et al. Mathematical Modeling of Effort of Mobile Application Development in a Planning Phase. International Workshop on Information-Communication Technologies & Embedded Systems. Vol. 2516, 2019. URL: https://ceur-ws.org/Vol-2516/paper7.pdf

14. Prykhodko S.B., Prykhodko N.V., Knyrik K.O. Three-factor non-linear regression equation to estimate the efforts of development of mobile applications in a planning phase. Academic notes of TNU named V.I. Vernadsky. Series: technical sciences. Vol. 5, № 1. Pp. 154–160. DOI:10.32838/2663-5941/2019.5-1/25.

15. Prykhodko S., Prykhodko N., Knyrik K. Estimating the Efforts of Mobile Application Development in the Planning Phase Using Nonlinear Regression Analysis. Applied Computer Systems. Vol. 25, № 2. Pp. 172–179. DOI:10.2478/acss-2020-0019.

16. Andre Nitze Measuring Mobile Application Size Using COSMIC FP. DASMA Metrik Kongress (MetriKon 2013). (2013). URL: https://www.academia.edu/57118674/Measuring_Mobile_Application_Size_Using_COSMIC_FP
(Last accessed: 31.05.2024).

17. Rita Francese, Carmine Gravino, Michele Risi et al. On the Use of Requirements Measures to Predict Software Project and Product Measures in the Context of Android Mobile Apps: A Preliminary Study. 2015 41st Euromicro Conference on Software Engineering and Advanced Applications. (Madeira, Portugal, 2015). Madeira, Portugal, 2015DOI:10.1109/SEAA.2015.22. Pp. 357–364.

18. Syed Ahmad Shahwaiz, Ali Afzal Malik, Nosheen Sabahat A parametric effort estimation model for mobile apps. 2016 19th International Multi-Topic Conference (INMIC)2016 19th International Multi-Topic Conference (INMIC). (12.2016). DOI:10.1109/INMIC.2016.7840114. Pp. 1–6.

19. Anureet Kaur, Kulwant Kaur. Effort Estimation for Mobile Applications using Use Case Point (UCP). 23.06.2017. P.11.

20. Ryabkov S.I., Prikhodko S.B. Improving the mathematical model for estimating the size of multimedia Android applications written in Java. Information technologies: models, algorithms, systems: coll. materials of the 1st All-Ukrainian scientific and practical Internet conference «ITMAS - 2019». (Mykolaiv: Admiral Makarov National University of Shipbuilding, 28.11.2019). Mykolaiv: Admiral Makarov National University of Shipbuilding, 2019. Pp. 10–12.

21. Telekhan A.M., Makarova L.N. A nonlinear regression model for estimating the size of personal organizer applications created in the Kotlin language for Android. *Information technologies: models, algorithms, systems: collection of materials of the III All-Ukrainian scientific and practical Internet conference «ITMAS - 2022»*. (Mykolaiv: Admiral Makarov National University of Shipbuilding, 2022). Mykolaiv: Admiral Makarov National University of Shipbuilding, 2022. Pp. 27–29.

22. Makarova L.N., Seryogin Y.M. Improving the mathematical model for estimating the size of mobile applications created on the Android platform. *Information technologies: models, algorithms, systems: coll. materials of the 1st All-Ukrainian scientific and practical Internet conference «ITMAS - 2019»*. (Mykolaiv: Admiral Makarov National University of Shipbuilding, 27.11.2019). Mykolaiv: Admiral Makarov National University of Shipbuilding, 2019. Pp. 70–72.

Lidiia Makarova **Лідія Макарова**	PhD, Associate Professor of the department of automated systems software, Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine. e-mail: lidiia.makarova@nuos.edu.ua https://orcid.org/0000-0003-2903-3001	кандидат технічних наук, доцент, доцент кафедри програмного забезпечення автоматизованих систем, Національний університет кораблебудування імені адмірала Макарова, Миколаїв, Україна.
Maxim Tatarenko **Максим Татаренко**	PhD student of the department of automated systems software, Admiral Makarov National University of Shipbuilding, Mykolaiv, Ukraine. e-mail: multi3volume@gmail.com https://orcid.org/0009-0009-3330-0169	аспірант кафедри програмного забезпечення автоматизованих систем, Національний університет кораблебудування імені адмірала Макарова, Миколаїв, Україна.