ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

UDC 004.413 DOI: https://doi.org/10.30837/0135-1710.2026.189.196

Ivan Kushnir, Valentyna Kyrii

SOFTWARE DEVELOPMENT EFFORT ESTIMATION MODEL BASED ON PRE-TRAINED ZERO-SHOT MODELS

Project management of software development involves planning, execution, and control of work to ensure adherence to budget and deadlines. The subject of the study is effort estimation, which is one of the most critical tasks – it allows managers to minimize risks and optimally allocate resources. With the emergence of artificial intelligence (AI) and the continuation of digital transformation, the application of natural language processing (NLP) methods and large language models (LLM) has become crucial. These tools can analyze textual requirements and user stories to predict the necessary effort. They enable semantic structuring of complex textual requirements and improve estimation accuracy. Using zero-shot models eliminates the need for additional training on target data, saving resources and ensuring greater adaptability. The goal of this work is to develop a model for effort estimation based on software requirements presented in various forms. This is achieved through the following tasks: to develop an effort estimation model for software development based on modern pre-trained zero-shot models, and to conduct an experimental study to evaluate the accuracy of the proposed models. The study used a public dataset and AI models with varying architectures, training methods, and licensing terms. The model evaluation was performed using metrics of mean absolute error, mean relative error, and root mean square deviation. As a result, an effort estimation model for software development was built, which in the experiment showed that the accuracy of effort prediction largely depends on the nature of the dataset and the style of formulating user stories. Compared to generalized models from previous studies, training project-specific models significantly increases prediction accuracy. The proposed effort estimation model based on zero-shot vectorization effectively automates the evaluation of user stories, facilitating resource planning at various stages of the software development lifecycle. The obtained results open the door to the flexible, scalable application of NLP based on large language models in software development project management.

Keywords: project management; software development; effort estimation; artificial intelligence; large language models; text embedding.

1. Introduction

Software development project management involves the systematic planning, execution, and control of software development activities to ensure the delivery of high-quality products within budget and on schedule. Quantifying the resources and time required to complete a project is one of the core processes in project management. Project managers need accurate estimates to plan their activities, minimize risks, and achieve desired outcomes. An inaccurate estimate can lead to project failure, so this critical process must be approached with care.

Project management theory has evolved in various directions. These include significant engineering achievements and the use of advanced technologies. Project management can be described as the strategic application of knowledge, combined with the experience, tools, and methods necessary to achieve desired goals by managing various resources and risks along the path of plan execution [1]. The rapid digital revolution, which was partly driven by the 2020 pandemic crisis, has led to radical changes in project management practices. The widespread use of artificial intelligence (AI) is now a driving force, as it takes on routine tasks and increases efficiency, while allowing project managers to focus on the strategic aspects of management. The integration of AI into project operations is taking place across all critical functions, such as resource planning and risk management, as well as team communication. This is prompting project managers to explore new opportunities for using AI in the workplace [2].

Artificial intelligence is generally defined as the simulation by machine systems of human intellectual processes, including the abilities to learn, think logically, make decisions, and solve problems [3–4]. It encompasses highly specialized applications designed for specific functions, as well as general-purpose systems capable of performing any intellectual tasks that a human can perform [5–6]. These systems have a significant impact on software development project management, particularly in the area of estimating development effort.

Despite noticeable progress in the development of project management across various areas, the issue of effort estimations during software development remains challenging. At its core, it determines the success of a project, as deadlines, budget allocation, and the consistency of the final product's quality depend on the

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

accuracy of these estimates [7–8]. From a practical standpoint, it can be agreed that traditional methods of preliminary labor cost estimation boil down, for the most part, to three approaches: expert estimates, algorithmic models, and methods based on the analysis of historical data.

The latter group includes, in particular, the analogy-based estimation method, which uses information about completed projects to forecast future labor requirements [9, 10]. The adoption of agile methodologies, with their iterative nature and user-story-oriented structure, creates additional challenges for applying traditional estimation paradigms, shifting the focus to the analysis of individual tasks rather than the project as a holistic system [11]. Such dynamics complicate the adaptation of the analogy-based estimation method within agile project management approaches, where the constant evolution of requirements and informal, "noisy" textual specifications – particularly user stories – cause significant difficulties [12]. Textual requirements, which are characterized by domain specificity, terminological ambiguity, and often combine natural language with source code or quantitative metrics, require the use of complex semantic processing methods to identify relevant labor cost indicators [13].

The use of natural language processing (NLP) methods in combination with AI allows for the systematization of text processing. NLP effectively identifies key concepts, establishes semantic relationships between them, and gradually transforms unstructured descriptions into semantically coherent representations [14, 15]. Word embedding models are among the most widely used in addressing these challenges. In such models, each word is converted into a numerical vector generated based on large text corpora, which allows for the consideration of hidden semantic relationships between terms. This is precisely what enables machine learning models to interpret the content of specifications and accompanying textual documentation more accurately [16]. It is also worth noting the impact of large language models (LLMs) built on transformer architectures and attention mechanisms. Such models form contextual representations not only of sentences but also of documents as a whole. This improves the accuracy of extracting key ideas from large text corpora, which is particularly important when working with software requirements [17, 18].

The aim of this study is to develop a model for estimating the effort required for software development using pre-trained LLM zero-shot models, based on requirements presented in various forms.

2. Literature Review and Problem Statement

Estimating the effort required for software development projects is a relevant area of scientific research and a practical task.

In [19, 20], current trends in approaches to effort estimations in software development projects were analyzed. The authors highlight machine learning and artificial intelligence methods as key trends in data analysis and the construction of resource cost prediction models. The paper highlighted the features of natural language processing (NLP) models as tools for analyzing textual requirements when effort estimations.

The authors of [21] conducted a comprehensive review of existing studies that use machine learning methods to forecast labor costs in software development. The paper highlighted the high effectiveness of such models based on public and private datasets. However, these models were largely based on algorithmic approaches to estimation, which in turn require large amounts of historical data and demonstrate low effectiveness when working with textual requirements, as noted in [19].

Studies [22, 25] examined the combination of NLP and artificial intelligence methods for estimating resource costs in software development projects. The authors used the codebase as input data in combination with classical estimation models, such as COSMIC and COCOMO. Although the results demonstrate significant effectiveness, it should be noted that the codebase becomes available primarily in the later stages of the development lifecycle, whereas text-based requirements, as artifacts of the early phases, allow for cost estimates to be obtained at earlier stages of project planning.

In [11], a deep learning model is proposed for predicting effort using story point estimates. The described approach combines deep learning algorithms for generating text embeddings with their subsequent integration into prediction models. The authors note the significant potential of applying machine learning methods to the analysis of textual requirements. Among the key challenges, they highlighted the search for and preparation of relevant data for training models. It is this work that laid the

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

foundation for a number of subsequent studies and is often used as a benchmark.

Expanding on this analysis, the authors [13] conducted a comparative analysis of various text vectorization models, specifically the context-free word2vec and the contextualized BERT models. The contextualized models demonstrated better results, providing higher accuracy in estimating effort in story points. In addition, it was found that the selection of model parameters can significantly affect prediction results.

In the study [24], the FastText model was used for text data vectorization followed by effort classification. The authors treat estimation not as a regression task but as a classification task, clustering the data and applying semi-supervised noise filtering. Afterward, text requirements are transformed into vectors using FastText, and a classification layer is implemented to determine the task complexity level (XS, S, M, L, XL). When using experts for data cleaning, this solution demonstrates good results; however, it does not account for other approaches to text vectorization and is limited to a fixed number of classes, which is a drawback compared to other prediction models.

In [25], the application of pre-trained FastText, GPT-2, and SBERT models in combination with machine learning algorithms, specifically Random Forest and Support Vector Machine (SVM) methods, is analyzed. This work compiles and analyzes existing state-of-the-art models [11, 13, 26]. The authors once again confirmed the high effectiveness of combining modern text vectorization methods with classical machine learning algorithms and developed a model that performs on par with existing ones and even shows better results in some metrics. However, the authors remain focused on using pre-trained models primarily on domain-specific datasets, which limits their scope of application compared to zero-shot models. They also do not consider models capable of processing queries submitted in other formats, such as graphical, multimedia, or formalized.

Thus, thanks to existing research, a set of models has already been established [11, 13, 25, 26] that can serve as a benchmark for future work. At the same time, it should be emphasized that these works leave a significant gap in the systematic evaluation of various pre-trained zero-shot vectorization models. They are mostly focused either on large, finely tuned models or on classical vectorization methods without zero-shot properties. However, the use of zero-shot models can facilitate the widespread adoption of effort estimation models built on their basis, as these models will not be domain-dependent. Also, most modern works consider only textual representations of requirements when forecasting the required resource volumes, whereas software development projects are not limited in their choice of how to represent requirements.

3. Research objectives and tasks

The objective of this work is to develop a model for effort estimations based on software requirements presented in various forms. This will allow for the effective estimation of human resource costs for software development at various stages of the project lifecycle.

To achieve this goal, the following objectives were formulated:

– to develop a model for effort estimations for software development based on modern pre-trained zero-shot models;

– conduct an experimental study to evaluate the accuracy of the proposed model, implemented using various pre-trained zero-shot models, on public datasets.

4. Materials and methods

This work uses a set of text-based requirements presented in [11, 13, 25]. This text dataset contains descriptions of requirements in the form of user stories. Labels corresponding to the development effort for each story, measured in story points, are attached to this dataset. It is important to note that all text requirements in this corpus are presented in English.

The dataset was collected from large open-source project management systems; it contains 23,313 requirements distributed across 16 projects, including well-known projects such as Apache, Appcelerator, DuraSpace, Atlassian, Moodle, Lsstcorp, Mulesoft, Spring, and Talendforge.

Despite the challenges of accurately measuring the actual labor costs required to implement requirements, the authors of the dataset were able to estimate the time taken to complete project phases based on changes in requirement statuses. Specifically, effort was recorded as the time interval from the "in progress" status to the update to "resolved". In [11], statistical analysis using Spearman’s and Pearson’s correlation tests demonstrated a significant relationship between assigned story points and actual recorded effort.

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

To cover a wide range of architectural and methodological trade-offs, seven competing text vectorization models were selected, identified based on recent large-scale comparative studies [27, 28]. These models differ in vector dimension, training methods, and licensing terms, which allows us to investigate the impact of these factors on the effectiveness of solving applied tasks.

Let us consider the models in order of the dimension of the resulting vector representations of text.

The group with the highest vector dimension includes three models. The Gemini Embedding model (gemini-embedding-001) from Google [29] generates 3,072-dimensional vectors and is built on the large Gemini language model, demonstrating consistently high results in multilingual and English-language benchmarks. It was evaluated on the Massive Multilingual Text Embedding Benchmark (MMTEB) [27], which covers over 100 tasks in more than 250 languages, consistently outperforming previous state-of-the-art solutions in classification, similarity measurement, clustering, ranking, and search. OpenAI's text-embedding-3-large model [30] also generates 3,072-dimensional vectors and is the company's most productive solution in the field of text vectorization, improving MIRACL scores from 44.0% to 54.9% and MTEB scores from 62.3% to 64.6% compared to the smaller text-embedding-3-small version. The reduced model generates 1,536-dimensional vectors – placing it at the lower end of this group – and provides a more computationally efficient and cost-effective option, while maintaining performance at 44.0% on MIRACL and 62.3% on MMTEB [27, 31].

The medium-dimensionality group consists of two models with 1024-dimensional vectors. The Qwen3-Embedding-0.6B model [32] is an open-source solution built on the large language model of the Qwen3 family. It combines scaled unsupervised pre-training with supervised fine-tuning on data covering various domains and languages. The model demonstrates high performance on the multilingual MMTEB tests and in code search tasks [27]. The multilingual-e5-large-instruct model [33] was evaluated simultaneously; it is an open-source, instruction-enhanced vectorization model. It was trained on one billion multilingual text pairs using a contrastive approach and further fine-tuned on labeled datasets. The E5-large-instruct variant demonstrated results that match or exceed those of proprietary models focused solely on English-language tasks. The two models with the lowest dimensionality in our evaluation generate 768-dimensional vectors. The gte-multilingual-base model [34] is an open-source multilingual text encoder optimized for long-context scenarios. It supports processing long text contexts of up to 8,192 tokens while maintaining high vectorization quality. Architecturally, this model is based on a transformer with an encoder-only architecture, which reduces the model size and ensures significantly lower hardware requirements during execution compared to models based on a decoder-only architecture. This enables nearly a tenfold increase in inference speed. The second model with 768-dimensional vectors – Gecko (text-embedding-005) from Google [35] – is a proprietary solution designed to achieve high search accuracy with a compact vector representation. It implements a new two-stage process for distilling knowledge from large language models (LLMs), which uses a synthetic paired training dataset combined with the selection and relabeling of text fragments.

Table 1. *Models, their dimensions, and MTEB scores*

	Embedding size	**MTEB[24]**
gemini-embedding-001	3072	1
text-embedding-3-large	3072	17
text-embedding-3-small	1536	33
Qwen3-Embedding-0.6B	1024	4
multilingual-e5-large-instruct	1024	7
gte-multilingual-base	768	27
text-embedding-005	768	170

To estimate the labor costs of software development, a feedforward neural network architecture was selected, based on the recommendations and proven effectiveness of similar models in [13].

A feedforward neural network is characterized by a simple and straightforward organization of information flows without feedback, which makes feedforward networks effective tools for solving approximation, classification, and regression problems. Such networks can have two or more layers, as shown in Fig. 1. The first layer is the input layer, and the last is the output layer. The additional layers between the end layers are called hidden layers, whose neurons are called hidden neurons. These neurons act as intermediaries between the input data coming from the external environment and the network's output. It is the number of hidden layers that determines the degree of complexity of the function, which reflects the relationship between the network's input and output vectors. In particular, a single-layer

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

network describes first-order relationships, a two-layer network describes second-order relationships, and so on.

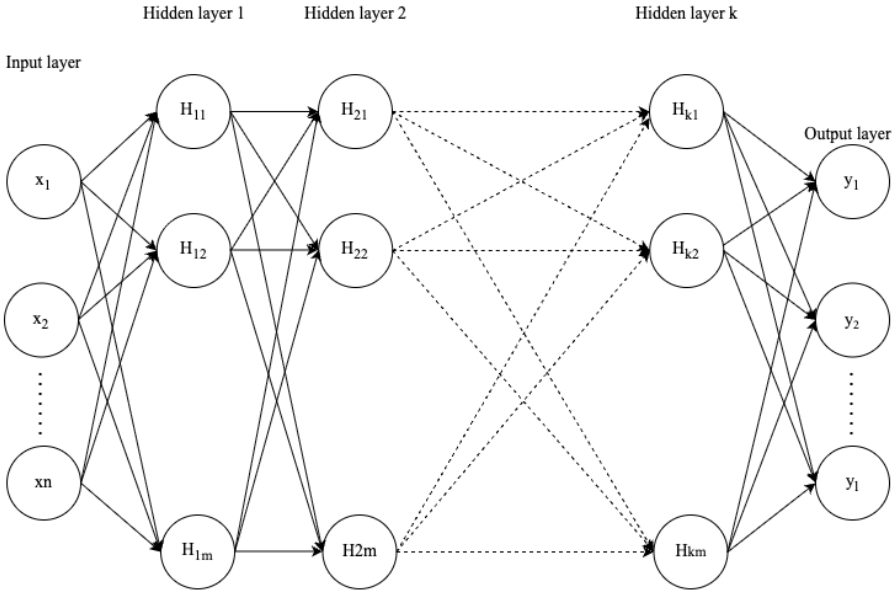

Fig. 1. Architecture of a feedforward neural network

Structurally, the neurons in the input layer form the elements of the input vector (N), which is fed to the neurons of the next layer. This can be expressed as a formula:

$$N_j = \sum_{i=1}^{n} Iw_{ij} x_i + B_j ,$$

where n – dimension of the input layer, Iw_{ij} – the weight matrix for the i-th input parameter and the j-th neuron in the hidden layer, B_j – the displacement vector of the hidden layer's j-th neuron.

This input vector is then passed through an activation function, producing an output vector that serves as the input for the next layer. It is calculated using the formula:

$$H_{ij} = f\left(N_j\right),$$

where i – hidden layer number, j – the neuron number in this layer and f – activation function.

The output signals from the neurons in this hidden layer serve as input for the next layer, and this process is repeated sequentially up to the output (final) layer. The calculation of the output layer's values can be expressed as a formula:

$$y_j = \sum_{i=1}^{m} Hw_{ij} H_i ,$$

where m – the thickness of the last hidden layer, Hw_{ij} – the weight matrix between the i-th neuron of the hidden layer and the j-th output parameter.

Thus, the set of output signals from the neurons in the final layer determines the network's overall response to the input signal generated by the neurons in the input layer.

Model quality was evaluated using a combination of three metrics: mean absolute error (MAE), root mean square error (RMSE), and mean mean relative error (MMRE), which are among the most popular and widely used methods for evaluating models in this field [19]. MAE measures the mean absolute value of errors in the same units as the target variable, allowing for an intuitive assessment of typical deviations of predictions from actual values. It is calculated using the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i\right| .$$

RMSE amplifies the impact of large forecast errors on the model's overall accuracy by squaring the error values. It is calculated using the formula:

$$PMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i\right)^2} .$$

To assess relative accuracy, the MMRE metric was used, which normalizes prediction errors by dividing them by the actual values; this is particularly useful in fields where project sizes vary widely, as relative errors

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

provide a more meaningful assessment than absolute errors. It is calculated using the formula:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - y_i\right|}{y_i}$$

Two approaches were used to work with embedding vector generation models. The first approach involved running the models locally using the relevant Python libraries, but this is only possible for publicly available models. In the case of proprietary models developed by Google and OpenAI, vectorization is implemented via API calls to the respective services.

5. Research results

5.1. A model for estimating the labor costs of software development based on modern pre-trained zero-shot models

Let's consider the general model for estimating the efforts required for software development, shown in Fig. 2.

The input data consists of software requirements, which are documented and stored in a requirements management system; this may be a subsystem of a general project management system. The model's governing elements include domain standards, text vectorization methods, and project management methods. The model's output is a numerical representation of the labor costs required to develop the specified requirements.

Let us note the model's limitations. This model is configured for a specific project or similar projects, as the assessment of software development complexity depends heavily on the team and the domain. Therefore, the historical data used to train the model must come from this project or a very similar one. The model is designed to work with numerical estimates of the labor costs for software development. If the development complexity assessment uses other metrics, such as the "T-shirt size" estimation method, it may not work.

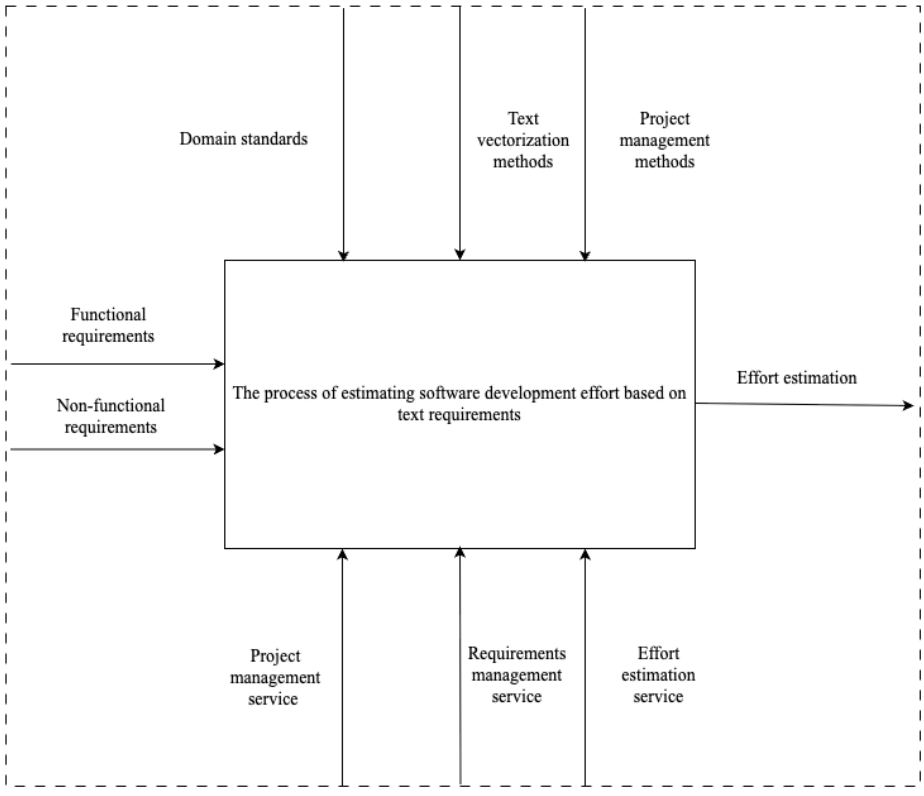

Fig. 2. General model for estimating the efforts required for software development

Let's examine the model's operation in more detail in Fig. 3. The input parameters, which represent the requirements, can be represented as a tuple X:

$$X = \{\{FR\}, \{NFR\}\},$$

where FR – functional requirements, specific functions that the software must perform; NFR – Non-functional requirements describe system characteristics that are not directly related to business functions; these may include security, productivity, and other requirements.

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

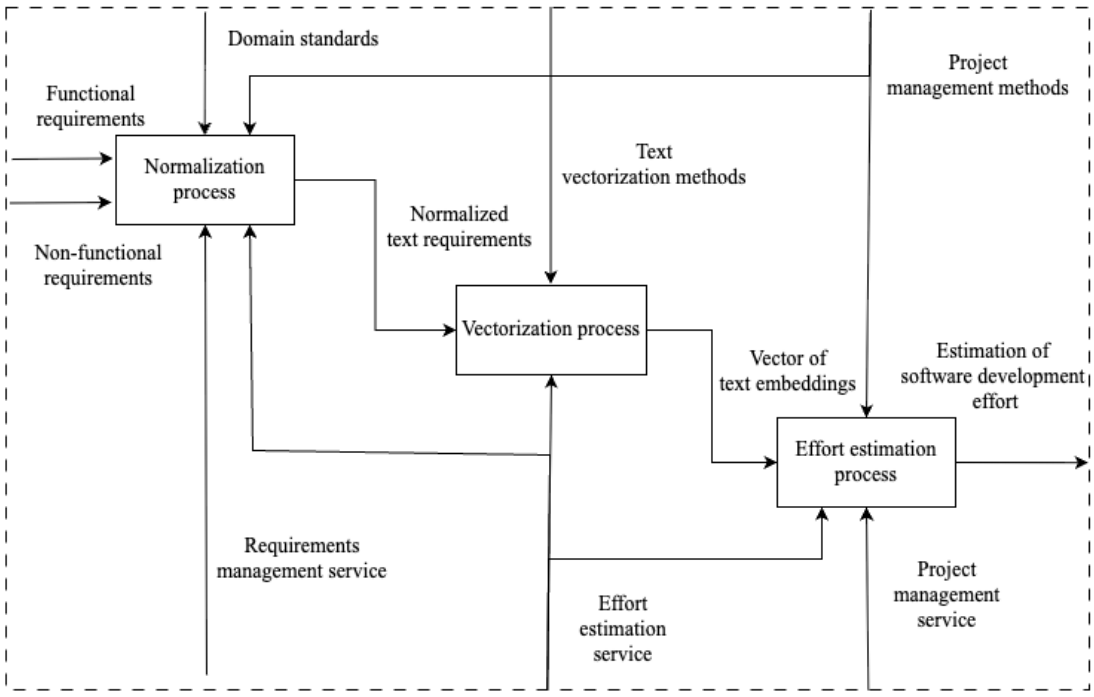

Fig. 3. Stages of estimating the efforts required for software development

In turn, the functional requirements can be presented as follows:

$$FR = \{\{TXT\},\{MLT\},\{FRM\}\},$$

where TXT – requirements presented in text form; MLT – visual requirements, which include mockups, interface prototypes, and so on, FRM – structured requirements presented in tabular form or as lists.

Non-functional requirements can be described as a tuple:

$$NFR = \{\{QAS\},\{GRP\}\},$$

where QAS – quality attribute scenarios, a common approach to the structured presentation of non-functional requirements; GRP – The requirements are presented in graphical form, including UML diagrams, block diagrams, and more

The first step in the model is the normalization process – it is necessary to prepare the data for use with vectorization methods. There are vectorization methods that can work with different types of input data, but it is necessary to obtain a single vector from all available requirements presented in various formats. Therefore, it is at this step that we convert them to a text type and combine them into a single continuous output text.

The normalization process can be represented as a formula:

$$x' = f(X),$$

where x' – textual description of requirements; X – the previously described set of requirements; f – normalization function.

The normalization results are used as input for the vectorization process, which converts textual information into a vector of text embeddings that describes the semantic meaning of the requirements.

This can be expressed as a formula:

$$V = g(x'),$$

where x' input text description of requirements; V – output numeric vector of text embeddings; g – vectorization function.

The quality of the results from this step has the greatest impact on the model's performance. After all, the semantic elements extracted from the text embeddings are the key components of the model's operation.

Once the features have been transformed into a vector, we proceed to the prediction process, which can be represented by the following formula:

$$y = h(V),$$

where V – the result of the previous stage, h – forecasting function, y – workload assessment in story points.

For forecasting, we will use machine learning models, specifically a multi-layer feedforward neural

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

network. It can be represented as a tuple and expressed by the following formulas:

$$\{N_{ij}, H_{ij}, I_{ij}, V\}\ , \quad (1)$$

$$N_{ij} = \sum_{k=1}^{n} Iw_{kij} H_{kj-1} + b_{ij}\ , \quad (2)$$

$$H_{ij} = f\left(N_{ij}\right)\ , \quad (3)$$

$$H_{i0} = V\ , \quad (4)$$

where N_{ij} – input values to the i-th neuron of the j-th layer, H_{ij} – output values to the i-th neuron of the j-th layer, w_{ki} – the weight coefficient of the connection between the k-th neuron of the previous layer and the i-th neuron of the j-th layer, V – the input layer of a neural network, consisting of text embeddings, n – the number of neurons in the previous layer.

To train the model, we will use the backpropagation algorithm. A key component of this method is calculating the model's error. It can be represented as follows:

$$E = C\left(y, \hat{y}\right)\ , \quad (5)$$

where E – is a forecasting error, y – output model value, $\hat{y}$ – is the expected value, C –loss function.

The first step in training the model is to use the input vector as the input layer of the neural network and perform a forward pass using equations (2) – (4).

Next, the error of the output layer is calculated using the formula:

$$\delta_i^J = \frac{dC}{dH_{iJ}} f'\left(N_{iJ}\right),$$

where δ – error, i – neuron number, J – output layer number, C – loss function, f' – the derivative of the activation function.

The error is then calculated for each layer (backpropagation) using the formula:

$$\delta_i^j = \left[\sum_{k=1}^{n} \delta_k^{j+1} \times w_{ki}^{j+1}\right] f'\left(N_{ij}\right),$$

where j – layer number $j = J-1\ldots1$, w – weight coefficients of neural connections.

Next, the weight coefficients are updated using the errors calculated at each layer of the neural network, using the formula:

$$w_t = w_{t-1} - o\left(\delta\right),$$

where w_t – updated weight coefficient value, w_{t-1} – the value of this coefficient in the previous iteration, o – an optimization function that calculates the bias based on the error calculated earlier.

These steps are performed at each iteration of the model training, allowing us to obtain the weights of the neural connections that best describe the relationship between the input and output parameters.

5.2. An experimental study to evaluate the accuracy of the proposed models on public datasets

We will conduct the experiment using data from the dataset [11], which consists of textual representations of requirements and includes a brief description of the user's history (title) and a detailed description (description). We will begin by normalizing the textual requirements. This process aims to reduce noise while preserving the semantic context of the requirement descriptions. In accordance with established practices [13], the cleaning stage involves removing special symbols (quotation marks, hashtags) and numerical values. Next, the entire text is converted to lowercase for unification and to facilitate tokenization. Additionally, stop words – tokens and symbols that do not carry meaningful information and typically arise from data exported from project management systems – are removed. Using a specialized stop list for this dataset helps filter out redundant or repetitive tokens, improving signal quality for subsequent vector representation and modeling. The list of special symbols and stop words to be removed during normalization is presented in Table 2.

Table 2. *List of special symbols and stop words removed during normalization*

0	1	2	3	4	5	6	7	8	9	0	!	“	#	%	&
‘	(	)	*	+	,	-	.	/	:	;	<	=	>	?	@
[	\	]	^	_	`	{	}	|	~	html		{html}			
		<p>		<pre>		<code>									

After text normalization, the total number of tokens was reduced by approximately 1.5 million and now stands at 2.7 million, as calculated using the text-embedding-3-large model (Fig. 4). At the same time, the average number of tokens per unit of text was reduced from 182 to 116.

After normalization, we proceed to convert the text features into vector form so that this transformed data

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

can be used in the next step. This process is performed separately for each of the models mentioned earlier, and each subsequent step is carried out in parallel for all generated text embeddings. The dimension of the output vectors depends on the models used.

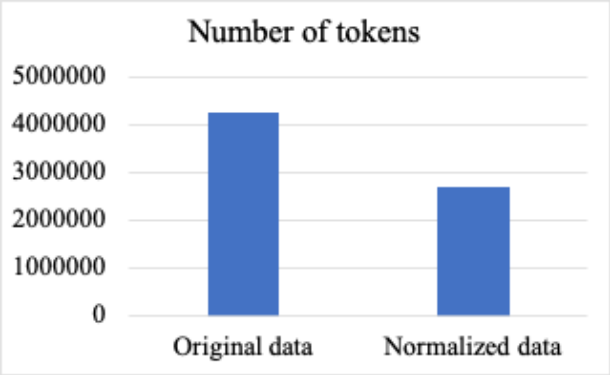

Fig. 4. Number of tokens before and after normalization

Next, we need to train the neural network responsible for forecasting. To do this, we must split the data into training and test sets, which will help evaluate the models' performance. For this, we use K-fold cross-validation. The number of folds was chosen dynamically depending on the project size to balance the ratio between the volume of training data and the reliability of validation: for projects with 1,000 or more samples, 10 folds were used; for projects with 500 to 999 samples, 5 folds; for 300–499 samples, 3 folds; and for smaller projects, 2 folds. This adaptive strategy ensures that each validation set is sufficiently representative without unduly reducing the training sample, thereby providing reliable and objective evaluations of productivity even with limited data. Results across all folds were averaged to form a comprehensive assessment of each model's predictive capabilities.

We will use a feedforward neural network for forecasting. The network consists of an input layer, the size of which corresponds to the dimension of the vectors resulting from the text transformation, followed by three dense hidden layers with the ReLU function as the activation function, and concludes with a linear output layer that generates the labor cost estimate. This architecture allows for a balance between complexity and generalization ability, ensuring effective modeling of nonlinear dependencies between text representations of user stories and the amount of labor required. Key hyperparameters were dynamically selected according to the characteristics of the dataset: the size of the hidden layers was set to half the dimension of the vectors, with a maximum limit of 512 neurons to maintain an optimal model size. The batch size was fixed at 32, and the Adam algorithm with standard learning rate settings was used for optimization due to its adaptability and success in training similar models.

To prevent overfitting and reduce unnecessary training time, an early stopping mechanism was implemented, which monitors the mean absolute error on the validation set with a limit of 10 epochs. That is, if the model's quality does not improve over 10 iterations, we will consider that the model has achieved the best possible result in this configuration. These settings reflect a practical yet effective approach, which was validated in [18] and serves as a reliable foundation for model creation in various software development projects.

Table 3. *Description of the layers of the direct expansion model*

Layer	Dimension
Input	N (size of text embeddings)
Hidden 1	MIN(N/2, 512)
Hidden 2	MIN(N/2, 512)
Hidden 3	MIN(N/2, 512)
Output	1

Separate models were built for each software project in the input data, since models for estimating effort based on text data are typically project-specific and generally do not generalize well across different projects. This approach accounts for the natural variability in project characteristics, requirements, and development practices that influence the relationship between text-based user stories and the corresponding effort. By training and evaluating models for each project separately, we aim to identify project-specific patterns and improve the accuracy and relevance of predictions within the unique context of each project.

As a result of training the prediction models, the following quality metrics were obtained. The predictions based on vector representations from the gemini-embedding-001 model showed the lowest mean absolute error. On average, the error is 2.7 story points (SP). Thus, when estimating the effort required to develop a feature using a 13SP model, the actual effort will range from 10.3 to 15.7.

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

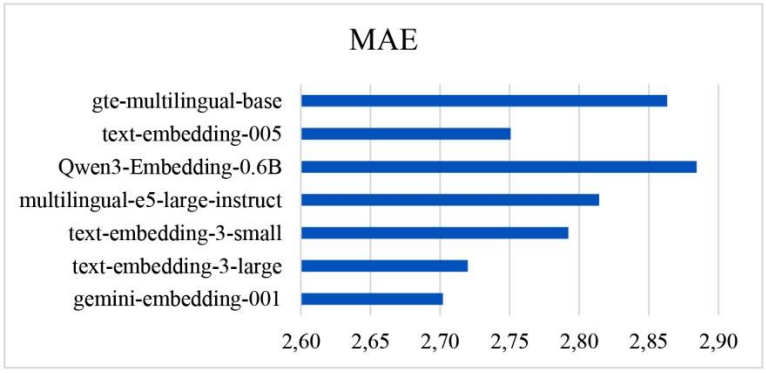

Fig. 5. Mean absolute error

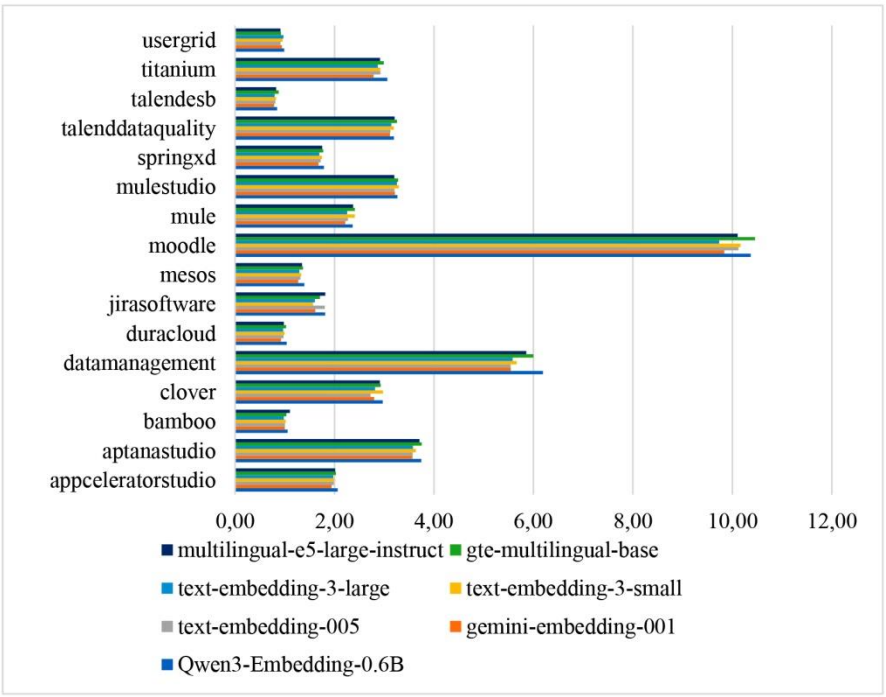

Fig. 6. Average absolute error by projects

The following evaluation is measured using the root mean square error, which is a more rigorous metric. Based on these error metrics, we can observe that gemini-embedding-001 does not perform the best (4.69), while text-embedding-3-large takes its place with a score of 4.66. This indicates that when evaluating more complex requirements, which will result in a high SP score, the error of the second model will be lower.

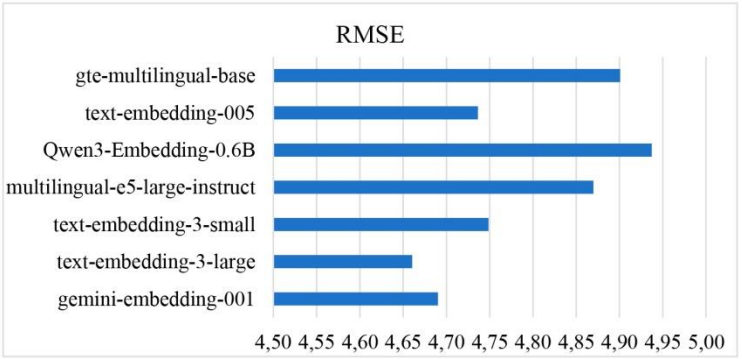

Fig. 7. Root of the root mean square deviation

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

Last but not least, let's examine the average relative error. Based on this metric, gemini-embedding-001 once again takes first place. It shows us that, on average, the estimation error is 0.64, or 64%. This relatively high figure may be due to the fact that the test data contains a large number of requirements with relatively low expert estimates of effort. About 70% of the records in the data used have an estimate below 8 SP.

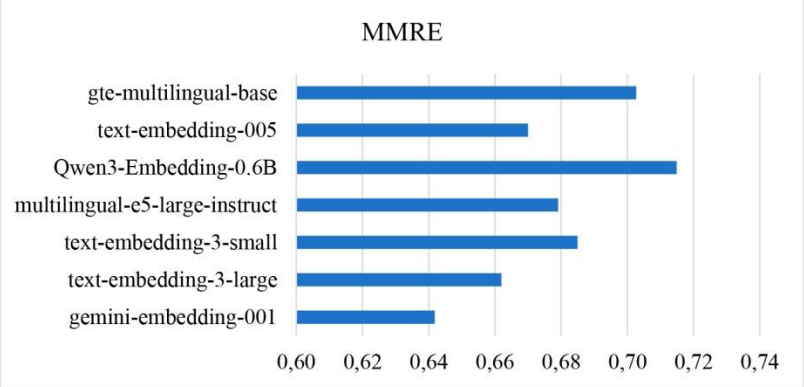

Fig. 8. Mean relative error

6. Discussion of the results

The experiment revealed that forecasting accuracy depends significantly on the characteristics of the dataset and the style of user story documentation. For example, the Talend ESB project showed the lowest error rates (MAE ranging from 0.79 to 0.88), while Moodle had the highest errors (MAE ranging from 9.73 to 10.37).

Based on averaged metrics across all projects for each embedding model, gemini-embedding-001 and text-embedding-3-large – the most advanced models from Google and OpenAI, respectively – performed best. These models generate the longest embedding vectors among those considered, suggesting that higher dimensionality is better able to capture the semantic context of software development requirements, which in turn directly impacts prediction accuracy.

When comparing results by MAE, the results are quite closely clustered – ranging from 2.70 (best, Gemini) to 2.88 (worst, qwen-06). Overall, across all error metrics, the qwen-06 and gte-multilingual models consistently ranked last. Given that they belong to models with medium and short embedding vector lengths, one might assume that this is the reason for their poor performance. However, the use of text-embedding-005 contradicts this. While falling within the same range of vector dimensions, it yields results similar to those of models with longer vectors. This indicates a significant influence of the training datasets, the training method itself, and the architecture of these models. Next, we should compare the obtained results with state-of-the-art models [11, 13, 25, 26]. Since most studies evaluated the quality of the built models using the MAE metric, we will use this metric as well.

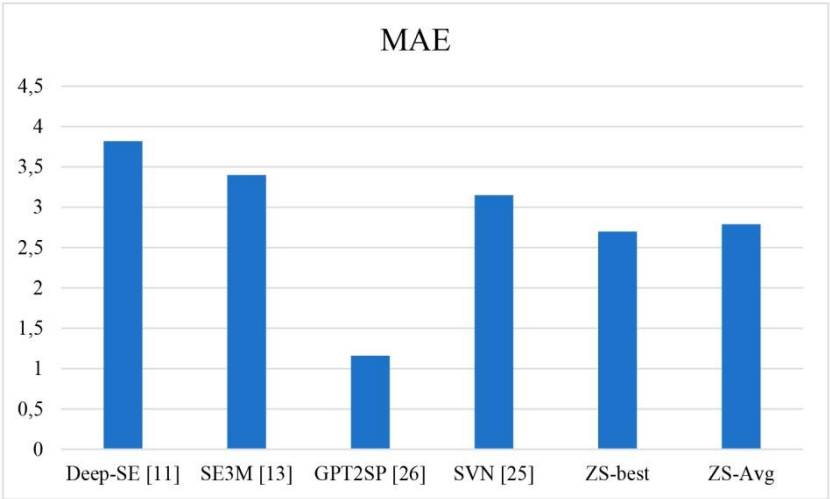

Fig. 9. MAE metrics for the developed model (ZS) and existing state-of-the-art models

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

The diagram shows two metrics obtained from the results of applying the developed model, namely the best metric (ZS-best) and the average metric (ZS). Overall, it can be noted that the use of modern zero-shot models in combination with feedforward neural networks yields good results compared to the metrics in existing studies [11, 13, 25], which use pre-trained models with fine-tuning on domain-specific data. In turn, GPT2SP [26] remains the leader in evaluation accuracy, and it is worth noting that such accuracy is achieved by building upon and fine-tuning the existing GPT2 model, which generally requires comparatively greater effort than using ready-made, publicly available, or proprietary zero-shot models.

7. Conclusions

1. A model for effort estimations based on the vectorization of software requirements has been developed. This model helps avoid the additional use of resources – in this case, experts – to evaluate user stories during planning at various stages of the software development project lifecycle. This can significantly help project managers plan time and labor resources.

2. An experimental study was conducted on the impact of using different text vectorization language models on the overall quality of labor cost prediction models based on text requirements. Considering most quality metrics, the gemini-embedding-001 model with a text embedding vector of 3072 showed the best results (MAE – 2.70, MMRE – 0.64). At the same time, the text-embedding-005 model with a significantly smaller vector (769) showed similar performance and is the best among models with small and medium-sized vectors.

3. For further development of the model for estimating effort based on software requirements, it is worth considering the possibility of adding a preprocessing step for different formats of requirement specifications. This should help identify structural elements that best reflect the complexity of implementing user stories in the generated embedding vectors. Another approach is to incorporate project-specific information, such as long-term or short-term goals, into the functionality requirements when generating text embeddings, which could improve the quality of the vectors for use in forecasting models.

Conflict of Interest

The authors declare that they have no conflicts of interest, including financial, personal, authorship, or any other conflicts that could influence the study or the results published in this article.

Funding

The study was conducted without financial support.

Data Availability

Data will be provided upon reasonable request.

Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence technologies to write this paper.

References

1. Project Management Institute (2017), *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 6th edn. Newtown Square, PA: Project Management Institute.
2. Project Management Institute Sweden (2024), *Artificial Intelligence and Project Management: A Global Chapter-Led Survey 2024*, Stockholm: Project Management Institute.
3. Taboada, I., Daneshpajouh, A., Toledo, N., de Vass, T. (2023), "Artificial Intelligence Enabled Project Management: A Systematic Literature Review", *Applied Sciences*, Vol. 13(8), p. 5014. DOI: https://doi.org/10.3390/app13085014
4. Martovytskyi, V., Argunov, V., Ruban, I., Romanenkov, Y. (2023), "Developing a risk management approach based on reinforcement training in forming an investment portfolio", *Eastern-European Journal of Enterprise Technologies*, Vol. 2(3(122)), pp. 106–116. DOI: https://doi.org/10.15587/1729-4061.2023.277997
5. Dam, H. K., Tran, T., Grundy, J., Ghose, A., Kamei, Y. (2019), "Towards Effective AI-Powered Agile Project Management", *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Montreal, Canada, 25-31 May. IEEE, pp. 41–44. DOI: https://doi.org/10.48550/arXiv.1812.10578

6. Serdechnyi, V., Barkovska, O., Kovalenko, A., Havrashenko, A., Martovytskyi, V. (2025), "Research on machine learning methods for detecting objects in difficult shooting conditions", *Radioelectronic and Computer Systems*, Vol. 2, pp. 64–77. DOI: https://doi.org/10.32620/reks.2025.2.04
7. Azzeh, M. (2011), "Adjusted case-based software effort estimation using bees optimization algorithm", *Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 315–324. DOI: https://doi.org/10.1007/978-3-642-23863-5_32
8. Bardsiri, V. K., Jawawi, D. N. A., Bardsiri, A. K., Khatibi, E. (2013), "LMES: A localized multi-estimator model to estimate software development effort", *Engineering Applications of Artificial Intelligence*, Vol. 26(10), pp. 2624–2640. DOI: https://doi.org/10.1016/j.engappai.2013.08.005
9. Jorgensen, M., Shepperd, M. (2006), "A systematic review of software development cost estimation studies", *IEEE Transactions on Software Engineering*, Vol. 33(1), pp. 33–53. DOI: https://doi.org/10.1109/TSE.2007.256943
10. Alsaadi, B., Saeedi, K. (2022), "Data-driven effort estimation techniques of agile user stories: a systematic literature review", *Artificial Intelligence Review*, Vol. *55*(7), pp. 5485–5516. DOI: https://doi.org/10.1007/s10462-021-10132-x
11. Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., Menzies, T. (2018), "A deep learning model for estimating story points", *IEEE Transactions on Software Engineering*, Vol. 45(7), pp. 637–656. DOI: https://doi.org/10.48550/arXiv.1609.00489
12. Usman, M., Mendes, E., Weidt, F., Britto, R. (2014), "Effort estimation in agile software development: a systematic literature review", *Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE '14)*, Turin, Italy, 18 September, New York: ACM, pp. 82–91. DOI: https://doi.org/10.1145/2639490.2639503
13. Fávero, E., Casanova, D., Pimentel, A. (2020), "SE3M: A Model for Software Effort Estimation Using Pre-trained Embedding Models", *arXiv*. DOI: https://doi.org/10.48550/arXiv.2006.16831
14. Moharreri, K., Sapre, A. V., Ramanathan, J., Ramnath, R. (2016), "Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments", *Proceedings - International Computer Software and Applications Conference*, 2, pp. 135–140. DOI: https://doi.org/10.1109/COMPSAC.2016.85
15. Zhang, C., Tong, S., Mo, W., Zhou, Y., Xia, Y., Shen, B. (2016), "ESSE: an early software size estimation method based on auto-extracted requirements features", *Internetware '16: Proceedings of the 8th Asia-Pacific Symposium on Internetware*, Beijing, China, 18 September, New York: Association for Computing Machinery, pp. 112–115. DOI: https://doi.org/10.1145/2993717.2993733
16. Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013), "Efficient estimation of word representations in vector space", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.1301.3781
17. Floridi, L., Chiriatti, M. (2020), "GPT-3: Its Nature, Scope, Limits, and Consequences", *Minds and Machines*, 30(4), pp. 681–694. DOI: https://doi.org/10.1007/s11023-020-09548-1
18. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019), "BERT: Pre-training of deep bidirectional transformers for language understanding", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, 2–7 June, Stroudsburg, PA: Association for Computational Linguistics, pp. 4171–4186. DOI: https://doi.org/10.18653/v1/N19-1423
19. Jadhav, A., Kaur, M., Akter, F. (2022), "Evolution of software development effort and cost estimation techniques: five decades study using automated text mining approach", *Mathematical Problems in Engineering*, Vol. 1, 5782587. DOI: https://doi.org/10.1155/2022/5782587
20. Rossi, B. B., Fontoura, L. M. (2025), "AI-Based Approaches for Software Tasks Effort Estimation: A Systematic Review of Methods and Trends", *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS)*, Vol. 2, pp. 144–151. DOI: https://doi.org/10.5220/0013218200003929
21. Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., Ali, M. (2022), "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation", *Software: Practice and experience*, Vol. 52(1), pp. 39–65. DOI: https://doi.org/10.1002/spe.3009
22. Tenekeci, S., Ünlü, H., Dikenelli, E., Selçuk, U., Soylu, G.K., Demirörs, O. (2024), "Predicting software size and effort from code using natural language processing", in *Joint Proceedings of the 33rd International Workshop on Software Measurement and the 18th International Conference on Software Process and Product Measurement (IWSM-MENSURA 2024)*, Montréal, Canada, September, Aachen: CEUR-WS, Vol. 3852
23. Sharma, A., Chaudhary, N. (2022), "Analysis of software effort estimation based on story point and lines of code using machine learning", *International Journal of Computing and Digital Systems*, Vol. 12(1). DOI: https://journal.uob.edu.bh/10.12785/ijcds/1201012
24. Catak, T., Durdu, P.O., Omurca, S.I. (2024), "Enhancing agile effort estimation: An nlp approach for software requirements analysis", *2024 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, pp. 1–8. DOI: https://doi.org/10.1109/HORA61326.2024.10550870

25. Atoum, I., Otoom, A. A. (2024), "Enhancing software effort estimation with pre-trained word embeddings: A small-dataset solution for accurate story point prediction", *Electronics*, Vol. 13(23), p. 4843. DOI: https://doi.org/10.3390/electronics13234843
26. Fu, M., Tantithamthavorn, C. (2022), "GPT2SP: A transformer-based agile story point estimation approach", *IEEE Transactions on Software Engineering*, Vol. 49(2), pp. 611–625. DOI: https://doi.org/10.1109/TSE.2022.3158252
27. Enevoldsen, K. et al. (2025), "MMTEB: Massive Multilingual Text Embedding Benchmark", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2502.13595
28. Shahinmoghadam, M., Motamedi, A. (2025), "Benchmarking pre-trained text embedding models in aligning built asset information", *Scientific Reports*, Vol. 15(1), p. 23866. DOI: https://doi.org/10.1038/s41598-025-09052-5
29. Lee, J., Chen, F., Dua, S., Cer, D., Shanbhogue, M., Naim, I., Duerig, T. (2025), "Gemini embedding: Generalizable embeddings from gemini", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2503.07891
30. OpenAI (2024), *OpenAI documentation*, available at: https://platform.openai.com/docs/ (last accessed 12.05.2026)
31. Zhang, X., Thakur, N., Ogundepo, O., Kamalloo, E., Alfonso-Hermelo, D., Li, X., Lin, J. (2022), "Making a MIRACL: Multilingual Information Retrieval Across a Continuum of Languages", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2210.09984
32. Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Zhou, J. (2025), "Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2506.05176
33. Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F. (2024), "Multilingual e5 text embeddings: A technical report", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2402.05672
34. Zhang, X., Zhang, Y., Long, D., Xie, W., Dai, Z., Tang, J., Lin, H., Yang, B., Xie, P., Huang, F., Zhang, M., Li, W., Zhang, M. (2024), "mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval", *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Miami, Florida: Association for Computational Linguistics, pp. 1198–1213. DOI: https://doi.org/10.18653/v1/2024.emnlp-industry.103
35. Lee, J., Dai, Z., Ren, X., Chen, B., Cer, D., Cole, J. R., Naim, I. (2024), "Gecko: Versatile text embeddings distilled from large language models", *arXiv preprint*. DOI: https://doi.org/10.48550/arXiv.2403.20327

Received (Надійшла) 07.04.2026
Accepted for publication (Прийнята до друку) 15.05.2026
Publication date (Дата публікації) 29.05.2026

Відомості про авторів / About the Authors

Кушнір Іван Сергійович –Харківський національний університет радіоелектроніки, аспірант кафедри електронних обчислювальних машин, Харків, Україна;
Ivan Kushnir – Kharkiv National University of Radio Electronics, Postgraduate Student, Department of Electronic Computers, Kharkiv, Ukraine;
e-mail: ivan.kushnir@nure.com
ORCID ID: https://orcid.org/0009-0004-5055-0331

Кирій Валентина Василівна – кандидат економічних наук, доцент, Харківський національний університет радіоелектроніки, завідувач відділу аспірантури та докторантури, Харків, Україна;
Valentyna Kyrii – PhD (Economic Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Head of the Department of Postgraduate Studies and Doctoral Studies, Kharkiv, Ukraine;
e-mail: valentyna.kyriy@nure.ua
ORCID ID: https://orcid.org/0000-0002-2537-264X

МОДЕЛЬ ОЦІНЮВАННЯ ТРУДОВИТРАТ НА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ПОПЕРЕДНЬО НАВЧЕНИХ ZERO-SHOT-МОДЕЛЕЙ

Управління проєктами створення програмного забезпечення (ПЗ) передбачає планування, виконання та контроль робіт для забезпечення дотримання бюджету й строків. **Предметом дослідження** є оцінювання трудовитрат, що є одним

ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

із найвагоміших завдань, оскільки дає змогу керівникам мінімізувати ризики й оптимально розподілити ресурси. За появи штучного інтелекту (ШІ) й продовження цифрової трансформації ключового значення набуває застосування методів оброблення природної мови (NLP) і великих мовних моделей (LLM). Ці інструменти здатні аналізувати текстові вимоги й користувацькі історії для прогнозування необхідних зусиль. Вони допомагають семантично структурувати складні текстові вимоги й покращувати точність оцінювання. Використання zero-shot-моделей не потребує донавчання на цільових даних, що економить ресурси й забезпечує високу адаптивність. **Мета дослідження** – розроблення моделі для оцінювання трудовитрат на основі вимог до ПЗ, поданих у різних формах. Це досягається за допомогою таких завдань: розроблення моделі оцінювання трудовитрат на створення ПЗ на основі сучасних попередньо навчених zero-shot-моделей; експериментальне дослідження процесу оцінювання точності запропонованих моделей. **Результати.** У роботі застосовано публічний набір даних і моделі ШІ, що різняться архітектурою, методами навчання й ліцензійними умовами. Моделі оцінювалися за метриками середньої абсолютної похибки, середньої відносної похибки й кореня середньоквадратичного відхилення. Побудовано модель оцінювання трудовитрат на розроблення ПЗ, яка в процесі експерименту продемонструвала, що точність прогнозування трудовитрат значною мірою залежить від характеру набору даних і стилю формулювання користувацьких історій. Порівняно з узагальненими моделями попередніх досліджень тренування проєктно-специфічних моделей значно підвищує точність прогнозів. Запропонована модель оцінювання трудовитрат на основі zero-shot-векторизації ефективно автоматизує оцінювання користувацьких історій, полегшуючи планування ресурсів на різних етапах життєвого циклу створення ПЗ. Досягнуті результати відкривають перспективи для гнучкого й масштабованого застосування NLP на основі великих мовних моделей у сфері управління проєктами розроблення програмного забезпечення.

Ключові слова: управління проєктами; розроблення програмного забезпечення; оцінювання трудовитрат; штучний інтелект; великі мовні моделі; векторизація тексту.

Бібліографічні описи / Bibliographic descriptions

Кушнір І. С., Кирій В. В. Модель оцінювання трудовитрат на розроблення програмного забезпечення на основі попередньо навчених zero-shot-моделей. *Автоматизовані системи управління та прилади автоматики*. 2026. № 2 (189). С. 196–210. DOI: https://doi.org/10.30837/0135-1710.2026.189.196

Kushnir, I., Kyrii, V. (2026), " Software development effort estimation model based on pre-trained zero-shot models", *Management Information System and Devices*, No. 2 (189), P. 196–210. DOI: https://doi.org/10.30837/0135-1710.2026.189.196