applied system innovation

MDPI

Article

Preliminary Estimation for Software Development Projects Empowered with a Method of Recommending Optimal Duration and Team Composition

Vasyl Teslyuk *, Anatoliy Batyuk and Volodymyr Voityshyn

Department of Automated Control Systems, Computer Science and Information Technologies Institute, Lviv Polytechnic National University, 79000 Lviv, Ukraine; anatolii.y.batiuk@lpnu.ua (A.B.); volodymyr.v.voityshyn@lpnu.ua (V.V.)

* Correspondence: vasyl.m.teslyuk@lpnu.ua

Abstract: In the early software development stages, the aim of estimation is to obtain a rough understanding of the timeline and resources required to implement a potential project. The current study is devoted to a method of preliminary estimation applicable at the beginning of the software development life cycle when the level of uncertainty is high. The authors' concepts of the estimation life cycle, the estimable items breakdown structure, and a system of working-time balance equations in conjunction with an agile-fashioned sizing approach are used. To minimize the experts' working time spent on preliminary estimation, the authors applied a decision support procedure based on integer programming and the analytic hierarchy process. The method's outcomes are not definitive enough to make commitments; instead, they are supposed to be used for communication with project stakeholders or as inputs for the subsequent estimation stages. For practical usage of the preliminary estimation method, a semistructured business process is proposed.

Keywords: software development effort estimation; rough order of magnitude; ballpark figures; semistructured business process

Citation: Teslyuk, V.; Batyuk, A.; Voityshyn, V. Preliminary Estimation for Software Development Projects Empowered with a Method of Recommending Optimal Duration and Team Composition. *Appl. Syst. Innov.* **2024**, *7*, 34. https://doi.org/10.3390/asi7030034

Academic Editor: Luís Oliveira

Received: 16 February 2024
Revised: 9 April 2024
Accepted: 18 April 2024
Published: 23 April 2024

1. Introduction

From the business standpoint, predictability is an essential aspect of software development. In other words, before starting a project, it is important to understand how many resources and how much time the implementation will require. This is why estimation is an integral part of the software development life cycle. Following the cone-of-uncertainty concept [1], estimation is envisioned as a multistage process starting from an early phase of project ideation and finishing along with the implementation completion. At each project life cycle stage, estimation has certain goals, requires specific methods, and produces different outcomes.

Most well-known and practically proven estimation methods were developed in the second half of the 20th century. As a result, the application of those methods to modern software development projects causes difficulties or even is hardly possible. Such methods as COCOMO II [2] are quite complex and require special training. Another widely used method, PERT [3,4], is relatively easy but does not take into account the agile nature of modern software development projects. Agile-oriented methods like planning poker, T-shirt sizing, affinity grouping, and their variations are rather applicable by agile teams at the implementation stage [5–8]. A common downside of the existing methods (except COCOMO II) is that they do not cover the whole software development life cycle starting from project ideation and ending with production, including maintenance and support.

To fill in this gap, the authors propose the concept of the estimation life cycle (ELC), tailoring estimation activities to the software development life cycle (SDLC) stages. According to the ELC, early project stages are accompanied by introductory, preliminary, and

intermediate [9] estimations (listed in the order of increasing levels of details, reliability, and accuracy). The commonality of these estimations is that their outputs are not definitive enough to take on obligations. Instead, the following step—precise estimation—aims at providing outcomes that can be used for making commitments. Precise estimation takes place before the implementation start, when the analysis and design are completed. At the implementation stage, the estimates are compared with actual efforts, and a project team is responsible for keeping up-to-date the remaining estimates. After completing implementation, the ELC prescribes finalizing feedback on estimates.

The current study is devoted to a method of preliminary estimation. The method's objective is to roughly understand the resources and time required to implement a project, minimizing the efforts spent on the estimation itself. Usually, this type of estimation is applicable in the initial steps of SDLC. The main outputs of the preliminary estimation are project scope (represented as an estimable-item breakdown structure, defined below), team composition, and project duration. Obviously, due to the quite high level of uncertainty, these outcomes are not supposed to be used for making commitments; instead, they will be elaborated on in the following stages: the intermediate and precise estimations [9,10].

The method of preliminary estimation proposed in this study is based on a simplified version of a system of working-time balance equations [9,10]. This applied simplification (in comparison with the intermediate estimation) aims, first of all, at the minimization of efforts spent on the estimation, intentionally sacrificing its accuracy. Additionally, the method is empowered with a decision support procedure based on a multiobjective optimization with the purpose of providing the involved experts with several estimate alternatives. For practical usage, the method of preliminary estimation involves a semistructured knowledge-intensive business process [11,12], predefining steps to be performed and, at the same time, leaving enough space for flexibility and creativity.

The rest of this paper is structured as follows: the research background is given in Section 2; a literature review is in Section 3; the main results, including the semistructured business process, the concept of an estimable-item breakdown structure, the sizing approach, the system of working-time balance equations, the approach to compose a project team, the project duration estimation, and the multiobjective decision support procedure, are presented in Section 3; the challenges of the project scope decomposition, development specializations, development working time coefficient, and validity of the proposed method are discussed in Section 4; conclusions are in Section 5.

2. Background

The current study is a logical continuation of the authors' past works—the preliminary estimation method is based on their estimation framework [9,10]. This section is devoted to the basic terms and concepts introduced previously, in particular, the concept of the estimation life cycle, the structure of software developer working time, and the "normalization" approach to measure development efforts.

2.1. Concept of Estimation Life Cycle

A common mistake is to consider estimation as a one-time activity that takes place before the project start. As a rule, such a mistake leads to a low-quality estimate that, in turn, causes the unpredictability of project resources and deadlines. The key to avoiding this issue is to organize the estimation as a multistep process taking place at each SDLC stage. Of course, depending on the project life cycle stage, the estimation targets different goals, requires various methods, and satisfies certain accuracies. In the current paper, the estimate accuracy means closeness of the estimated values to the corresponding actual value requirements; obviously, the accuracy defined in this way can be measured only after receiving the actual values, i.e., during the project implementation or even after its completion. In Figure 1, the concept of the estimation life cycle (ELC) proposed by the authors is visualized. Being based on the cone-of-uncertainty idea [1], the concept represents estimation as an integral part of SDLC.

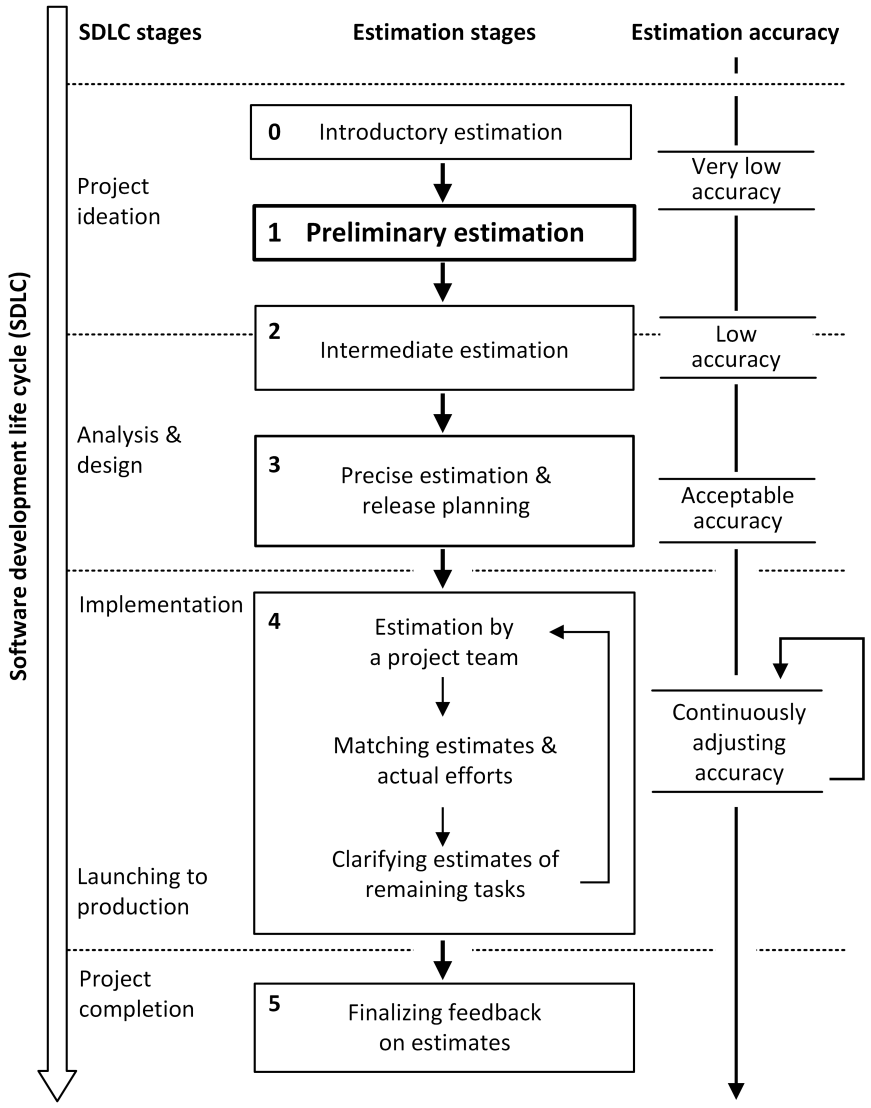

Figure 1. Concept of estimation life cycle.

The estimation starts from the so-called ideation phase (which is a very early stage of a project), when the level of uncertainty in the understanding of different aspects of the project is the highest. As a rule, the estimates at this stage are represented as wide ranges that rather express a rough order of the required resources and time. In the project ideation stage, introductory and preliminary estimates are undertaken. The main difference between these successive steps is that introductory estimates are mostly based on similarity with past projects and do not need a project scope analysis. In its turn, the preliminary estimation requires the project scope analysis. Definitely, such estimate types are rather informative and not recommended for commitments (e.g., when signing a contract). The estimation goal at the project ideation stage is to roughly understand the project scope and resources required for the project's implementation while minimizing the efforts to prepare the estimate itself.

As the level of uncertainty decreases, it becomes possible to provide more accurate estimates using the outcomes from the previous stages as inputs. Higher accuracy of intermediate estimates is mostly achieved through the elaboration of the project scope as well as distinguishing the project implementation phases. In order to reduce the efforts spent on the estimation itself, the corresponding methods do not consider dependencies among the project tasks. Although their definitiveness is higher, intermediate estimates are still not reliable enough to be used for commitments. Further information about intermediate estimation can be found in [9,10].

After a certain amount of analysis and design, providing a precise estimate and a project release plan becomes possible. Unlike in the previous stages, the precise estimate's accuracy is acceptable to use to make commitments. In this regard, precise estimation is similar to what is called the definitive estimate [13]. Since the efforts required to prepare such an estimate are greater in comparison to the previous stages, it is worth undertaking it in the case of a high probability of the project implementation's start.

In the project implementation phase, it is important to constantly monitor progress, comparing the actual efforts with the estimate and reacting proactively to undesirable deviations from the release plan. To avoid missing deadlines, regular re-estimation of the remaining project tasks is necessary. As this part of the project life cycle is likely to follow an agile methodology, the agile team is the main source of feedback on the estimate, and the team is responsible for adjusting the estimate according to the actual project state.

After project completion, it is quite useful to finalize the feedback on the estimates comparing them with the actual working time as well as analyzing lessons learned. The collected feedback is valuable for future project estimation and planning.

The main idea of splitting estimation into several stages is that the estimate accuracy becomes higher, while the level of uncertainty decreases. The stages are consequent—outcomes of the previous stage are inputs for the next one. Therefore, performing estimation successively by following the ELC ensures higher quality and less total effort in comparison with the case where, for example, the previous stages are skipped and only the precise estimation is performed before the project implementation start.

2.2. Structure of Software Developer Working Time

One of the core concepts of the authors' estimation framework [9,10] is the structure of software developer working time. The *project working time* (PWT), W, means total working time spent by a software engineer working on a project. PWT consists of the following three parts: M, the working time spent on project scope implementation; G, the working time spent on general project activities such as daily meetings, sprint plannings, etc.; N, nonworking time including idle time as well as days off, sick leaves, vacations.

In turn, the project scope implementation time, M, is split into two parts: *development working time* (DWT), D, and supplementary development activities, A. D is the time that a software engineer spends on coding activities. And A is the time spent on supplementary activities such as writing unit tests, defect fixing, team collaboration, etc.

The nonworking time, N, also consists of two parts. The first is idle time, I, when a software engineer does not do actual work due to, for example, lack of project tasks. And leave time, O, includes planned vacations, unplanned days off, and sick leave.

Hence, the structure of software developer working time is expressed as follows:

$$W = M + G + N = (D + A) + G + (I + O). \quad (1)$$

The preliminary estimation proposed in the current paper is based on a simplified version of the structure of software developer working time including only the PWD and DWT, omitting the other variables (Equation (8) in (Section 4).

2.3. Measurement of Development Efforts

The existing approaches to measurement (or sizing) of development efforts can be grouped into the three main categories [14,15]. The most commonly used category is based on the idea of function points [16]. Such units as object points [2], use-case points [17], story points [5–8], etc., are derived from the concept of function points. Another well-known development effort measurement unit comprises logical lines of code (the so-called SLOC metric) used by COCOMO [18] and COCOMO II [2]. And the third category is working time-based (e.g., used in PERT [3,4]), expressing development efforts in man-hours, man-days, man-months, etc. The main downside of the approaches from the first and second categories is the difficulty in converting them to time-based units, which usually requires some historical data. To eliminate such an issue, the authors' estimation framework

operates with time-based units to measure development efforts. The theoretical basis of this is given in the current section below.

An *"average" software developer* (ASD) is a software developer who possesses a high enough competency level to act as a team player and implement project tasks of acceptable quality without permanent supervision by more experienced colleagues; and, at the same time, such a developer has room for improvement in terms of efficiency and complexity of the work performed. A middle software engineer is the position which is closest to the ASD. The development effort "normalization" approach follows from the concept of the ASD.

The *normalized development working time* (NDWT), U, is the DWT spent by an ASD implementing certain project tasks who is 100% involved (i.e., 8 h per working day) in a project. If D is the DWT spent by a developer who is different from the ASD, then the following expression takes place:

$$U = \rho D, \tag{2}$$

where ρ is the *productivity coefficient* (PC). Assuming that the development productivity is influenced by the competency level and the project involvement,

$$\rho = \alpha\eta, \tag{3}$$

where α is the *competency-level productivity coefficient* (CLPC) and η is the *involvement productivity coefficient* (IPC). For an ASD, $\alpha = 1$; $0 < \alpha < 1$ for a developer of a competency level lower than the ASD, and $\alpha > 1$ for a competency level higher than the ASD. In the case of 100% of project involvement, $\eta = 1$; for partial involvement or overtime, $0 < \eta < 1$. The IPC and CLPC parameters are used in Section 4 to define the project team composition.

The *normalized development capacity* (NDC), C, is maximum possible value of NDWT that can be spent on the project tasks implementation:

$$C = \max_{U \in \mathbb{U}} U = \rho \max_{D \in \mathbb{D}} D, \tag{4}$$

where $\mathbb{U}$ and $\mathbb{D}$ are the sets of all possible values of NDWT and DWT respectively. In particular, the maximization in (4) is aimed at minimizing idle time I and leaves time O. For the preliminary estimation, NDC is defined as (12) and (13).

The *normalized development estimate* (NDE), E, of a certain portion of the project scope is a forecasted DWT assuming that the work is performed by an ASD. Usage of the NDE for preliminary estimation is explained in Section 4, in particular, for sizing of estimable items.

In the authors' estimation framework, the NDC and NDE occupy one of the key places, being "common denominators" for measuring development efforts. Importantly, these parameters determine project duration (Inequalities (14) and (15) in Section 4). It is worth noting that the described "normalization" approach has been expanded to include the project involvement definition, introducing a new term—*normalized development full-time equivalent* (Equation (7) in Section 4).

3. Literature Review

The preliminary estimation represented in the current paper is close to such well-known terms as "rough order of magnitude" (ROM) [13,19] and more informal ones—"guesstimate" or "ballpark figures"—that are usually used to express an approximation of future project efforts, schedule, and budget. The method proposed by the authors is supposed to be applied under the following circumstances: (a) an early project life cycle stage when the level of uncertainty is high; (b) limited time to prepare an estimate; (c) acceptability of low estimate accuracy; (d) an estimate that is not recommended for making commitments; (e) unknown dependencies among project tasks. In the current section, a thorough overview is provided of the existing methods that can be applied under the conditions mentioned above.

An estimation by analogy [20–23] allows one to understand the approximate efforts and duration based on similar past projects. Such methods can either utilize formal

approaches using similarity metrics [24] and a database with historical projects or rely more on experts' opinion. A significant drawback of those methods is the lack of project scope analysis; the main advantage is the speed of estimation. In the case of an existing solid database of past projects, analogy-based methods can be empowered with machine learning and AI techniques [23,25,26].

One of the most mature of the existing estimation methodologies—COCOMO II [2]—supports the concept of tailoring estimation models to the project life cycle stages. In the early prototyping phase, the Application Composition model is applicable. This model is based on the idea of measuring the size of software in so-called object points (which, in turn, derives from the concept of a function point [16]). The application-points approach involves identifying objects of a software product (e.g., screens, reports, database tables, etc.) and defining their complexity levels, which are used to calculate application points that are translated to the efforts and schedule. In the subsequent early design phase, the COCOMO II Early Design model is suggested. The usage of this model requires trained estimation experts, availability of historical data, and calibration of the model parameters. Its challenging part is also the sizing, which requires the calculation of so-called logical lines of source code (i.e., the SLOC metric) or function points. Whilst both Application Composition and Early Design models are applicable in the initial SDLC stages, the second one seems to be more difficult for usage in practice.

In this regard, it is also worth mentioning the approach based on use-case points [17,27–31]. An advantage of such a method is the usage of UML use-case models to analyze project functionality. The strong side of this method is its relatively simple visual notation representing both the users (actors) and functionality (use cases) of a software system (which is quite handy in the early project life cycle stages). However, the calculation formulas proposed in the original publication [17] most probably will require some adjustments before application to a specific project.

Widespread use of agile methodologies has led to the creation of agile-fashion estimation techniques. Usually, such techniques are used by project teams during the project implementation phase. However, some of their elements can also be applied in the early project life cycle phases before the development start, even before forming an agile team. One such method is affinity grouping [5–8]: tasks are grouped into clusters by their similarity (e.g., estimated efforts or complexity), and then the clustered tasks are assigned with estimates. Another well-known method is T-shirt sizing [5–8], where tasks are attributed to one of the predefined categories that usually correspond to the sizes of T-shirts: XS, S, M, L, XL, etc. Both methods can be used either by teams or by individual experts for sizing of estimable items in the preliminary estimation stage.

Another quite popular method is the so-called three-point estimation, where experts provide three values: optimistic, pessimistic, and most likely. Then, the final estimate is calculated as a weighted average of these three points. Such an approach is inherited from the PERT methodology [3,4]. To apply this method to preliminary estimation, it is required to prepare a work breakdown structure that, in turn, is estimated by experts.

Despite the considerable number of existing methods, none of them fully covers the requirements of the preliminary estimation provided at the beginning of the section. Furthermore, the majority of the methods suffer from the absence of a holistic vision of estimation as a multistep process inseparable from SDLC.

4. Results

4.1. Preliminary Estimation as a Semistructured Business Process

Before starting a discussion of the preliminary estimation method in detail, it is worth analyzing its usage in practice. Since preliminary estimation heavily relies on the knowledge, experience, and creativity of the involved experts, it makes sense to keep it as a semistructured business process [11,12], leaving some level of freedom for the participants. In other words, such a process involves certain steps; however, there are no strict recommendations on the steps' order (i.e., some of the steps can be interchanged, and

some of them can be repeated several times). The preliminary estimation semistructured process is represented in Figure 2.

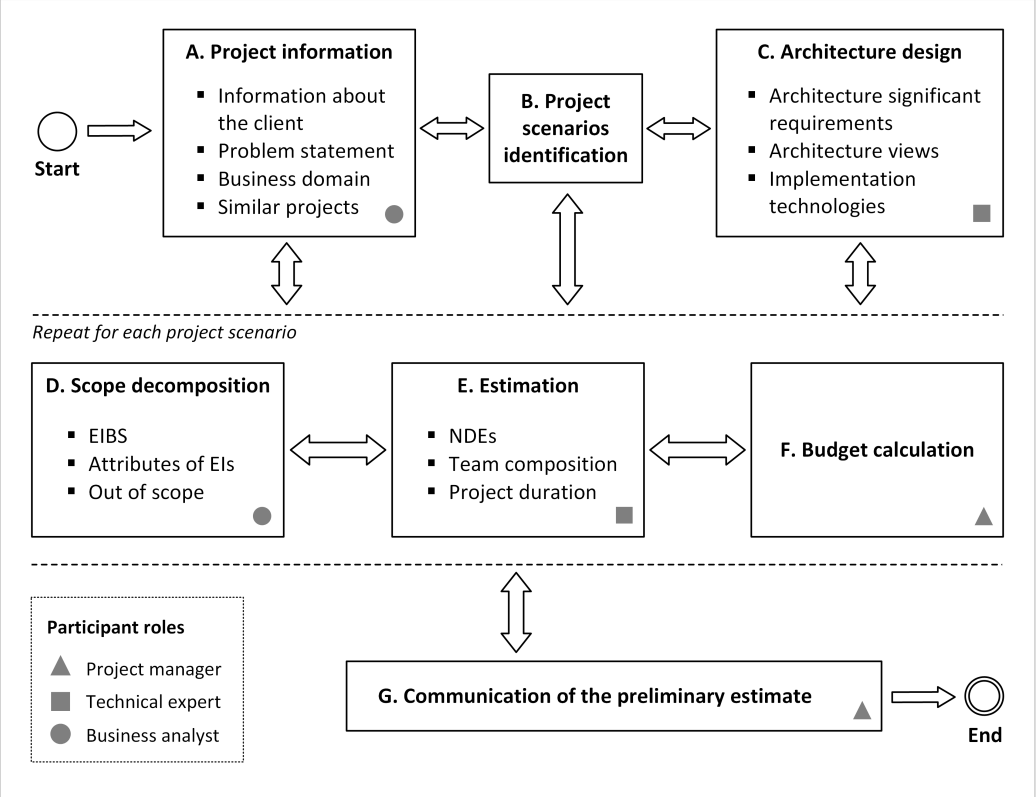

Figure 2. Preliminary estimation as a semistructured business process.

The process starts with understanding the essence of a project and getting familiar with the available requirements (step A). An important aspect of the process is identifying the project scenarios (step B). Under a project scenario, a hypothetical way of project implementation is understood. Depending on the circumstances, the criteria of scenario identification might be different, for example, development efforts, scope of work, implementation technologies, architecture design, etc. The "backbone" of the process consists of three steps: D, E, and F. In these steps, the estimation outcomes are produced. The rest of the Section 4 covers step E and, partially, step D. Importantly, steps D, E, and F are performed for each of the identified scenarios (it is worth noting that optimistic and pessimistic estimates represent an estimate range for a single scenario, not two different scenarios). The final step G is aimed at communication of the estimates to the concerned parties (e.g., to a potential client).

The process involves the following participant roles: a project manager, a technical expert, and a business analyst. The primary responsibility areas of each role are shown in Figure 2. However, it is worth noting that regardless of the primary responsible role, other roles are also supposed to contribute to a certain process step (e.g., a business analyst is responsible for the project scope decomposition; however, a technical expert can also contribute to this). In practice, a person can combine the duties of more than one role (e.g., a technical expert can also perform the tasks of a business analyst). Or, in the opposite case, the business analyst role can be covered by two people: a business analyst and a business domain expert (or a subject matter expert).

4.2. Estimable-Item Breakdown Structure

To provide reliable estimates, it is necessary to have some representation of the project scope—the object of estimation. Usually, the project scope is decomposed into a treelike construction called a work breakdown structure (WBS) [32] or into one of its subtypes (e.g., a

component-based work breakdown structure, CBWBS [33]). In practice, such a breakdown structure is received as the result of a combination of several decomposition approaches (e.g., work packages, work items, epics, features, components, use cases, etc.). In order to incorporate a project scope breakdown into the authors' estimation framework, the terms "estimable item" (EI) and "estimable-item breakdown structure" (EIBS) are introduced.

An *estimable item* (IE), x, is a representation of a project scope portion which can be sized (i.e., assigned with a normalized development estimate, NDE [9]), decomposed to child estimable items, and analyzed in terms of assumptions, dependencies, risks, etc. An estimable item x possesses a set of attributes; the value of an attribute can be denoted in squared brackets—x[attribute name]. For example, x[parent] is a parent of estimable item x; x[risks] is a set of risks associated with x. Especially important is x[NDE]. An item is called a *leaf estimable item* (LEI) when it does not have any child items, $x[\text{children}] = \varnothing$. In turn, if x does have child items, $x[\text{children}] \neq \varnothing$, it is named a *composite estimable item* (CEI).

An *estimable-item breakdown structure* (EIBS) X is a tree (in terms of graph theory) with estimable items as the vertices, parent–child relationships between the estimable items as the edges, and the root item representing the scope of the whole project.

To show how the preliminary estimation is applied, a software product named "Real-time Business Process Monitoring for Estimation" (RTBPM-E) [34,35] is used here and below. In Table 1 and in Figure 3, an EIBS is provided for RTBPM-E.

Table 1. Table-based representation of EIBS for RTBPM-E.

Identifier	Title	Description
RTBPM-E	Real-time business process monitoring for estimation	The whole scope of the RTBPM-E project.
EI-1	Table visualization of estimation process logs	Table-based visual representation of logs collected from performed estimation processes including standard table features such as filtering and grouping.
EI-2	Etalon model of the estimation process	Read-only visualization of the etalon model is based on the 4-stage estimation process: (a) introductory estimate, (b) preliminary estimate, (c) intermediate estimate, and (d) precise estimate.
IE-3	Actual model of the estimation process	Visualization of the estimation process model based on the collected logs built with process mining [11].
IE-3.1	Method of building a process model	Implementation of the method of building a process model utilizing the collected log data.
IE-3.2	Visualization of a process model	Graph-based visualization of a process model on a web page.
IE-4	Alerting on project estimation process	Alerts highlighting issues or risks during estimation of a particular project. Alerts are visualized on the graphic user interface and sent via email.
IE-5	Data processing pipeline	Integration with the data sources and implementation of a data processing pipeline [34].
IE-6	Security and user management	Typical security-related functionality: (a) authentication, (b) authorization, (c) user management, etc.
IE-7	Administration and configuration	Admin dashboard and configuration of the system.
IE-8	Project infrastructure	Setting up project infrastructure including structuring of the source code and continuous integration.

An important part of EIBS creation is assigning attributes to items. Such an attribute is nothing but a piece of information associated with an EI. In Table 2, attribute types are listed. They, in the authors' opinion, correspond to the most frequently analyzed aspects of the project scope.

Considering the limit on the preparation time and the high level of uncertainty, as well as not-so-strict accuracy requirements, an EIBS created during the preliminary estimation stage is not supposed to be quite detailed. Even identification of the first-level items might be enough to provide a preliminary estimate.

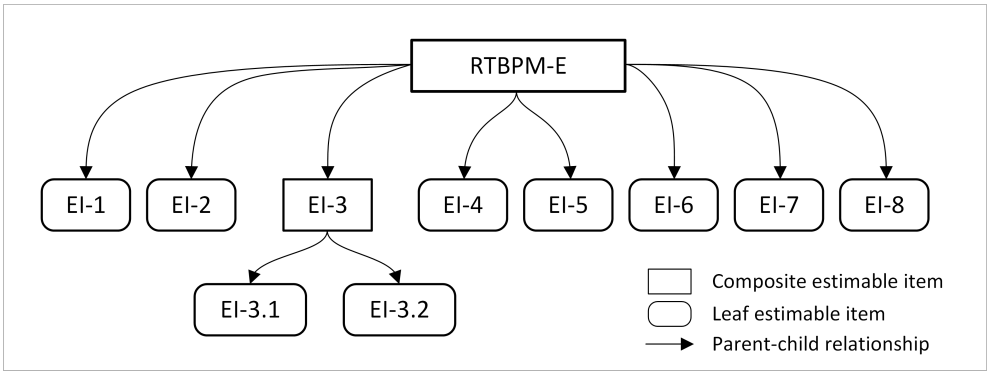

Figure 3. Treelike representation of EIBS for RTBPM-E.

Table 2. Most common types of estimable item attributes.

Name	Shortcut	Description
Identifier	ID	Uniquely identifies an EI in an EIBS.
Title	TITLE	Several words of summary of an EI.
Description	DESC	Explanation of what an EI is about.
Assumptions	ASMP	Assumptions associated with an EI.
Risks	RISK	Risks associated with an EI.
Dependencies	DEPN	Dependencies of an IE.
Functional requirements	FR	Functional requirements associated with an EI.
Nonfunctional requirements	NFR	Nonfunctional requirements associated with an EI.
Constraints	CON	Constraints associated with an EI.
Architecture considerations	ARCH	Architecture ideas, patterns, tactics.
Out of scope	OOS	Explicitly stated what is out of the project scope.
Normalized development estimate	NDE	NDE associated with an EI.
Questions	Q	Questions associated with an EI.

4.3. Sizing of Estimable Items

As already mentioned in Section 2, the authors' estimation framework uses NDE as a measure of development efforts. NDE is a time-based unit expressing the amount of work in man-hours, man-days, etc. Such time-based measuring ensures seamless translation of the estimated efforts into the project schedule. However, it is worth highlighting that the NDE itself is defined in a way that makes it independent from neither the project schedule nor the team composition.

In essence, sizing is aimed at designating each EI as an NDE. For the preliminary estimation, a two-step sizing approach is proposed: (a) designate each LEI using such attributes as an estimable item point (EIP) and an estimable item uncertainty (EIU); (b) then, obtain optimistic and pessimistic NDEs from the corresponding EIP and EIU.

Let $x \in X$ be an LEI belonging to EIBS X. The *estimable item point* (EIP) of x, $x[\text{EIP}]$, is a positive number representing the relative measure of development efforts required to implement x. In turn, the *estimable item uncertainty* (EIU), $x[\text{EIU}]$, is a non-negative dimensionless number expressing how much is unknown with regard to x; in other words, the bigger the $x[\text{EIU}]$, the less definitive the $x[\text{NDE}]$ and vice versa. It is worth emphasizing that EIPs and EIUs are supposed to be designated for LEIs (not CEIs).

EIPs and EIUs are based on experts' judgment. In Figure 4, an example is shown of EIP and EIU estimation based on the idea of affinity grouping [5]: the LEIs are placed on a coordinate plane where the horizontal axis corresponds to the size (EIP), and the vertical axis defines the level of uncertainty (EIU). One of the strengths of the described approach is its visual representation of the project scope on a two-dimensional plain, allowing a relative comparison of EIs' sizes and uncertainties.

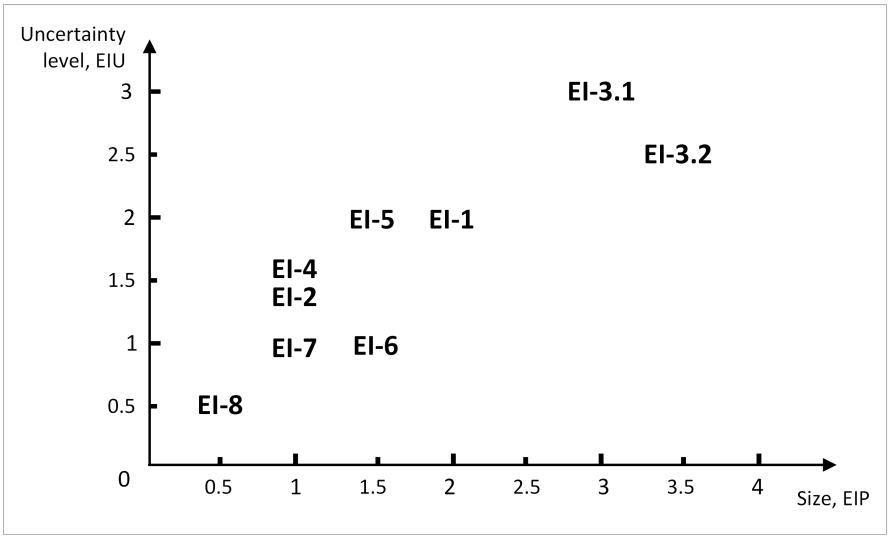

Figure 4. Defining EIPs and EIUs for the RTBPM-E estimable items.

After estimating EIPs and EIUs, it is necessary to transform them into NDEs. Define the relationship between NDE and EIP as follows:

$$x[\text{NDE}_{\text{basic}}] = p \cdot x[\text{EIP}], \tag{5}$$

where $p > 0$ is the NDE corresponding to one EIP. In turn, optimistic and pessimistic NDEs are related to the EIU as follows:

$$\begin{aligned} x\left[\text{NDE}_{\text{opt}}\right] &= (1 - u_1 \cdot x[\text{EIU}]) \cdot x[\text{NDE}_{\text{basic}}], \\ x\left[\text{NDE}_{\text{psm}}\right] &= (1 + u_2 \cdot x[\text{EIU}]) \cdot x[\text{NDE}_{\text{basic}}], \end{aligned} \tag{6}$$

where $x\left[\text{NDE}_{\text{opt}}\right]$ and $x\left[\text{NDE}_{\text{psm}}\right]$ are the optimistic and pessimistic NDEs, respectively; $0 \leq u_1 < 1$ and $u_2 \geq 0$ are the coefficients defining deviation of the optimistic and pessimistic estimates from the basic NDE. Values of parameters p, u_1, u_2 can be either based on experts' judgment or defined statistically from past projects. In Table 3, an example of applying the above approach to RTBPM-E is represented; for calculations, the following values of the parameters were chosen by the authors: $p = 168 \frac{\text{man-hour}}{\text{EIP}}$, $u_1 = 0.1$, $u_2 = 0.2$.

Table 3. NDEs of the RTBPM-E estimable items.

IE Identifier	EIP	Basic NDE, Man-Hours	EIU	Optimistic NDE, Man-Hours	Pessimistic NDE, Man-Hours
EI-1	2	336.0	2	268.8	470.4
EI-2	1	168.0	1.5	142.8	218.4
IE-3.1	3	504.0	3	352.8	806.4
IE-3.2	3.5	588.0	2.5	441.0	882.0
IE-4	1	168.0	1.5	142.8	218.4
IE-5	1.5	252.0	2	201.6	352.8
IE-6	1.5	252.0	1	226.8	302.4
IE-7	1	168.0	1	151.2	201.6
IE-8	0.5	84.0	0.5	79.8	92.4
Total	–	2520.0	–	2007.6	3544.8

Therefore, the RTBPM-E NDEs are the following:

$$\begin{aligned} \text{RTBPM-E}[\text{NDE}_{\text{basic}}] &= 2520.0 \,\text{man-hours}, \\ \text{RTBPM-E}\left[\text{NDE}_{\text{opt}}\right] &= 2007.6 \,\text{man-hours}, \\ \text{RTBPM-E}\left[\text{NDE}_{\text{psm}}\right] &= 3544.8 \,\text{man-hours}, \end{aligned}$$

where the optimistic and pessimistic estimates form the range $-20.3\% \ldots + 40.7\%$ (relative to the NDE_{basic}), which, from authors' perspective, is acceptable for the preliminary estimation.

4.4. Project Team Composition

Along with the NDE discussed in the previous section, project team composition is one of the key ingredients of an estimate. As can be seen in the sections below, the varying of the project team composition allows to one to obtain estimates with different project durations and costs.

Project team composition means a set of project team member roles, T, and the attributes associated with each role (e.g., a full-time equivalent, FTE). A project team includes roles in two main categories: development, $T_D \subseteq T$ (software engineers), and nondevelopment, $T_{ND} \subseteq T$ (e.g., project managers, test engineers, etc.). The main difference between these categories is that the efforts spent by team members in development roles are estimated in the NDE, while the efforts of the nondevelopment roles are not included in the NDE.

In order to match FTEs of development roles with the NDE, let us extend the estimation framework with a new term—*normalized development full-time equivalent* (ND-FTE), ψ:

$$\psi = \rho\,\phi, \tag{7}$$

where ρ is the productivity coefficient (PC) defined in [9]; ϕ is the corresponding FTE. Using the introduced term, a team composition with nondifferentiated specializations applicable to RTBPM-E is provided in Table 4.

Table 4. Team composition for RTBPM-E.

Identifier (m)	Type *	Role	FTE (ϕ^m)	CLPC (α^m)	IPC (η^m)	PC (ρ^m)	ND-FTE (ψ^m)
1	D	Senior developer	1.5	1.2	0.9	1.08	1.62
2	D	Middle developer	4.0	1.0	1.0	1.00	4.00
3	D	Junior developer	2.0	0.7	1.0	0.70	1.40
4	ND	Project manager	1.0	n/a	n/a	n/a	n/a
5	ND	Technical leader	0.5	n/a	n/a	n/a	n/a
6	ND	Business analyst	1.0	n/a	n/a	n/a	n/a
7	ND	UX designer	1.0	n/a	n/a	n/a	n/a
8	ND	DevOps engineer	1.0	n/a	n/a	n/a	n/a
9	ND	Test engineer	3.0	n/a	n/a	n/a	n/a

* D and ND stand for "development" and "nondevelopment", respectively.

In most cases, the simplest type of development team composition—with nondifferentiated specializations—fulfills the preliminary estimation needs. However, in situations where highlighting development specializations is quite important, development teams with differentiated or even mixed specializations can also be applicable at the preliminary estimation stage. Further information about the team composition types is in Section 5.

4.5. System of Working-Time Balance Equations

The idea of a system of working-time balance equations was introduced in the authors' past works [9,10]. Its purpose is to define the relationships between the key estimate ingredients such as the structure of software developer working time, project team composition, project duration, and NDE.

For the preliminary estimation, a simplified version of the system of working-time balance equations is used. To achieve the simplification, we assume the following:

1. The project timeline is not split into sprints or phases.
2. The project team does not change throughout the whole project.
3. There is no differentiation of the development specializations.
4. There is a linear relationship between the project working time, W, and the development working time, D (in contradiction to (1), where that relationship is based on the structure of software developer working time):

$$W^m = \xi\, D^m,\ m \in T_D, \tag{8}$$

where $\xi > 1$ is the *development working time coefficient* (DWTC) (again, for simplicity reasons, it is assumed that ξ does not depend on project role $m \in T_D$). Further information about the DWTC is provided in Section 5.

Therefore, the system of working-time balance equations for the preliminary estimation is as follows:

$$\begin{cases} W^m = \xi D^m, \, m \in T_D, \\ W^m = \phi^m L, \, m \in T, \\ E = \sum\limits_{m \in T_D} \rho^m D^m, \end{cases} \tag{9}$$

where T is the set of project roles; $T_D \subseteq T$ is the subset of development roles; W^m is the project working time (PWT) of role $m \in T$; D^m is the development working time (DWT) of role $m \in T_D$; $\xi > 1$ is the development working time coefficient (DWTC); ϕ^m is the full-time equivalent (FTE) of role $m \in T$; L is the duration of the project; E is the normalized development estimate (NDE) of the entire project; ρ^m is the productivity coefficient (PC) of development role $m \in T_D$.

4.6. Estimation of Normalized Development Capacity

One of the key characteristics of a project team is the normalized development capacity (NDC), which is the maximum possible NDWT that can be spent on the project scope implementation (Equation (4) in Section 2). For the preliminary estimation, the following expression for NDC takes place:

$$C^m = \max_{U^m \in \mathbb{U}^m} U^m, m \in T_D, \tag{10}$$

where C^m is an NDC of development role $m \in T_D$; U^m is an NDWT of development $m \in T_D$; $\mathbb{U}^m$ is a set of all possible values of NDWT. Taking into account (2), NDC is expressed as follows:

$$C^m = \rho^m \max_{D^m \in \mathbb{D}^m} D^m, m \in T_D, \tag{11}$$

where $\mathbb{D}^m$ is a set of all possible values of the DWT for development role $m \in T_D$. Then, from (9), it follows:

$$C^m = \frac{L}{\xi} \psi^m, m \in T_D, \tag{12}$$

and

$$C = \sum_{m \in T_D} C^m = \frac{L}{\xi} \sum_{m \in T_D} \psi^m. \tag{13}$$

In practice, (12) and (13) can be used to solve an inverse problem—finding the amount of project scope that a particular project team is capable of implementing if project duration L is known.

4.7. Estimation of Project Duration

In the case of the preliminary estimation, the main criterion of estimating project duration is that the project team must have enough NDC to implement the project scope, measured as NDE, E:

$$C = \frac{L}{\xi} \sum_{m \in T_D} \psi^m \geq E. \tag{14}$$

Therefore, project duration, L, is estimated with the following inequation:

$$L \geq \frac{\xi E}{\sum\limits_{m \in T_D} \psi^m}. \tag{15}$$

An example of estimating the optimistic and pessimistic durations for RTBPM-E is represented in Table 5—to implement the project scope from Table 3 utilizing the project team defined in Table 4, it will take from $L_{opt} = 6.5$ to $L_{psm} = 11.5$ months. Due to the

high level of uncertainty, the estimated duration range is wide, which is expected for the preliminary estimation.

Table 5. Duration estimation for the RTBPM-E project.

	Optimistic	Pessimistic
NDE, man-hours	2007.6	3544.8
DWTC	3.8 *	3.8 *
Development FTE	7.5	7.5
ND-FTE	7.02	7.02
Duration, hours	1086.8	1918.9
Working hours per month	168	168
Duration, months	6.47	11.42
Rounded duration, months	6.5	11.5

* The DWTC value is based on the authors' expert judgment.

The duration estimation based on (15) does not guarantee high accuracy; instead, it allows a roughly evaluation of the duration of the project using relatively simple calculations. Narrowing down the estimate range will be undertaken in the next estimation stages.

4.8. Optimization of Project Duration and Team Composition

As can be seen above, the preliminary estimation operates with these three main ingredients: project scope, team composition, and project duration. Given that the project scope is fixed (i.e., the NDE does not vary), the other two components are interdependent: changes in the team composition imply different project durations and vice versa: depending on the project duration, different team compositions are required. Manual selection of the best combination of these two ingredients requires time spent on calculations. To make this more efficient, a multiobjective optimization is proposed:

$$F = L \sum_{m \in T} r^m \phi^m \rightarrow \min, \tag{16}$$

$$L \rightarrow \min, \tag{17}$$

$$\Phi = \sum_{m \in T} \phi^m \rightarrow \min, \tag{18}$$

$$I = C - E \rightarrow \min, \tag{19}$$

where T is the set of project team roles; $m \in T$ is a particular role belonging to the team; $r^m \in [0,1]$ is the normalized hourly rate of role $m \in T$; W^m is the PWT of role $m \in T$; ϕ^m is the FTE of role $m \in T$; L is the project duration; C is the NDC of the development team $T_D \subset T$; E is the NDE; I is the development team idle time. It is worth noting that applying normalization to the cost-related variables brings the following benefits: (a) avoiding disclosure of commercially sensitive information; (b) avoiding too-big values of the objective function; (c) currency-independent calculations with further conversion of the normalized costs to a required currency.

One of the main constraints to be satisfied is (14)—the chosen team composition and project duration have to allow the implementation of the project scope estimated as E. Also, it is worth applying restrictions on the project duration:

$$[L]_{\min} \leq L \leq [L]^{\max}. \tag{20}$$

The other group of constraints is applicable to the project role FTEs:

$$\left[\phi^{T^*}\right]_{\min} \leq \sum_{m \in T^*} \phi^m \leq \left[\phi^{T^*}\right]^{\max}, \tag{21}$$

where $T^* \subseteq T$ is a subteam of project team T. For example, a team has to include at least one middle software engineer: $1 \leq \sum_{m \in T^{\text{mid}}} \phi^m$ (where $T^{\text{mid}} \subseteq T$ is a subset of middle software engineers); or, the size of the whole team, T, does not have to exceed 25 FTEs: $\sum_{m \in T} \phi^m \leq 25$. The interrelation of FTEs for different project roles is expressed with this type of constraint:

$$\gamma_1 \sum_{m_1 \in T_1} \phi^{m_1} \leq \gamma_2 \sum_{m_2 \in T_2} \phi^{m_2}, \tag{22}$$

where $T_1 \subseteq T$ and $T_2 \subseteq T$ are a subteams of T; $\gamma_1 > 0$ and $\gamma_2 > 0$ are constants. For example, 1 project manager cannot lead more than 15 team members: $\sum_{m_1 \in T \setminus T^{\text{mgt}}} \phi^{m_1} \leq 15 \sum_{m_2 \in T^{\text{mgt}}} \phi^{m_2}$ (where $T^{\text{mgt}} \subseteq T$ is a subteam of project managers).

Let us substitute real decision variables ϕ^m with the corresponding integer variables f^m:

$$\phi^m = \mu^m f^m,\ m \in T, \tag{23}$$

where $f^m \in \mathbb{N}_0 = \{0, 1, 2, \ldots\}$ is a whole number of minimum FTEs for role $m \in T$; $\mu^m > 0$ is a minimal possible step of FTE change for role $m \in T$. Also, let us vary the project duration within a range (20):

$$[L]_{\min} = L_1 < L_2 < \ldots < L_p = [L]^{\max}. \tag{24}$$

Therefore, for each L_k, $k = \overline{1, p}$, a sequence of integer programming problems with an objective (16), constraints (14), (21), (22), and decision variables (23) is received. Solving these optimization tasks produces a sequence of p alternatives $(L_k, T_k, F_k, \Phi_k, I_k)$, $k = \overline{1, p}$. Then, the alternatives are ranked using the analytic hierarchy process (AHP) [36].

In application to the RTBPM-E example, let us solve a sequence of integer programming problems, varying the project duration from $[L]_{\min} = 4$ months to $[L]^{\max} = 16$ months with the step of 0.5 month for both optimistic and pessimistic estimates. Then, the received alternatives are ranked with AHP according to the criteria from Table 6. The top alternatives are provided in Tables 7 and 8 for the optimistic and pessimistic estimates, respectively. As a result, for the optimistic estimate, alternative 1 is chosen (as recommended according to the AHP ranking). However, alternative 2 is selected for the pessimistic estimate (in this regard, it is worth emphasizing that the AHP-based alternative ranking is just a decision support tool, while the final conclusion is made by the experts). Table 9 provides the project team composition corresponding to the chosen alternatives.

Table 6. AHP criteria comparison for RTBPM-E.

	Cost	Duration	Team Size	Idle Time
Cost	1	3	1	5
Duration	1/3	1	1	3
Team size	1	1	1	1
Idle time	1/5	1/3	1	1

Table 7. Top alternatives for the RTBPM-E optimistic estimate.

No.	AHP Rank	Duration (L), Months	Norm. Cost (F)	NDC (C), Man-Hours	NDC-NDE, Man-Hours	Team FTE (ϕ)	Develop. FTE (ϕ^D)
1	0.069	5.5	10,395.00	2061.98	54.38	15.75	9.00
2	0.068	4.0	10,668.00	2072.59	64.99	22.25	12.50
3	0.061	6.5	11,193.00	2149.52	141.92	14.25	8.00
4	0.060	6.0	11,340.00	2249.43	241.83	15.75	9.00

Table 8. Top alternatives for the RTBPM-E pessimistic estimate.

No.	AHP Rank	Duration (L), Months	Norm. Cost (F)	NDC (C), Man-Hours	NDC-NDE, Man-Hours	Team FTE (ϕ)	Develop. FTE (ϕ^D)
1	0.075	6.5	18,099.90	3655.33	110.53	23.25	13.50
2	0.072	9.5	17,955.00	3561.60	16.80	15.75	9.00
3	0.070	7.0	18,669.00	3627.03	82.23	22.25	12.50
4	0.068	9.0	18,711.00	3589.01	44.21	17.00	9.50
5	0.068	11.0	18,942.00	3637.64	92.84	14.25	8.00

Table 9. Optimized team composition for RTBPM-E.

Identifier (m)	Type *	Role	FTE (ϕ^m)	CLPC (α^m)	IPC (η^m)	PC (ρ^m)	ND-FTE (ψ^m)
1	D	Senior developer	1	1.2	0.9	1.08	1.08
2	D	Middle developer	6	1	1	1	6
3	D	Junior developer	2	0.7	1	0.7	1.4
4	ND	Project manager	1	n/a	n/a	n/a	n/a
5	ND	Technical leader	1	n/a	n/a	n/a	n/a
6	ND	Business analyst	1	n/a	n/a	n/a	n/a
7	ND	UX designer	0.75	n/a	n/a	n/a	n/a
8	ND	DevOps engineer	0.5	n/a	n/a	n/a	n/a
9	ND	Test engineer	2.5	n/a	n/a	n/a	n/a

* D and ND stand for "development" and "nondevelopment", respectively.

The calculations in the current section were performed with a Python script using the following libraries: (a) Pyomo v.6.5.0 (https://www.pyomo.org/, (accessed on 11 May 2023)) as an optimization model builder; (b) FICO Xpress v.9.1.0 under the community license (https://www.fico.com/, (accessed on11 May 2023)) as an optimization task solver; (c) ahpy v.2.0.0 (https://github.com/PhilipGriffith/AHPy, (accessed on 11 May 2023)) for the AHP-based ranking of the alternatives.

A comparison of the manual (Tables 4 and 5) and optimized (Tables 7–9) estimates is given in Tables 10 and 11—the optimized estimate, on the one hand, slightly increases the ND-FTE and, on the other hand, outperforms the manual estimate, reducing the project duration and cost.

Table 10. Comparison of the manual and optimized optimistic estimates for RTBPM-E.

	Manual	Optimized	Manual–Optimized
Duration, months	6.5	5.5	−1.00 (−15.4%)
Team FTE	15.00	15.75	+0.75 (+5.0%)
Development FTE	7.50	9.00	+1.50 (+20.0%)
ND-FTE	7.02	8.48	+1.46 (+20.8%)
Normalized cost	12,285.00	10,395.00	−1890.00 (−15.4%)

Table 11. Comparison of the manual and optimized pessimistic estimates for RTBPM-E.

	Manual	Optimized	Manual–Optimized
Duration, months	11.5	9.5	−2.00 (−17.4%)
Team FTE	15.00	15.75	+0.75 (+5.0%)
Development FTE	7.50	9.00	+1.50 (+20.0%)
ND-FTE	7.02	8.48	+1.46 (+20.8%)
Normalized cost	21,735.00	17,955.00	−3780.00 (−17.4%)

The proposed decision support tool set reduces the experts' time spent on deciding on the team composition and the project duration. However, it requires a specific software implementation and calibration of the parameters.

5. Discussion

5.1. Project Scope Decomposition

Building an EIBS might be one of the most challenging and time-consuming parts of a preliminary estimation. This is mostly caused by quite high uncertainty and limited time.

Despite this, it is necessary to break down the project scope at a high level without diving into details but, at the same time, covering all of the functionality of the future project.

Working with an EIBS includes these three main steps: (a) decomposition; (b) analysis; (c) sizing. Each of them is related to the necessity of processing a quite large amount of information under time restrictions. In this regard, the following mistakes are possible: (a) missing significant parts of the project functionality; (b) blowing up the scope by including unnecessary features; (c) incorrect structuring of the project scope, which can be difficult to elaborate on during the next estimation stages.

These are some recommendations that can help with the challenges mentioned above:

1. Use a project scope visualization that makes the perception and analysis of large amounts of information easier.
2. Cover the unknown with assumptions and risks.
3. Add placeholder EIs to cover the unknown parts of the functionality.
4. Use EIBS templates based on historical estimates and projects.

Methods of building an EIBS still require further research and testing in practice.

5.2. Development Specializations

Depending on distinguishing specializations, a development team belongs to one of the following three types. The simplest is a team with *nondifferentiated specializations*. It is worth noting that this does not mean that the developers on such a team do not have specializations during the project implementation phase; instead, it means that at certain estimation stages, those specializations are just not defined. For the preliminary estimation, this kind of team is used for simplicity reasons. In the case of a development team with *differentiated specializations*, each development role belongs to one of the specializations. Developers belonging to this type of team can perform tasks of a single specialization only (not two or more). And the third type is a development team with *mixed specializations*, where a particular software engineer can perform tasks belonging to several specializations. Such a situation takes place, for example, when a so-called full-stack developer performs both front-end and back-end tasks. In the case of nondifferentiated specializations, NDE and NDC are single numbers, while for the other two types they are defined as vectors, each component of which corresponds to a particular specialization [9,10].

5.3. Development Working Time Coefficient

The current section aims at disclosing the nature of the development working time coefficient (DWTC).

For intermediate estimation [9], the relationship between PWT and DWT reflects the structure of developer working time:

$$\overline{W}_D = \sum_{m \in \overline{T}_D} \sum_{i \in \overline{K}} \overline{W}_i^m = \sum_{m \in \overline{T}_D} \sum_{i \in \overline{K}} \left[\sum_{s \in \overline{Q}} \left(\overline{D}_i^{m(s)} + \overline{A}_i^{m(s)} \right) + \overline{G}_i^m + \left(\overline{O}_i^m + \overline{I}_i^m \right) \right], \tag{25}$$

where $\overline{W}_D$ is the development project working time; $\overline{T}_D$ is the set of development project roles; $\overline{K}$ is the set of project phases; $\overline{Q}$ is the set of development specializations; $\overline{W}_i^m$, $\overline{D}_i^m$, $\overline{A}_i^m$, $\overline{G}_i^m$, $\overline{O}_i^m$, $\overline{I}_i^m$ are, respectively, the project working time (PWT), the development working time (DWT), the supplementary development activities, the general project activities, the leave time, and the idle time for development role $m \in \overline{T}_D$ in project phase $i \in \overline{K}$ (further information about these variables is provided in [9]).

The DWTC from (8) is introduced in order to decrease complexity by reducing the number of variables in the system of working-time balance equations. Therefore, for the preliminary estimation, development project working time, W_D, is expressed as follows:

$$W_D = \sum_{m \in T_D} W^m = \xi \sum_{m \in T_D} D^m, \tag{26}$$

where ξ is the development working time coefficient (DWTC); D^m is the development working time (DWT) of role $m \in T_D$. Following the concept of the ELC, a preliminary estimate is supposed to be "wider" than the corresponding intermediate estimate (given that the NDE is the same for both estimates); on the other hand, W_D does not have to be too big to prevent making a wrong impression regarding the project duration and budget:

$$q \cdot \overline{W}_D \geq W_D \geq \overline{W}_D, \tag{27}$$

where $q > 1$ is the coefficient defining the high boundary of PWT, W_D, received as the outcome of preliminary estimation; $\overline{W}_D$ is the corresponding PWT resulting from the intermediate estimation.

Summarizing the above, DWTC depends on what is defined by (25): (a) the structure of software developer working time; (b) the project release plan (splitting into project phases); (c) the development team composition (roles, development specializations, involvement in different project phases). And the coefficient's boundaries are limited according to (27). To define DWTC, it is suggested that one rely on the data from historical projects implemented in a particular company as well as on experts' judgment. For practice usage, it is recommended that a decision tree is created that allows one to obtain a value of DWTC based on the specifics of a particular project.

5.4. Alternative Approach to Define Development Working Time Coefficient

In the current section, an alternative way to simplify the system of working-time balance equations is presented. Let us keep Assumptions 1–3 from Section 4.5, replacing the fourth assumption with a linear relationship between the PWT of the development team, $W_D = \sum\limits_{m \in T_D} W^m$, and the NDE, E:

$$W_D = \kappa E, \tag{28}$$

where $\kappa > 1$ is an alternative way to define the DWTC (which is originally introduced in (8)). This implies the following system of working-time balance equations:

$$\begin{cases} \kappa E &= \sum\limits_{m \in T_D} W^m, \\ W^m &= \phi^m L, \; m \in T, \\ E &= \sum\limits_{m \in T_D} \rho^m D^m, \end{cases} \tag{29}$$

an expression for NDC:

$$C = \frac{L}{\kappa} \sum_{m \in T_D} \phi^m, \tag{30}$$

and the estimated project duration:

$$L \geq \frac{\kappa E}{\sum\limits_{m \in T_D} \phi^m}. \tag{31}$$

As can be noticed, (30) and (31) are quite similar to (13) and (15), respectively. The only difference is that (30) and (31) rely on the development FTE, $\sum\limits_{m \in T_D} \phi^m$, while (13) and (15) are based on the ND-FTE (which, in turn, takes into account productivity coefficients ρ^m of each development role $m \in T_D$). From this standpoint, (28) results in even simpler estimation formulas in comparison to the ones following from (8).

To choose between the two approaches for a particular case, it is recommended that one take into account such factors as the necessity of considering the PCs and the availability of historical data to define the DWTC (in particular, (8) requires more granular historical project data than (28)).

5.5. Threats to Validity

The conditions under which usage of the proposed preliminary estimation is not recommended are discussed in the current section.

The preliminary estimation is supposed to be applicable in the early project stages, when the level of "unknown" is quite high. However, its usage is not recommended when the level of uncertainty is so high that it is not possible to define the project scope (i.e., build an EIBS) even at a high level without going into details. In such a case, an introductory estimation based on project similarity might be more suitable.

In situations where it is difficult to define the DWTC, usage of preliminary estimation is not desirable. This might happen, for example, in the case of completely new implementation technologies or project teams whose performance is difficult to forecast. As a solution, intermediate estimation [9,10] can be applied. Intermediate estimation does not use the DWTC; instead, it requires more detailed analysis and operates with more parameters.

Another case when preliminary estimation is not applicable is the situation when the composition of a project team significantly changes along the project timeline. Intermediate estimation [9,10] is more suitable under this condition, since team composition changes are built into this method.

As can be seen from (18) and (19), the preliminary estimation defines the project duration depending on the NDE of the project scope and the NDC of the project team. Obviously, in situations where there are strict dependencies between project tasks or dependencies on other projects, the preliminary as well as intermediate estimation project duration formulas are not going to work. Using approaches based on the critical path method (CPM) and the project evaluation and review technique (PERT) [3,4,37,38] might be a solution in such a case.

Although the ELC (Figure 1) defines preliminary estimation as one of the successive steps that are complementary to the SDLC stages, under the circumstances described above, it is suggested that one skip the preliminary estimation, replacing it with other methods.

6. Conclusions

The preliminary estimation proposed in the current study is an integral part of the authors' estimation framework [9,10]. According to the concept of the estimation life cycle (ELC), the preliminary estimation can be either the first stage or a logical continuation of the predecessor stage—the introductory estimation. In the second case, apart from an approximation of a timeline and budget, the introductory stage passes along the problem statement understanding, information about a business domain, and identified similar projects or products. Then, at the subsequent stage—the intermediate estimation—the inputs from the previous step are made more definitive by (a) reducing the number of project scenarios (ideally, to one main scenario); (b) detailing an EIBS; (c) adding specializations to a development team; and (d) splitting a project timeline into phases.

The key advantage of the proposed preliminary estimation is that it belongs to the estimation framework covering the whole SDLC. Except for COCOMO II [2], none of the existing methods is aimed at supporting the entire software project life cycle. Despite its high level of maturity, the main obstacle to using COCOMO II is its complexity—preliminary estimation clearly has an advantage in this aspect. Use-case point analysis [17,27–30], which is also applicable in the early project stages, does not lose to the preliminary estimation regarding ease of use; however, in the authors' opinion, the formulas aimed at calculating use-case points and translating them to time-based units require a certain rethinking. Probably, an adjusted version of use-case point analysis might be a useful extension of the preliminary estimation sizing approach. The agile estimation methods [5–8] are supposed to be applied rather at the implementation phase by an agile team when the project scope is decomposed to the user story level. Furthermore, those methods suffer from the lack of a tool set to translate story points to time-based units. Therefore, agile estimation methods (as they are originally defined) are hardly applicable at the early project stages. Similarity-

based estimation [20–23] can be either used in the introductory estimation stage or applied in conjunction with the preliminary estimation in order to verify the results.

One of the most time-consuming and challenging parts of preliminary estimation is the project scope decomposition, i.e., building an EIBS. Usually, for this reason, an expert (a business analyst or a domain expert) has to process a large amount of information within a limited time frame. Some ideas on how to overcome such difficulties are given in Section 5. However, this matter is a separate topic that still requires further research.

An important role that preliminary estimation plays is in the development working time coefficient (DWTC). Despite the considerations in Section 5, the recommendations on how to choose a DWTC value for a particular project are still not definitive enough. Obviously, the past project estimates are supposed to be used to define the coefficient's values. In the case of insufficiency of such historical data, DWTC values can rather be based on experts' judgment. To fully cover DWTC-related questions, further research is needed.

From the practical implementation standpoint, preliminary estimation is a semistructured business process that, on the one hand, states what has to be done, and, on the other hand, intentionally leaves certain space for the involved experts' creativity.

The proposed method of preliminary estimation is simple enough to be easily implemented with a spreadsheet like MS Excel or Google Spreadsheet. However, to ensure the use of the decision support tool set and to take full advantage of the estimation life cycle (ELC) concept, the method's integration into a fully scaled estimation information technology is envisioned [34], including other estimation methods with the ability to accumulate historical data (which, in particular, will be beneficial for the use of AI-based techniques). Apart from estimation methods, the envisioned information technology will include a process-mining module aimed at monitoring and improving the estimation processes [35].

Author Contributions: Conceptualization, V.T. and V.V.; investigation, A.B. and V.V.; methodology, V.V.; project administration, V.T. and A.B.; software, V.V.; validation, V.T. and A.B.; visualization, V.V.; writing—original draft, V.T. and V.V.; writing—review and editing, V.T. and V.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to express appreciation for the editors and the anonymous reviewers for their insightful suggestions to improve the quality of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AHP	Analytic hierarchy process
AI	Artificial intelligence
ASD	Average software developer
CEI	Composite estimable item
CLPC	Competency-level productivity coefficient
COCOMO	Constructive cost model
CPM	Critical path method
DWT	Development working time
DWTC	Development working time coefficient
EI	Estimable item
EIBS	Estimable-item breakdown structure
EIP	Estimable item point
EIU	Estimable item uncertainty
ELC	Estimation life cycle
FTE	Full-time equivalent
IPC	Involvement productivity coefficient

LEI	Leaf estimable item
NDC	Normalized development capacity
NDE	Normalized development estimate
ND-FTE	Normalized development full-time equivalent
NDWT	Normalized development working time
PC	Productivity coefficient
PERT	Project evaluation and review technique
PWT	Project working time
ROM	Rough order of magnitude
RTBPM-E	Real-time business process monitoring for estimation
SDLC	Software development life cycle
SLOC	Source lines of code
UML	Unified modeling language

References

1. McConnell, S. *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last*; Microsoft Press: Redmond, WA, USA, 1998.
2. Boehm, B.W.; Abts, C.; Brown, A.W.; Devnani-Chulani, S.; Clark, B.K.; Horowitz, E.; Madachy, R.J.; Reifer, D.J.; Steece, B. *Software Cost Estimation with COCOMO II*; Prentice-Hall: Saddle River, NJ, USA, 2000.
3. Bureau of Naval Weapons, United States, Special Projects Office. *Program Evaluation Research Task PERT Summary Report: Phase 1. Technical report, Special Projects Office, Bureau of Naval Weapons*; Department of the Navy: Washington, DC, USA, 1958.
4. Bureau of Naval Weapons, United States, Special Projects Office. *Program Evaluation Research Task PERT Summary Report: Phase 2. Technical report, Special Projects Office, Bureau of Naval Weapons*; Department of the Navy: Washington, DC, USA, 1958.
5. Mallidi, R.K.; Sharma, M. Study on Agile Story Point Estimation Techniques and Challenges. *Int. J. Comput. Appl.* **2021**, *174*, 9–14. [CrossRef]
6. Sudarmaningtyas, P.; Mohamed, R. A Review Article on Software Effort Estimation in Agile Methodology. *Pertanika J. Sci. Technol.* **2021**, *29*, 837–861. [CrossRef]
7. Munialo, S.W.; Muketha, G.M. A Review of Agile Software Effort Estimation Methods. *Int. J. Comput. Appl. Technol. Res.* **2016**, *5*, 612–618. [CrossRef]
8. Vyas, M.; Bohra, A.; Lamba, D.C.S.; Vyas, A. A Review on Software Cost and Effort Estimation Techniques for Agile Development Process. *Int. J. Recent Res. Asp.* **2018**, *5*, 1–5.
9. Teslyuk, V.; Batyuk, A.; Voityshyn, V. Method of Software Development Project Duration Estimation for Scrum Teams with Differentiated Specializations. *Systems* **2022**, *123*, 123. [CrossRef]
10. Teslyuk, V.; Batyuk, A.; Voityshyn, V. Method of Recommending a Scrum Team Composition for Intermediate Estimation of Software Development Projects. In Proceedings of the 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 10–12 November 2022; pp. 373–376. [CrossRef]
11. van der Aalst, W. *Process Mining: Data Science in Action*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
12. Di Ciccio, C.; Marrella, A.; Russo, A. Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. *J. Data Semant.* **2015**, *4*, 29–57.[CrossRef]
13. Project Management Institute. *Project Management Institute, a Guide to the Project Management Body of Knowledge (PMBOK-Guide)—Sixth Version*; PMBOK® Guide; Project Management Institute: Newtown Square, PA, USA, 2017.
14. Stutzke, R.D. *Estimating Software-Intensive Systems: Projects, Products, and Processes*; SEI Series in Software Engineering; Addison Wesley: Upper Saddle River, NJ, USA, 2005.
15. Trendowicz, A.; Jeffery, R. *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*; Springer International Publishing: Cham, Switzerland, 2014. [CrossRef]
16. Albrecht, A.J. Measuring Application Development Productivity. In *Proceedings of the IBM Applications Development Symposium*, Monterey, CA, USA, 14–17 October 1979; IBM Corporation: White Plains, NY, USA, 1979.
17. Karner, G. Resource Estimation for Objectory Projects. 1993. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=17b5f04743cd13f6077fbdec227719e5d83dba10 (accessed on 12 July 2023).
18. Boehm, B.W. *Software Engineering Economics*; Prentice-Hall: Saddle River, NJ, 1981.
19. Gorey, J.M. Estimate Types. *AACE Bull.* **1958**, *1*, 4–17.
20. Keung, J. Software Development Cost Estimation Using Analogy: A Review. In Proceedings of the Software Development Cost Estimation Using Analogy: A Review, Gold Coast, QLD, Australia, 14–17 April 2009; pp. 327–336. [CrossRef]
21. Idri, A.; azzahra Amazal, F.; Abran, A. Analogy-based software development effort estimation: A systematic mapping and review. *Inf. Softw. Technol.* **2015**, *58*, 206–230. [CrossRef]
22. Phannachitta, P. Robust comparison of similarity measures in analogy based software effort estimation. In Proceedings of the 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Malabe, Sri Lanka, 6–8 December 2017; pp. 1–7.[CrossRef]
23. Phannachitta, P. On an optimal analogy-based software effort estimation. *Inf. Softw. Technol.* **2020**, *125*, 106330. [CrossRef]

24. Auch, M.; Weber, M.; Mandl, P.; Wolff, C. Similarity-based analyses on software applications: A systematic literature review. *J. Syst. Softw.* **2020**, *168*, 110669. [CrossRef]
25. Hameed, S.; Elsheikh, Y.; Azzeh, M. An optimized case-based software project effort estimation using genetic algorithm. *Inf. Softw. Technol.* **2023**, *153*, 107088. [CrossRef]
26. Mustafa, E.I.; Osman, R. A random forest model for early-stage software effort estimation for the SEERA dataset. *Inf. Softw. Technol.* **2024**, *169*, 107413. [CrossRef]
27. Satapathy, S.M.; Acharya, B.P.; Rath, S.K. Early stage software effort estimation using random forest technique based on use case points. *Inst. Eng. Technol. Softw.* **2016**, *10*, 10–17. [CrossRef]
28. Azzeh, M.; Nassif, A.B. A hybrid model for estimating software project effort from Use Case Points. *Appl. Soft Comput.* **2016**, *49*, 981–989. [CrossRef]
29. Project Estimation Using Use Case Metrics Tutorial in Enterprise Architect | Sparx Systems. Available online: https://sparxsystems.com/resources/tutorials/use-case-metrics.html (accessed on 12 July 2023
30. Azzeh, M.; Nassif, A.B.; Elsheikh, Y.; Angelis, L. On the value of project productivity for early effort estimation. *Sci. Comput. Program.* **2022**, *219*, 102819. [CrossRef]
31. Azzeh, M.; Bou Nassif, A.; Attili, I.B. Predicting software effort from use case points: A systematic review. *Sci. Comput. Program.* **2021**, *204*, 102596. [CrossRef]
32. Webster, F.M. The WBS. *PM Netw.* **1994**, *8*, 40–46.
33. Luby, R.E.; Peel, D.; Swahl, W. Component-based work breakdown structure (CBWBS). *Proj. Manag. J.* **1995**, *26*, 38–43.
34. Batyuk, A.; Voityshyn, V. Software Architecture Design of the Information Technology for Real-Time Business Process Monitoring. *Econtechmod Int. Q. J. Econ. Technol. Model. Process.* **2018**, *7*, 13–22.
35. Batyuk, A.; Voityshyn, V. Process mining-based information technology for operational support of software projects estimation. In Proceedings of the XVI International Scientific Conference on Intellectual Systems of Decision-Making and Problems of Computational Intelligence (ISDMCI'2020), Kherson, Ukraine, 25–29 May 2020; pp. 9–11.
36. Saaty, T.L. How to make a decision: The analytic hierarchy process. *Eur. J. Oper. Res.* **1990**, *48*, 9–26. [CrossRef]
37. Kelley, J.E.; Walker, M.R. Critical-Path Planning and Scheduling. In Proceedings of the Eastern Joint IRE-AIEE-ACM Computer Conference, Boston, MA, USA, 1–3 December 1959; Association for Computing Machinery: New York, NY, USA, 1959; pp. 160–173. [CrossRef]
38. Trietsch, D.; Baker, K.R. PERT 21: Fitting PERT/CPM for use in the 21st century. *Int. J. Proj. Manag.* **2012**, *30*, 490–502. [CrossRef]