

INFORMATION TECHNOLOGY FOR ESTIMATING THE SIZE OF SOFTWARE PROJECTS

Oriekhov O., Farionova T.

The aim of the study is an information technology development for automated processing of the information from source code metrics for the software size estimation using nonlinear regression models. The object of the study is the process of the source code metrics information processing for the software size estimation using information technology. The subject of the study is information technology for source code metrics information processing for the software size estimation. A review of the usage of software development effort estimation displays that nowadays, software development companies and teams start to use effort estimation in their CI/CD tools and require efforts from DevOps engineers to integrate the software solutions in their IT infrastructure. The study offers a software-as-a-service information technology solution for software size estimation for integration into the CI/CD infrastructure of software development companies. A system design is given as a sequence diagram and a class diagram. The information technology is implemented as a software-as-a-service solution to offer possibilities for integration into the CI/CD infrastructure of software development companies.

Introduction

Estimating the size of a software project is a critical task in software development effort estimation. Accurate size estimation helps project managers plan resources, set deadlines, and control budgets. Traditionally, software size estimation in thousands of lines of code (KLOC) is used in parametric models such as COCOMO, COCOMO II, COSYSMO, SLIM, etc. Software development effort estimation in a KLOC basis is more accurate in comparison with functional points, because the KLOC variant takes into account such environmental factors as programming language and software category and offers alignment of the efforts with the evolution of the programming language. A disadvantage of the KLOC usage is the requirement to have relevant mathematical models for an accurate estimation, because some math models could be missing or to be obsolete with intensive evolution of the programming language [1].

Software development effort estimation remains an actual task that every software development company faces before starting any new software development project or continuing to implement existing software products. The Standish Group report [2] indicates a moderate positive trend in the share of successfully completed projects. In 1994, only 16% of software projects were successfully implemented, meeting deadlines, staying in the budget range, and fulfilling all declared

requirements. By 2020, the share of successfully implemented projects had risen to 35%. The percentage of failed projects decreased from 31% in 1994 to 19% in 2020. The share of projects which are facing challenges, such as exceeding deadlines, overusing budgets, or failing to meet all declared functional requirements, has shown a downward trend of approximately 4%. Also, the research [2] demonstrates a strong correlation between project size and failure probability or potential challenges and risks during software project development. Larger software projects are more difficult to manage because they involve bigger resource allocation, wider coordination, and they have higher probability of unforeseen obstacles. The CHAOS Report findings show that large projects require more accurate estimation models and risk management strategies to improve their chances of success. Without accurate effort and size estimation at the initial planning stage, the expectation may be mismatched with actual development realities and, it leads to an increased probability of project failure.

Nowadays, traditional software exists for software development effort estimation: SEER-SEM, COCOMO II, QSM SLIM, EstimateTool, IBM Rational Tools, etc and, alongside with traditional software, there are continuous integration and continuous delivery (CI/CD) tools for software size and software development effort estimation are demanded. However, as software development becomes more dynamic and continuous, especially with the use of Agile and DevOps practices, there is a growing demand for more flexible and integrated software estimation tools. In this context, CI/CD tools are being used not only to automate building, testing, and deployment, but also to support software size and software development effort estimation in real-time. Integrating SDEE tools into CI/CD pipelines allows teams to automatically estimate effort when changes are made to a UML class diagram, a codebase is modified, a new feature is added, or the existing code is refactored. These estimates can be generated and updated continuously throughout the development process. Moreover, the integration enables comparison between expected and actual development effort in an automated manner, supporting better planning, prediction, and project control without manual input [3-5].

As we can see, active growth and injection of CI/CD tools in the software planning and development process requires transformation of the traditional SDEE tools to offer services that could be injected in automation of the software size and development effort estimation process at the IT infrastructure side. To achieve the automated code metrics processing and usage of modern tools for the software size and effort estimation, it is necessary to propose scalable, easy to update and maintain software solutions. In this case, to provide appropriate and relevant math models,

the Software-as-a-Service (SaaS) distribution model guarantees that modern software size estimation tools will always be up to date, can be easily integrated into the IT infrastructure, and ready for use in CI/CD pipelines for software development efforts assessment.

Therefore, the aim of the study is an information technology development for automated processing of the information from source code metrics for software size estimation using nonlinear regression models. The object of the study is the process of the source code metrics information processing for the software size estimation using information technology. The subject of the study is information technology for source code metrics information processing for the software size estimation.

Modeling of the information technology for software size estimation

The SaaS distribution model was chosen to provide easy access to software size calculation possibilities. It ensures fast updates, maintenance, and improvement of the software application without user intervention, providing instant access to new mathematical models, software features, security enhancements, and bug fixes; providing an application programming interface (API) for integrating KLOC estimation into the software tools of customer's services or existing IT infrastructure, allowing for the creation of comprehensive solutions combined into a single infrastructure. This approach ensures high compatibility and flexibility, allowing organizations and companies to use calculation tools as part of their IT environment, which improves the efficiency of analytical assessment of software development projects [6].

The SaaS information technology for processing information from code metrics to evaluate the size parameter of the software project should solve the following tasks:

- 1) information code metrics processing for estimating the average KLOC software size of software applications, depending on a specific programming language in the early stages of software project planning, using mathematical models that are based on quantitative code metrics.

- 2) the confidence interval bounds estimation of the average KLOC size of software applications depending on a given confidence probability;

- 3) the prediction interval bounds estimation of the average KLOC size of software applications depending on a given confidence probability;

- 4) providing an API for integrating software into customer systems;

5) returning a list of existing mathematical models for estimating information from software code metrics depending on a programming language, that declares necessary code metrics and their acceptable value ranges.

Mathematical background for the IT for estimating the size of software applications considers mathematical models for KLOC estimation depending on a programming language and independent factors that could be obtained from UML class diagrams. The three-factor nonlinear regression models for Data Science and Machine Learning JAVA applications size estimation, the five-factor non-linear regression models for JAVA applications size estimating, and three-factor non-linear regression model for Kotlin applications size estimation are selected for the initial integration into the SaaS information technology [7–9]. These models were built upon modern open-source software projects and offer software size estimation for the highly used JAVA programming language, and intensively growing Kotlin programming language.

The target audience of the SaaS is software development engineers, DevOps, scientists, and project managers. The users have the necessary experience to obtain software size estimates and integrate it into the IT infrastructure of a company using the SaaS variant and the declared documentation. The entire audience is declared as a user role for the IT. Another role is an administrator to provide access to manage the IT, and it will be omitted in the further use case description below.

The user role has the following specification of **use cases** of the SaaS:

1) Obtain information about mathematical models

The user receives a list of available mathematical models that can be used for calculations or obtains information about a mathematical model by its identifier. The list of parameters includes the programming language, the category of the mathematical model, or the identifier of the mathematical model in the system.

The main flow of the use case:

- the user sends a request for information about mathematical models;
- the system checks the request parameters and proceeds with it;
- the system forms a list of available mathematical models in JSON format or a single model in JSON format if the request was made using a unique model identifier.

Alternative flows:

- if the request or input data is incorrect, the system returns an error (HTTP 400 Bad Request);

- if the mathematical model is missing, the system returns an error (HTTP 404 Not Found);
- if the user is not authorized, the system returns (HTTP 401 Unauthorized).

2) Calculate the software size KLOC value using the selected mathematical model

The user forms a request with the values of the declared independent factors of the mathematical model and the significance levels α for calculating prediction intervals and confidence intervals. The system returns the calculation results. The pre-condition of the use case – the math model is selected for the calculation.

The main flow of the use case:

- the user selects the mathematical model's identifier for the query;
- the user sends the input data form for calculating the dependent factor;
- the system validates the query parameters and performs calculations of the dependent factor, as well as prediction intervals and confidence intervals;
- the system forms the calculation result in JSON format.

The SaaS platform's estimation workflow is presented in Figure 1 in the form of a UML activity diagram that illustrates behaviour of the two use cases. It visualizes the sequential interaction between use case № 1, «Obtain information about mathematical models», and use case № 2, «Calculate the software size KLOC value using the selected mathematical model». The diagram highlights the flow of actions and decision points required to retrieve model parameters and perform calculation of software size estimates.

The declared workflow of the SaaS offers all necessary opportunities to integrate the IT solution into CI/CD infrastructure of the software companies and build continuous software size and software development effort estimation solutions.

Overview of the SaaS system design

Information support for the SaaS solution for code metrics information processing includes a software development solution, a database, and predefined data models' structures designed to store the parameters of nonlinear regression models for evaluating the size of software applications and the parameters for evaluating the bounds of confidence and prediction intervals. This approach allows adding new models or refining existing parameters of the math models without recompiling the software solution, which allows avoiding the CI/CD flow breaking.

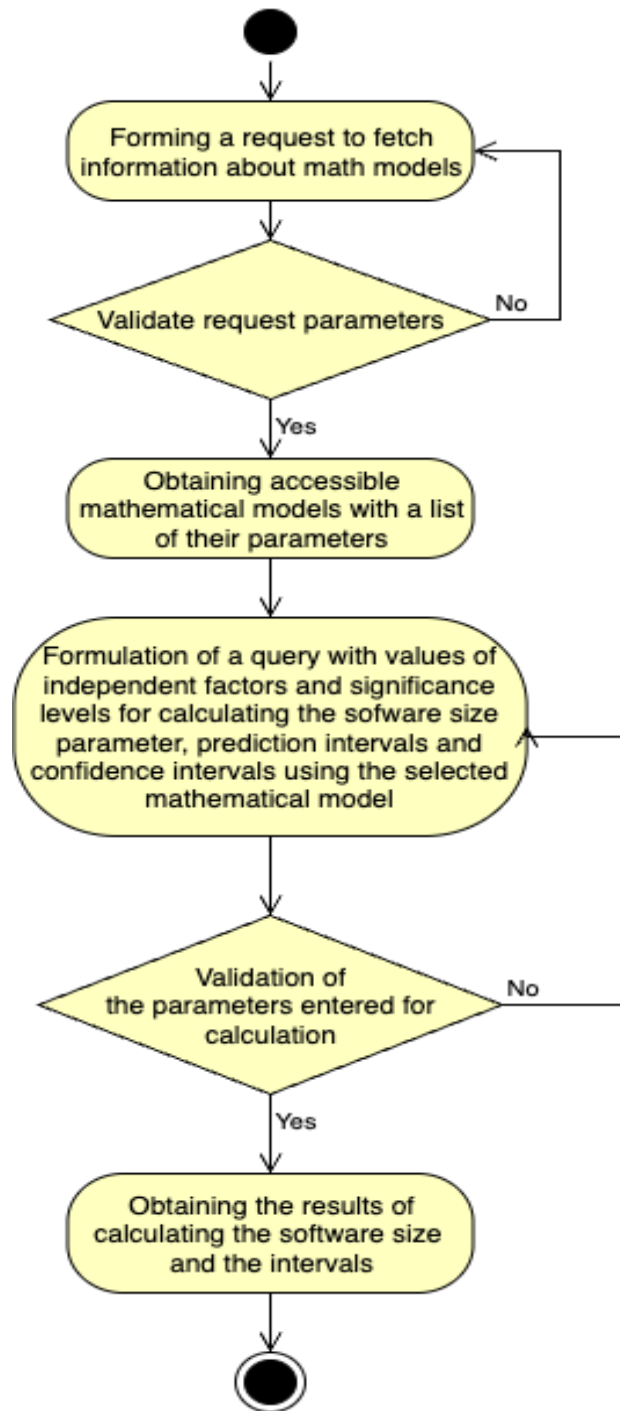


Fig. 1. UML-activity diagram of the software size estimation workflow

The Kotlin programming language with the Spring Framework was chosen as the programming language for developing the estimation service to build network layer interaction. Kotlin is a high-level object-oriented programming language that is distinguished by its speed and performance, and it is fully compatible with the JAVA programming language, which is a great advantage, since a large number of libraries and frameworks have been developed in JAVA, allowing for the implementation of a variety of functionalities, including mathematical calculations [10].

Spring Framework was chosen as the framework for developing the web application due to its advantages, such as a free distribution model, a wide range of tools for implementing network functionality and support for REST API, quick startup and automatic configuration, which reduces the amount of manual configuration for various network interaction components, integration with databases, provides support for working with JSON data, resource access control and support for standards such as OAuth2 and JWT (JSON Web Tokens), which is especially important for network applications such as SaaS, where data security is critical [11].

Technical support for the SaaS information technology has the following hardware requirements: CPU with at least 1 GHz, recommended CPU should have 2 GHz or higher, RAM size should be minimum 512 MB, recommended 1 GB or higher, minimum disk space is 500 MB of free space, internet connection. The hardware requirements are based on the JVM requirements [12].

The user's hardware environment requires uninterrupted access to the Internet. The user's hardware and software must allow requests to be made via the HTTPS protocol (SSL/TLS support) over the Internet [13].

The logical model of the IT takes into account different programming languages, categories, and multivariate parameters structure of the math models. It includes a unique identifier to store all different models in the system, programming language, category, math model name, math model description, quantity of the independent factors, list of the independent factors, dependency factor, datetime and reference when the model were published, accuracy testing information, model restriction information, and a complex data structure to store the math model coefficients and constants.

A UML class diagram was created for the SaaS. The class diagram shows the main entities of the system, their relationships, and interactions between them. The class diagram is shown in Figure 2.

The software information processing SaaS has the following requirements for the software on the server-side environment [12].

Operating system requirements for running the SaaS application:

- Windows: Windows 10, Windows 11, Windows Server 2016, 2019, 2022;
- Distributions that support glibc 2.17+ (e.g., Ubuntu 18.04+, CentOS 7+, etc.);
- Mac OS: macOS 10.15 (Catalina) and newer.

Runtime environment requirements are Java Runtime Environment 8 or higher, PostgreSQL database server version 14 or higher, and OpenSSL version 1.1.1 or newer.

Conclusion

The results of the study propose an automated processing of the information from code metrics for software size estimation at an early stage of the software project development and provide software tools solutions for integrating the calculator in CI/CD infrastructure of a company for software size estimation or effort monitoring systems.

The practical significance of the obtained results allows us to recommend the described software construction, or the software estimation tools for use in practice for integration in software development life cycle monitoring tools for use with COCOMO II, COSYSMO, etc, to obtain effort estimates.

Prospects for further research may include extending the software metrics estimation service software with a wider range of models for estimating the size of projects for different programming languages and adding software development effort estimation models like COCOMO and COCOMO II.

References

1. Farionova, T. A., & Oriekhov, O. S. (2024). A review of software development effort estimation methods. *Collection of Scientific Papers of Admiral Makarov National University of Shipbuilding*, 494, 102–111. DOI: [https://doi.org/10.15589/znp2024.1\(494\).15](https://doi.org/10.15589/znp2024.1(494).15)
2. Johnson, J., & Mulder, H. (2021). Endless Modernization: How Infinite Flow Keeps Software Fresh. *The Standish Group, Hans Mulder's Lab*. URL: https://www.researchgate.net/publication/348849361_Endless_Modernization_How_Infinite_Flow_Keeps_Software_Fresh
3. Meedeniya, D., & Thennakoon, H. (2021, August). Impact factors and best practices to improve effort estimation strategies and practices in DevOps. In *Proceedings of the 11th International Conference on Information Communication Management (ICICM 2021)*, pp. 11–17. ACM. DOI: <https://doi.org/10.1145/3484399.3484401>
4. Tyagi, A. (2021). Intelligent DevOps: Harnessing Artificial Intelligence to Revolutionize CI/CD Pipelines and Optimize Software Delivery Lifecycles. *Journal of Emerging Technologies and Innovative Research*, 8, 367–385.
5. Port, D., Taber, B., & Emkani, P. (2024). Investigating effectiveness and compliance to DevOps policies and practices for managing productivity and quality variability. *Journal of Systems and Software*, 213. DOI: <https://doi.org/10.1016/j.jss.2024.112030>.
6. Golding, T. (2024). *Building Multi-Tenant SaaS Architectures*. O'Reilly Media.
7. Oriekhov, O., & Farionova, T. (2024). Nonlinear regression models for software size estimation of Data Science and Machine Learning JAVA-applications. In *Proceedings of the V International Workshop IT Project Management (ITPM 2024), Bratislava*. URL: http://dx.doi.org/10.23939/IW_itpm2024.054
8. Oriekhov, O., & Farionova, T. (2025). Five-factor nonlinear regression model for JAVA software size estimation. In *Proceedings of the V International Workshop IT Project Management (ITPM 2025), Kyiv*.

9. Prykhodko, S. B., Prykhodko, N. V., & Koltsov, A. V. (2024). A nonlinear regression model for early LOC estimation of open-source Kotlin-based applications. *Radio Electronics, Computer Science, Control*, 85(1), 85–95. DOI: <https://doi.org/10.15588/1607-3274-2024-1-8>.
10. JetBrains. (2025). *Kotlin programming language*. KotlinLang. URL: <https://kotlinlang.org/>
11. Spring. (n.d.). Spring Boot. URL: <https://spring.io/projects/spring-boot>
12. Oracle. (n.d.). *Oracle JDK 8 and JRE 8 Certified System Configurations*. URL: <https://www.oracle.com/java/technologies/javase/products-doc-jdk8-jre8-certconfig.html>.
13. Rescorla, E. (2018). *The transport layer security (TLS) protocol version 1.3 (RFC 8446)*. Internet Engineering Task Force. DOI: <https://doi.org/10.17487/RFC8446>