Галузь знань – «Інформаційні системи і технології»

Шифр: **«ТРИВАЛІСТЬ ЗАДАЧ»**

ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ ЗАДАЧ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ AGILE МЕТОДОЛОГІЇ

2019 – 2020 н.р.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних реаліях, коли розробка програмних продуктів ведеться в умовах значно обмеженого часу та фінансових ресурсів, максимально точне оцінювання необхідних трудових витрат на розробку є одною з основних проблем при управлінні проектами.

Існуючі моделі та методи для оцінювання трудомісткості використовують різні фактори (кількість рядків коду та функціональних точок, кількість сторінок програмно-технічної та користувацької документації, досвід членів команди, та ін.) [1–5]. Через особливості розробки програмних проектів в рамках методології Agile важко за короткий термін та в повному обсязі отримати інформацію для оцінювання трудомісткості розробки на основі деяких з перелічених факторів (наприклад, кількості сторінок користувацької документації, кількості рядків коду та функціональних точок), оскільки ця інформація може бути недоступною на ранніх етапах розробки проекту [6, 7]. Ці моделі також не враховують час, який витрачається на перевірку знайдених дефектів. Крім того, більшість існуючих моделей є лінійними, а тому не враховують нелінійні зв’язки між факторами. Саме тому виникає необхідність розробки нелінійної регресійної моделі оцінювання трудомісткості розробки програмного забезпечення, яке створюється за Agile методологією, на основі нормалізуючого перетворення Джонсона. Ця модель дасть можливість підвищити достовірність оцінювання часу розробки, що робить задачу актуальною.

Мета і завдання дослідження. Метою роботи є підвищення достовірності оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі методи та моделі для оцінювання тривалості виконання задач з розробки програмного забезпечення;

– обґрунтувати необхідність удосконалення моделі для оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології;

– розробити удосконалену регресійну модель для оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології із застосуванням нормалізуючих перетворень;

– розробити програмне забезпечення для оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології, використовуючи розроблену регресійну модель.

Об’єктом дослідження є процес оцінювання тривалості розробки програмного забезпечення, яке створюються за Agile методологією.

Предметом дослідження є нелінійна регресійна модель оцінювання тривалості розробки програмного забезпечення, яке створюються за Agile методологією.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики, математичного моделювання, регресійного аналізу, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в удосконаленні однофакторної нелінійної регресійної моделі для оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології за рахунок використання одновимірного нормалізуючого перетворення Джонсона, що дозволило підвищити достовірність оцінювання тривалості виконання задач в порівнянні з існуючими моделями COCOMO та ISBSG.

Практичне значення отриманих результатів. Програма для оцінювання тривалості розробки програмного забезпечення з використанням Agile методології, яка розроблена в рамках дослідження, дозволить автоматизувати та скоротити час відповідних розрахунків.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ ПРОГРАМНИХ ПРОЕКТІВ З ВИКОРИСТАННЯМ AGILE МЕТОДОЛОГІЇ

1.1 Особливості розробки програмного забезпечення з використанням Agile методології

Agile розробка, або гнучка розробка програмного забезпечення (ПЗ) – це сімейство методологій розробки, яке базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами. Основними характеристиками Agile є:

– мінімізація ризиків;

– розробка базується на коротких циклах (ітераціях);

– увага акцентована на спілкуванні між членами команди.

Ітерації в Agile зазвичай тривають один-два тижні, та мають вигляд програмного проекту в мініатюрі, який включає в себе всі завдання, які потрібно виконати, щоб отримати готовий до випуску наприкінці кожної ітерації програмний продукт: планування, аналіз вимог, проектування, кодування, тестування, документування.

Список завдань потрапляє до ітерації і є мірою її завершеності. Кожна задача перед дим як потрапить до ітерації проходить стадію оцінювання, під час якої визначається скільки необхідно витратити часу для її виконання.

Особливість розробки Agile проектів – це направленість на співпрацю з клієнтом та готовність до змін. Це означає, що в рамках коротких ітерацій вимоги до ПЗ можуть змінюватись, а тривалість ітерації не дозволяє витрачати час на підтримання проектної документації та технічного завдання в актуальному стані. Маніфест Agile говорить про те, що основною метою розробки є робочий продукт, а не вичерпна документація.

Протягом усього процесу розробки важливу роль відіграє щоденна комунікація між розробниками та представниками бізнесу. Це означає, що кожен день необхідно мати актуальну інформацію про стан готовності програмного забезпечення.

Враховуючи все вищезазначене, можна виділити дві основні проблеми розробки ПЗ з використанням Agile методології:

– неможливість одразу сформувати вичерпний та актуальний набір варіантів використання через постійні зміни вимог продукту;

– необхідність швидко та чітко визначити, скільки часу та ресурсів використати для розробки в умовах коротких та інтенсивних ітерацій, щоб мати можливість якомога скоріше надати бізнесу інформацію про стан розробки ПЗ.

1.2 Аналіз існуючих моделей оцінювання тривалості розробки програмного забезпечення

COCOMO

Методика COCOMO дозволяє оцінити трудомісткість та час розробки програмного продукту. Модель складається з ієрархії трьох послідовно уточнюючих рівнів [3]. На кожному рівні всі проекти розбиваються на три групи для різних типів проектів за рівнем складності:

– органічний тип (organic projects);

– напів-розділений тип (semidetached projects);

– вбудований тип (embedded projects).

Органічний тип характеризується тим, що проект виконується невеликою групою фахівців, що мають досвід у створенні подібних проектів і досвід застосування технологічних засобів. Умови роботи стабільні, і проект має відносно невисоку складність.

Напів-розділений тип займає проміжне положення між органічним і вбудованим – це проекти середньої складності. Виконавці знайомі лише з

деякими характеристиками (або компонентами) створюваної системи, мають середній досвід роботи з подібними проектами, проект має елемент новизни. Тільки частина вимог до проекту жорстко фіксується, в інших аспектах розробки є ступені вибору.

Вбудований тип характеризується дуже жорсткими вимогами на програмний продукт, інтерфейси, параметри комп`ютера. Як правило, у таких виробів висока ступінь новизни і планування робіт здійснюється за недостатньої інформації, як про самому виробі, так і про умови роботи. Вбудований проект вимагає великих витрат на зміни і виправлення.

Тип тієї чи іншої групи можна розглядати як один з параметрів моделі COCOMO.

Базовий рівень (Basic COCOMO)

Модель цього рівня – двопараметрична. В якості параметрів виступають тип проекту і обсяг програми (число рядків коду).

Рівняння базового рівня моделі мають вигляд:

$$PM = a_i * (SIZE)^{b_i}$$

$$TM = c_i * (PM)^{d_i},$$

$$SS = \frac{PM}{TM},$$

$$P = \frac{SIZE}{PM},$$

де *PM* (People × Month) – трудомісткість (чол. × міс.);

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code – KSLOC);

TM (Time at Month) – час розробки в календарних місяцях;

SS – середня чисельність персоналу;

P – продуктивність.

Коеффіцієнти a_i, b_i, c_i та d_i вибираються з відповідних таблиць в залежності від типу проекту.

Модель цього рівня підходить для ранньої швидкої приблизної оцінки витрат, але її точність дуже низька, тому що не враховуються такі чинники, як кваліфікація персоналу, характеристики обладнання, досвід застосування сучасних методів розробки програмного забезпечення і сучасних інструментальних середовищ розробки та ін.

Середній рівень (Intermediate COCOMO)

На цьому рівні базова модель уточнена за рахунок введення додаткових 15 «Атрибутів вартості» (або факторів витрат) Cost Drivers (CD_k), які згруповані за чотирма категоріями. Значення кожного атрибута також вибирається з відповідних таблиць згідно з його ступенем значущості (рейтингом) в конкретному проекті.

Рівняння середнього рівня моделі має вигляд:

$$PM = EAF * a_i * (SIZE)^{b_i},$$

де *PM* (People × Month) – трудомісткість (чол. × міс.);

EAF (Effort Adjustment Factor) – добуток обраних атрибутів з відповідних таблиць;

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (Kilo of Source Line of Code – KSLOC).

Коефіцієнти моделі a_i та b_i вибираються з відповідної таблиці в залежності від типу проекту. Час розробки розраховується за формулою (1.2).

Детальний рівень (Advanced COCOMO)

Підвищує точність оцінки за рахунок ієрархічної декомпозиції створюваного ПЗ і обліку вартісних факторів на кожному рівні ієрархії і за фазами робіт.

COCOMO II

У 1997 році методика COCOMO була вдосконалена і отримала назву COCOMO II. Калібрування параметрів проводилася по 161 проекту розробки ПЗ. Розрізняються дві стадії оцінки проекту: попередня оцінка на початковій фазі (Early Design) і детальна оцінка після опрацювання архітектури (Post Architecture).

Формула оцінки трудомісткості проекту в чол.×міс. має вигляд:

$$PM = EAF * A * (SIZE)^E,$$

де *EAF* (Effort Adjustment Factor) – добуток обраних множників трудомісткості: $EAF = \prod_{k=1}^{n} EM_k$;

EMj – множники трудоемкості (Effort Multiplier), кількість і значення яких відрізняються для різних стадій оцінки проекту (сім множників для стадії попередньої оцінки та сімнадцять множників для стадії детальної оцінки після опрацювання архітектури) та вибирається з відповідних таблиць;

A=2,94 для попередньої оцінки та *A*=2,45 для детальної оцінки;

SIZE – обсяг програмного продукту в тисячах рядків вихідного тексту (KSLOC – Kilo of Source Line of Code);

$E = B + 0.01 * \sum_{j=1}^{5} SF_j$, *B*=0,91;

SF_j – фактори масштабу (Scale Factors), які вибираються з відповідної таблиці, у методиці COCOMO II використовуються п'ять факторів масштабу (прецедентність, наявність досвіду аналогічних розробок; гнучкість процесу розробки; архітектура і дозвіл ризиків; спрацьованість команди; зрілість процесів), а також шість рівнів впливу факторів (дуже низький; низький; середній; високий; дуже високий; критичний), які застосовуються на обох стадіях оцінки проекту.

Оцінка тривалості проекту

Час розробки проекту *TM* в методиці COCOMO II для обох рівнів розраховується за формулою:

$$TM = SCED * C * (PM_{NS})^{D+0.2*(E-B)},$$

де *SCED* – множник, що визначає стиснення розкладу;

C=3,67; *D*=0,28;

PM_{NS} – розрахована трудомісткість проекту без урахування множника *SCED*.

Інші параметри визначені вище.

З врахуванням того, що один людино-місяць складається з 152 людино-години, для попереднього оцінювання тривалості робіт в залежності від трудомісткості програмних проектів можуть бути використані наступні розрахункові формули

– для органічного типу проектів

$$D = 0{,}371 \cdot E^{0{,}38}, \quad (1.1)$$

– для напів-розділеного типу проектів

$$D = 0{,}431 \cdot E^{0{,}35}, \quad (1.2)$$

– для вбудованого типу проектів

$$D = 0{,}501 \cdot E^{0{,}32}, \quad (1.3)$$

де *D* – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Модель ISBSG – регресійна модель для оцінювання тривалості робіт в проектах з розробки програмного забезпечення. Як фактор в цих регресійних моделях було використано трудомісткість програмних проектів:

$$D = 0{,}662 \cdot E^{0{,}328}, \tag{1.4}$$

де D – тривалість проекту (в місяцях);

E – трудомісткість проекту (в людино-годинах).

Закон розподілу емпіричних даних тривалості робіт в програмних проектах та трудомісткості цих проектів зазвичай не є нормальним. Крім того, регресійні моделі COCOMO та ISBSG не дозволяють виконувати оцінювання довірчих інтервалів тривалості робіт.

1.3 Способи удосконалення регресійних моделей оцінювання тривалості розробки програмного забезпечення

В загальному вигляді регресійна модель може бути представлена рівнянням:

$$y = \bar{y} + \varepsilon_t = f(x) + \varepsilon_t; \tag{1.5}$$

де y – залежна змінна, або результативна ознака;

$f(x)$ – функція, яка визначає вид регресійної моделі: нелінійна або лінійна;

x – незалежна змінна, або фактор;

ε_t – випадкова помилка, або збурення.

Нелінійні регресійні моделі можуть бути двох видів:

– нелінійні відносно незалежних змінних, але лінійні по оцінюваним параметрам, наприклад, поліноми різних ступенів, гіперболічні функції;

– нелінійні по оцінюваним параметрам, наприклад, степенева, показова, експоненціальна функції.

Однак при побудові подібних регресійних моделей виникає ряд труднощів:

– якщо випадкові величини, що входять до регресійної моделі, не підпорядковуються нормальному закону розподілу, це призводить до нелінійної регресії;

– якщо залежна випадкова величина залежить одночасно від двох і більше факторів, що зазвичай і буває при вирішенні практичних завдань, отримуємо множинну регресію.

Для побудови нелінійних регресійних моделей існує кілька методів: метод простого перебору, метод лінеаризуючих перетворень, метод нормалізуючих перетворень.

Метод простого перебору, або метод усіх можливих регресій, вимагає завдання різних видів рівняння регресії і вибору найкращого наближення із заданих за певним критерієм. Це вимагає досить великої кількості обчислень і не завжди призводить до отримання найкращого рішення. Всі ці недоліки роблять даний метод неефективним [8].

Метою *методу лінеаризуючих перетворень* є перехід від нелінійної регресії до лінійної шляхом заміни вихідних змінних і коефіцієнтів, проте не завжди можливо підібрати таку заміну [9–11]. Крім того, така заміна призводить до спрощення регресійної моделі і деякої втрати інформації, пов'язаної з нелінійністю.

Метою *методу нормалізуючих перетворень* є пошук перетворень, за допомогою яких можна здійснити перехід від вихідних негаусівських випадкових величин до гаусівських. Для отриманих гаусівських випадкових величин будують лінійне рівняння регресії, яке далі перетворять на нелінійне рівняння за допомогою раніше знайдених нормалізуючих перетворень [10].

У разі нормального закону розподілу випадкових величин можна побудувати лінійну регресійну модель:

$$z_y = b_1 z_x + b_0 + \varepsilon; \quad (1.6)$$

де b_1, b_0 – коефіцієнти лінійної регресії, які знаходяться методом найменших квадратів:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - \left(\sum_{i=1}^{n} z_{xi}\right)^2};$$

$$b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n};$$

ε – випадкова величина, розподілена за нормальним законом, що визначається як ε~N(0,1).

Для оцінювання точності побудови рівняння регресії намагаються знайти його довірчий інтервал. У випадку нормального закону розподілу випадкових величин довірчий інтервал лінійного рівняння регресії можливо побудувати традиційним методом з використанням t-розподілу Стьюдента [12]:

$$y = \hat{y} \pm t(_{\alpha/2,n-2}) \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}; \quad (1.7)$$

де $\hat{y}$ – значення y, розраховане за рівнянням регресії;

$t(_{\alpha/2,n-2})$ – квантіль t-розподілу Стьюдента;

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

$$S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\ .$$

Інтервал прогнозування лінійного рівняння регресії у даному випадку можна побудувати за наступною формулою:

$$y = \hat{y} \ \pm t_{(\alpha/2,n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}, \qquad (1.8)$$

де $\hat{y}$ – значення y, розраховане за рівнянням регресії;

$t_{(\alpha/2,n-2)}$ – квантіль t-розподілу Стьюдента;

α – рівень значущості;

n – кількість значень випадкових величин у вибірці;

$$S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\ .$$

Оскільки використання методу нормалізуючих перетворень для рішення аналогічних задач дає гарні результати [13–15], це дозволяє застосувати даний метод для удосконалення регресійної моделі оцінювання тривалості розробки ПЗ, яке створюються за Agile методологією.

2 УДОСКОНАЛЕННЯ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ТРИВАЛОСТІ ВИКОНАННЯ ЗАДАЧ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ AGILE МЕТОДОЛОГІЇ

2.1 Регресійна модель для оцінювання тривалості виконання задач з розробки програмного забезпечення з використанням Agile методології

Розподіл емпіричних даних трудомісткості та часу виконання задач з розробки програмного забезпечення відрізняється від нормального закону розподілу. Тому для побудови нелінійної регресійної моделі будемо використовувати нормалізуючи перетворення для отримання нормалізованих значень трудомісткості та часу виконання задач з розробки ПЗ.

Сім'ї розподілів Джонсона отримані шляхом перетворення нормованої нормально розподіленої випадкової величини. У загальному випадку перетворення має вигляд [16]:

$$z = \gamma + \eta h(x, \varphi, \lambda);\ \ \eta > 0; -\infty < \gamma < \infty; \lambda > 0; -\infty < \varphi < \infty; \tag{2.1}$$

де z – нормована нормально розподілена випадкова величина;

γ, η, φ, λ – параметри перетворення Джонсона, з яких γ і η – параметри форми, φ – параметр зсуву, λ – параметр масштабу;

x – випадкова величина, яка нормалізується;

h – функція певної сім'ї:

$$h_1(x, \varphi, \lambda) = \ln(\tilde{x}), x > \varphi; \tag{2.2}$$

$$h_2(x, \varphi, \lambda) = \ln\left(\frac{\tilde{x}}{1-\tilde{x}}\right), \varphi < x < \varphi + \lambda; \tag{2.3}$$

$$h_3(x,\varphi,\lambda) = Arsh(\tilde{x}), -\infty \le x \le +\infty. \quad (2.4)$$

Сім'ї функцій h_1 відповідає логарифмічно нормальний розподіл S_L Джонсона, сім'ї функцій h_2 відповідає сім'я розподілів S_B Джонсона, сім'ї функцій h_3 відповідає сім'я розподілів S_U Джонсона, $\tilde{x} = \frac{x-\varphi}{\lambda}$.

Перетворення (2.1) має зворотне перетворення:

$$x = \varphi + \lambda h^{-1}(z,\gamma,\eta);\ \ \eta > 0; -\infty < \gamma < \infty; \lambda > 0; -\infty < \varphi < \infty; \quad (2.5)$$

де x – випадкова величина з розподілом Джонсона;

γ, η, φ, λ – параметри перетворення Джонсона;

z – нормально розподілена випадкова величина;

h^{-1} – функції певної сім'ї:

$$h_1^{-1}(z,\gamma,\eta) = e^{\zeta}; \quad (2.6)$$

$$h_2^{-1}(z,\gamma,\eta) = \frac{1}{1+e^{-\zeta}}; \quad (2.7)$$

$$h_3^{-1}(z,\gamma,\eta) = \frac{e^{\zeta}-e^{-\zeta}}{2}. \quad (2.8)$$

Функція h_1^{-1} – для сім'ї S_L Джонсона, функція h_2^{-1} – для сім'ї S_B Джонсона, функція h_3^{-1} – для сім'ї S_U Джонсона, $\zeta = \frac{z-\gamma}{\eta}$.

Для вибору конкретної сім'ї розподілу Джонсона використовують або діаграму в площині ексцес – асиметрія в квадраті $\varepsilon - A^2$, або аналітичну залежність $\varepsilon(A^2)$. Для автоматизації розрахунків та при розробці ПЗ зручніше використовувати аналітичну залежність, наведену в [17]:

$$\varepsilon(A^2) = 3{,}59 \cdot 10^{-6} A^8 - 4{,}8805 \cdot 10^{-4} A^6 + 4{,}1655 \cdot 10^{-2} A^4 + \\ + 1{,}8203 A^2 + 2{,}9658 \quad (2.9)$$

Якщо точка з координатами (A^2, ε) знаходиться біля лінії S_L, то можна використовувати розподіл з сім'ї S_L. Якщо точка з координатами (A^2, ε) знаходиться вище лінії S_L, то можна використовувати розподіл з сім'ї S_U, а якщо нижче лінії S_L до лінії критичної області – то розподіл з сім'ї S_B. Якщо ж точка потрапляє у критичну область, то використовувати для апроксимації сім'ї розподілів Джонсона не можна.

2.2 Оцінка адекватності регресійної моделі

Для перевірки адекватності лінійного рівняння регресії використаємо коефіцієнт детермінації R^2:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \quad (2.10)$$

де y_i – емпіричне значення y;

$\hat{y}_i$ – розрахункове значення y;

$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ – середнє значення випадкової величини y.

R^2 характеризує частку дисперсії, яка обумовлена регресією, в загальній дисперсії показника y. Коефіцієнт детермінації R^2 приймає значення від 0 до 1. Чим ближче значення коефіцієнта до 1, тим тісніше зв'язок результативної ознаки з досліджуваними факторами.

При значенні $R^2 \geq 0{,}5$ можна вважати, що дана модель є прийнятною. Достатньо ефективною та результативною можна вважати модель з показником

детермінації $R^2 \geq 0{,}8$. Якщо ж $R^2=1$, то лінія регресії точно відповідає усім спостереженням та вимогам, а модель можна вважати адекватною та достовірною.

Величина коефіцієнта детермінації виступає важливим критерієм оцінки якості лінійних і нелінійних моделей. Чим вагоміша частка пояснюваної варіації, тим менше роль інших факторів, а отже, модель регресії краще апроксимує вихідні дані і такою регресійної моделлю можна скористатися для прогнозу значень результативного показника.

Величину відносної похибки MRE (Magnitude of Relative Errors) для лінійного та нелінійного рівнянь регресії можна знайти за формулою:

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad (2.11)$$

де $\hat{y}$ – значення y, розраховане за рівнянням регресії;

y_i – фактичне значення випадкової величини y.

MMRE (Mean of Magnitude of Relative Errors – середня величина відносної похибки) можна розрахувати за формулою:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \, . \quad (2.12)$$

Рівень прогнозування PRED(l) можна розрахувати за формулою:

$$PRED(l) = \frac{k}{N}, \quad (2.13)$$

де k – кількість значень з $MRE \leq l$.

Таким чином, маємо ряд показників для перевірки адекватності регресійних моделей та порівняння їх між собою.

2.3 Перевірка вихідних емпіричних даних на викиди

Одним з варіантів перевірки даних на викиди є перевірка за допомогою квадрата відстані Махаланобіса (Mahalanobis squared distance), який знаходиться за формулою:

$$d_i^2 = (z_i - \bar{z})^T S_n^{-1} (z_i - \bar{z}) \tag{2.14}$$

де $\bar{z}$ – середній вектор вибірки;

S_N – матриця кореляції вибірки, яка визначається за формулою:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (z_i - \bar{z})(z_i - \bar{z})^T .$$

Точка, що має найбільшу відстань Махаланобіса до решти безлічі заданих точок, вважається що має найбільшу значимість, так як вона має найбільший вплив на кривизну і на коефіцієнти рівняння регресії.

Розрахувавши d_i^2, можна використати або критерій Пірсона χ^2, або критерій Фишера *F*. При використанні критерію Фишера *F* потрібно додатково розрахувати T_S (Test statistic). T_S для d_i^2 може бути розрахований таким чином:

$$Ts = \frac{(N-m)Nd_i^2}{(N^2-1)m}, \tag{2.15}$$

який має апроксимований розподіл *F* з m та $N - m$ ступенями свободи.

У данній роботі будемо використовувати критерій Фишера *F*. Тестова статистика для квадрату відстані Махаланобіса порівнюється з квантилем

розподілу F, який позначається як $F_{m,N-m,\alpha}$. Тут α – це рівень значущості, який у нашому випадку дорівнюватиме 0,05. Точки, для яких значення T_S, розраховане за формулою (2.15), буде більше, ніж квантиль розподілу F вважаються викидами, і ці значення видаляються з набору даних.

Після усунення викидів отриманий новий набір значень знову нормалізується, використовуючи перетворення (2.1) та для нього знову знаходиться квадрат відстані Махаланобіса за формулою (2.14) та тестова статистика за формулою (2.15). Процедура повторюється до того, поки всі значення T_S не будуть менше або дорівнюватимуть квантилю розподілу F.

2.4 Побудова удосконаленої однофакторної нелінійної регресійної моделі для оцінювання тривалості розробки програмного забезпечення з використанням Agile методології

Для побудови удосконаленої нелінійної регресійної моделі в якості вихідних емпіричних даних були розглянуті 886 задач розробки ПЗ з використанням Agile методології. В якості випадкової величини x бралася трудомісткість в годинах; випадкової величини y – час виконання задач в місяцях (перетворення годин в місяці проведено для порівняння з моделями COCOMO та ISBSG).

Для продовження роботи необхідно переконатися в якості вихідних емпіричних даних та відсутності викидів, тому що наявність викидів може бути причиною того, що дані не розподіляються за нормальним законом. Перевірка на наявність викидів проводилася згідно з методикою, описаною у підрозділі 2.3, ітераційно.

Після перевірки вихідних емпіричних даних на викиди за 3 ітерації було видалено 125 випадків, скоригована таким чином вибірка склала 761 задачу.

Ймовірнісні характеристики вибірки емпіричних даних: n=761; $\hat{m}_X = 115{,}0920$, $\hat{D}_X = 29413{,}80$, $\hat{\sigma}_X = 171{,}5045$, $\hat{A}_X = 2{,}5863$, $\hat{\varepsilon}_X = 10{,}1742$; $\hat{m}_y = 0{,}1787$, $\hat{D}_y = 0{,}0140$, $\hat{\sigma}_y = 0{,}1185$, $\hat{A}_y = 1{,}5307$, $\hat{\varepsilon}_y = 5{,}1706$.

За значеннями оцінок A^2 і ε з урахуванням області визначення сімей розподілів Джонсона (2.9) була обрано сім'я розподілів S_B для випадкових величин X та Y.

У відповідності з перетворенням (2.1) для сім'ї розподілів Джонсона S_B (2.3) була виконана нормалізація випадкової величини X, в результаті якої отримані значення нормованої нормально розподіленої випадкової величини Z_X та випадкової величини Y, в результаті якої отримані значення нормованої нормально розподіленої випадкової величини Z_Y.

Було знайдено наступні параметри перетворення Джонсона: γ_X=3,1023, η_X=0,5724, ϕ_X=0,4538, λ_X=10925,91; γ_Y=3,6796, η_Y=0,8486, ϕ_Y=0,05, λ_Y=6,5045.

Тепер будуємо лінійне рівняння регресії для нормалізованих даних згідно з (1.6) та за допомогою методу найменших квадратів знаходимо його коефіцієнти: b_1 = 0,2005, b_0 = 0,000047. Лінійна регресійна модель має вигляд:

$$Z_y = 0{,}2005\, Z_x + 0{,}000047 + \varepsilon.$$

Для побудови нелінійного рівняння регресії використаємо вже побудоване лінійне рівняння регресії та зворотне нормалізуюче перетворення Джонсона (2.5) з відповідною сім'єю S_B:

$$y = \frac{e^c(\lambda_y + \varphi_y) + \varphi_y}{1 + e^c},$$

де: $c = \frac{1}{\eta_y} \cdot \left(b_1 \left[\gamma_x + \eta_x \ln\left(\frac{x - \varphi_x}{\lambda_x + \varphi_x - x} \right) \right] + b_0 - \gamma_y \right).$

Остаточно нелінійне рівняння регресії має вигляд:

$$y = \frac{e^{c} \cdot 6{,}5544 + 0{,}05}{1 + e^{c}},$$

де $c = 0{,}1352 \cdot \ln\left(\frac{x - 0{,}4538}{6{,}9582 - x}\right) - 3{,}6031$.

(1-α)% довірчий інтервал нелінійного рівняння регресії можна побудувати, використовуючи лінійне рівняння регресії, *t*-розподіл Стьюдента та зворотне нормалізуюче перетворення Джонсона (2.5) з відповідною сім'єю S_B:

$$y = \frac{e^{k_1}(\lambda_y + \varphi_y) + \varphi_y}{1 + e^{k_1}},$$

де: $k_1 = \frac{1}{\eta_y} \cdot \left(b_1 \cdot z_x + b_0 - \gamma_y \pm t(\alpha/2, n-2) \cdot S_{z_y} \cdot \sqrt{\frac{1}{n} + \frac{(z_x - \bar{z}_x)^2}{\sum_{i=1}^{n}(z_{xi} - \bar{z}_x)^2}} \right),$

$z_x = \gamma_x + \eta_x \ln\left(\frac{x - \phi_x}{\lambda_x + \phi_x - x}\right).$

Аналогічно знайдемо і інтервал прогнозування нелінійного рівняння регресії. Перехід до (1-α)% інтервалу прогнозування нелінійного рівняння регресії можна здійснити, використовуючи побудований (1-α)% інтервал прогнозування та зворотне перетворення Джонсона (2.5) з відповідною сім'єю S_B.

На рисунку 2.1 зображено графік побудованого рівняння регресії з довірчими інтервалами та інтервалами передбачення.

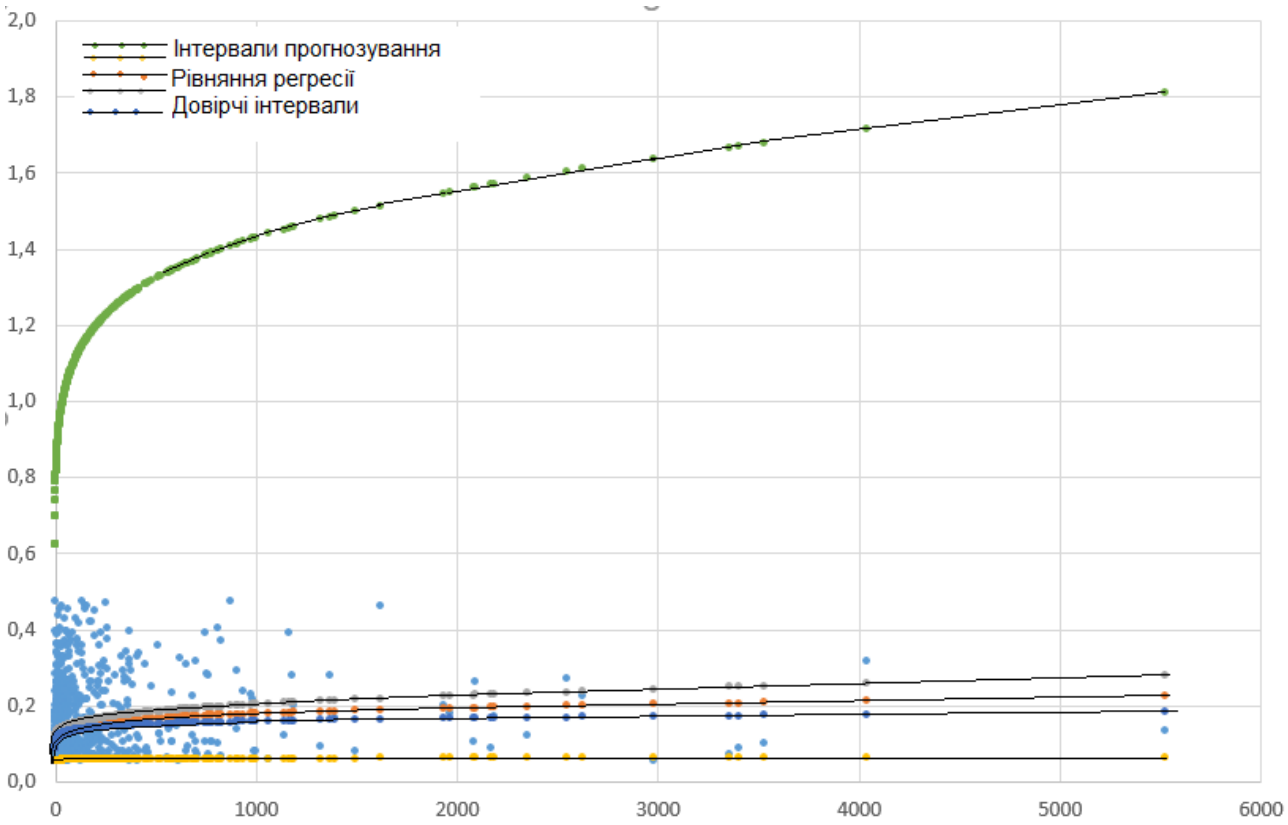

Рисунок 2.1 – Рівняння регресії з довірчими інтервалами та інтервалами передбачення

Значення коефіцієнту детермінації R^2, знайдене за формулою (2.10), MMRE, знайдене за формулою (2.12), та значення Pred(0,25), знайдене за формулою (2.13), наведені в табл. 2.1.

Порівняємо наведені параметри із аналогічними, які отримані за результатами розрахунків з використанням моделей COCOMO за формулою (1.1) та ISBSG за формулою (1.4), результати також наведені в табл. 2.1.

Таблиця 2.1 – Оцінка адекватності регресійних моделей

	Нелінійне рівняння регресії з використанням нормалізуючого перетворення Джонсона	Модель COCOMO	Модель ISBSG
R^2	0,6460	0,4883	0,1598
MMRE	0,2063	0,3415	0,4009
Pred(0,25)	0,7698	0,4567	0,1133

Як видно із наведених даних, рівняння, побудоване із використанням нормалізуючого перетворення Джонсона, має кращі значення параметрів оцінки адекватності регресійних моделей. Отримане нелінійне рівняння регресії з використанням нормалізую чого перетворення Джонсона можна вважати прийнятним.

Порівняємо значення для границь довірчого інтервалу нелінійного рівняння регресії та границь інтервалу прогнозування нелінійного рівняння регресії, побудованого із використанням нормалізуючого перетворення Джонсона, з аналогічними результатами, які отримані для моделей COCOMO та ISBSG. Зведені результати для границь довірчого інтервалу наведено у таблиці 2.2, для границь інтервалу прогнозування - у таблиці 2.3. У таблицях наведено фрагмент даних.

Таблиця 2.2 – Довірчі інтервали рівнянь регресії (фрагмент даних)

Нелінійне рівняння регресії з використанням нормалізуючого перетворення Джонсона			Модель COCOMO			Модель ISBSG		
верхня границя	нижня границя	довжина	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
0,1699	0,2412	0,07	6,6534	8,8505	2,20	7,6511	10,5999	2,95
0,1469	0,1764	0,03	3,3228	3,6897	0,37	4,3546	4,8472	0,49
0,1478	0,1786	0,03	3,4528	3,8326	0,38	4,5002	5,0099	0,51
0,1518	0,1884	0,04	4,0126	4,4864	0,47	5,1135	5,7493	0,64
0,1101	0,1261	0,02	0,7079	1,0721	0,36	1,1644	1,6532	0,49
0,1284	0,1431	0,01	1,5247	1,8773	0,35	2,2277	2,7010	0,47
0,1253	0,1395	0,01	1,3434	1,6993	0,36	1,9991	2,4768	0,48
0,1299	0,1451	0,02	1,6265	1,9772	0,35	2,3546	2,8253	0,47
0,1299	0,1451	0,02	1,6265	1,9772	0,35	2,3546	2,8253	0,47
0,1369	0,1558	0,02	2,1815	2,5221	0,34	3,0307	3,4880	0,46
0,1208	0,1350	0,01	1,1183	1,4778	0,36	1,7102	2,1926	0,48
0,1361	0,1545	0,02	2,1099	2,4516	0,34	2,9449	3,4036	0,46
0,1233	0,1374	0,01	1,2381	1,5957	0,36	1,8647	2,3447	0,48
0,0938	0,1142	0,02	0,2995	0,6661	0,37	0,5850	1,0770	0,49
0,0978	0,1172	0,02	0,3801	0,7464	0,37	0,7034	1,1950	0,49
0,1030	0,1208	0,02	0,5011	0,8667	0,37	0,8769	1,3677	0,49
0,1064	0,1233	0,02	0,5946	0,9597	0,37	1,0083	1,4983	0,49
0,1078	0,1244	0,02	0,6352	1,0000	0,36	1,0646	1,5542	0,49

Таблиця 2.3 – Інтервали прогнозування рівнянь регресії (фрагмент даних)

Нелінійне рівняння регресії з використанням нормалізуючого перетворення Джонсона			Модель COCOMO			Модель ISBSG		
нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина	нижня границя	верхня границя	довжина
0,0615	1,6365	1,58	2,6049	12,8990	10,29	2,2173	16,0337	13,82
0,0584	1,2762	1,22	-1,5255	8,5381	10,06	-2,1526	11,3544	13,51
0,0585	1,2905	1,23	-1,3893	8,6747	10,06	-1,9988	11,5089	13,51
0,0590	1,3503	1,29	-0,7845	9,2835	10,07	-1,3251	12,1879	13,51
0,0551	0,8537	0,80	-4,1418	5,9217	10,06	-5,3446	8,1623	13,51
0,0565	1,0344	0,98	-3,3305	6,7326	10,06	-4,2889	9,2175	13,51
0,0562	1,0012	0,95	-3,5102	6,5530	10,06	-4,5153	8,9912	13,51
0,0566	1,0519	1,00	-3,2296	6,8334	10,06	-4,1632	9,3430	13,51
0,0566	1,0519	1,00	-3,2296	6,8334	10,06	-4,1632	9,3430	13,51
0,0573	1,1365	1,08	-2,6796	7,3831	10,06	-3,4936	10,0123	13,51
0,0559	0,9556	0,90	-3,7336	6,3297	10,06	-4,8019	8,7047	13,51
0,0572	1,1264	1,07	-2,7506	7,3121	10,06	-3,5787	9,9272	13,51
0,0560	0,9806	0,92	-3,6147	6,4485	10,06	-4,6486	8,8580	13,51
0,0540	0,6983	0,64	-4,5490	5,5146	10,06	-5,9225	7,5845	13,51
0,0542	0,7378	0,68	-4,4686	5,5950	10,06	-5,8043	7,7027	13,51
0,0546	0,7866	0,73	-4,3479	5,7157	10,06	-5,6312	7,8758	13,51
0,0548	0,8189	0,76	-4,2546	5,8089	10,06	-5,5002	8,0068	13,51
0,0549	0,8318	0,78	-4,2141	5,8494	10,06	-5,4441	8,0629	13,51

Як видно із наведених таблиць, нижні границі інтервалу прогнозування, отримані для нелінійного рівняння регресії із використанням нормалізуючого перетворення Джонсона, більші нуля, тоді як ті ж самі границі, отримані за іншими моделями, мають від`ємні значення.

До того ж, довжини довірчого інтервалу та інтервалу прогнозування, отримані без використання нормалізації, більші відповідних довжин, отриманих із використанням нормалізуючого перетворення Джонсона.

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ТРУДОМІСТКОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Ескізний проект

Проектування будь-якого ПЗ розпочинається з розробки ескізного проекту, у якому представляються результати зовнішнього проектування програмного забезпечення.

Діаграма варіантів використання – діаграма, на якій зображено відношення між акторами та прецедентами в системі, наведена на рис. 3.1.

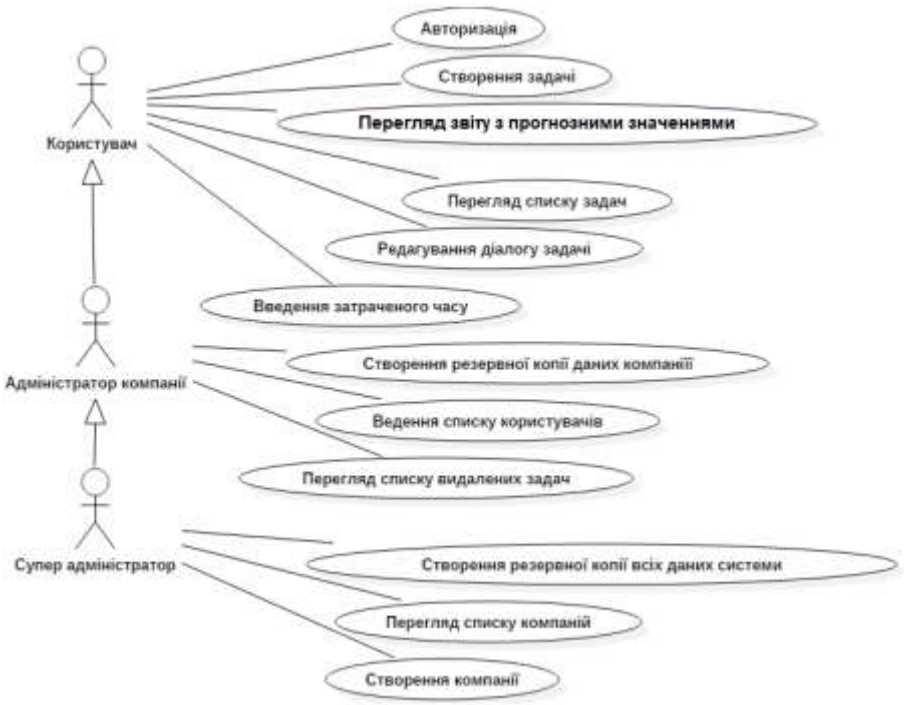

Рисунок 3.1 – Діаграма варіантів використання

Проектування інтерфейсу користувача, е важливим етапом розробки програмного забезпечення. Проектування інтерфейсу допомагає наглядно оцінити результат замовником, а також вдосконалитись в зручності інтерфейсу та перевірка виконуваних функцій. Приклади ескізів деяких екранних форм наведені на рисунках 3.2 – 3.3.

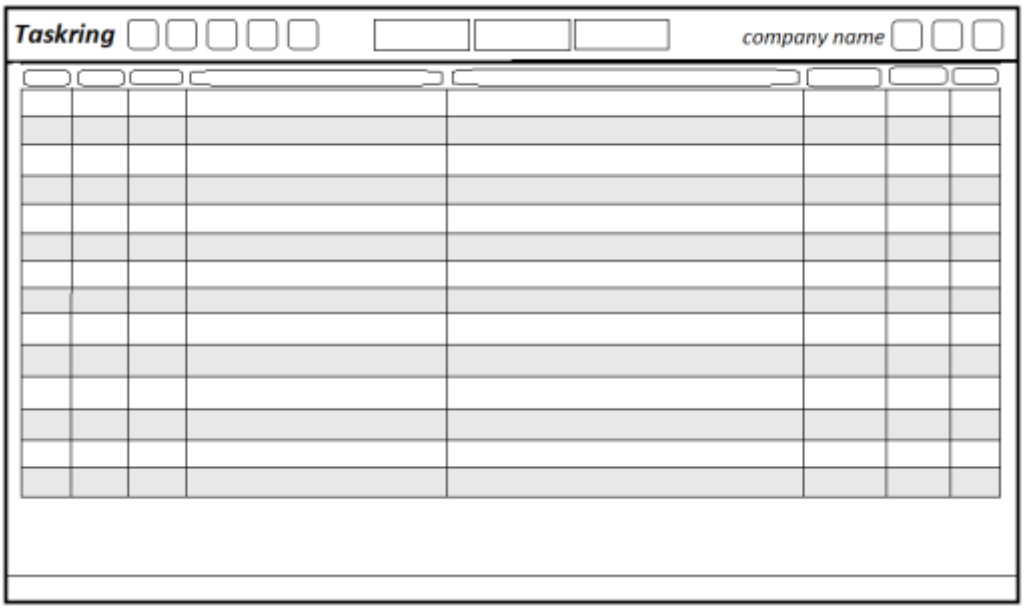

Рисунок 3.2 – Вікно перегляду списку задач

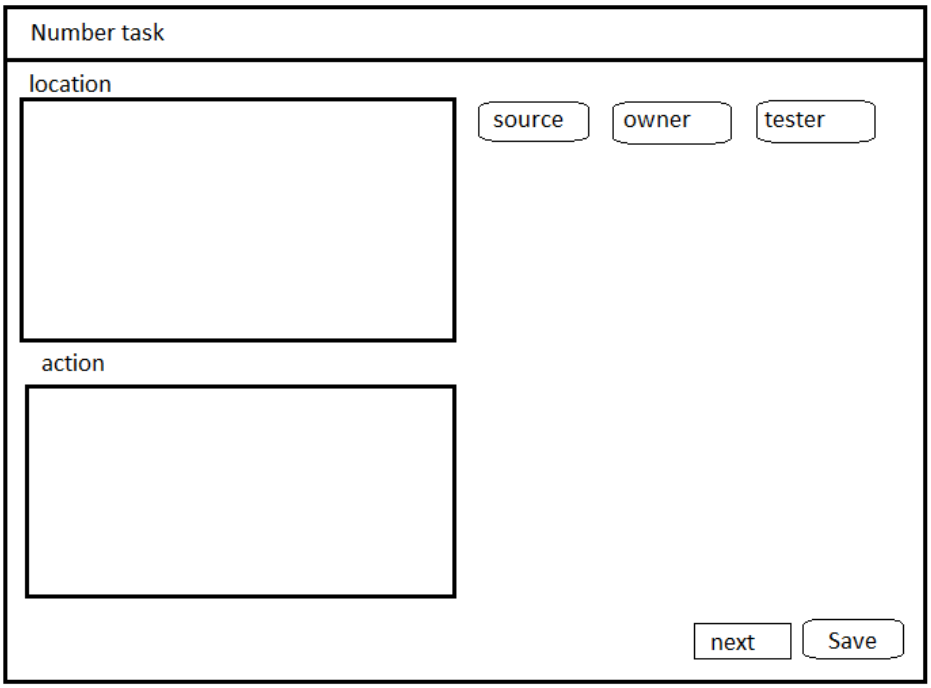

Рисунок 3.3 – Вікно редагування задачі

Концептуальне проектування – побудова семантичної моделі предметної області. Така модель створюється без орієнтації на якусь конкретну СУБД і модель даних. Концептуальна модель (див. рис. 3.4) представлена за допомогою ER-діаграми.

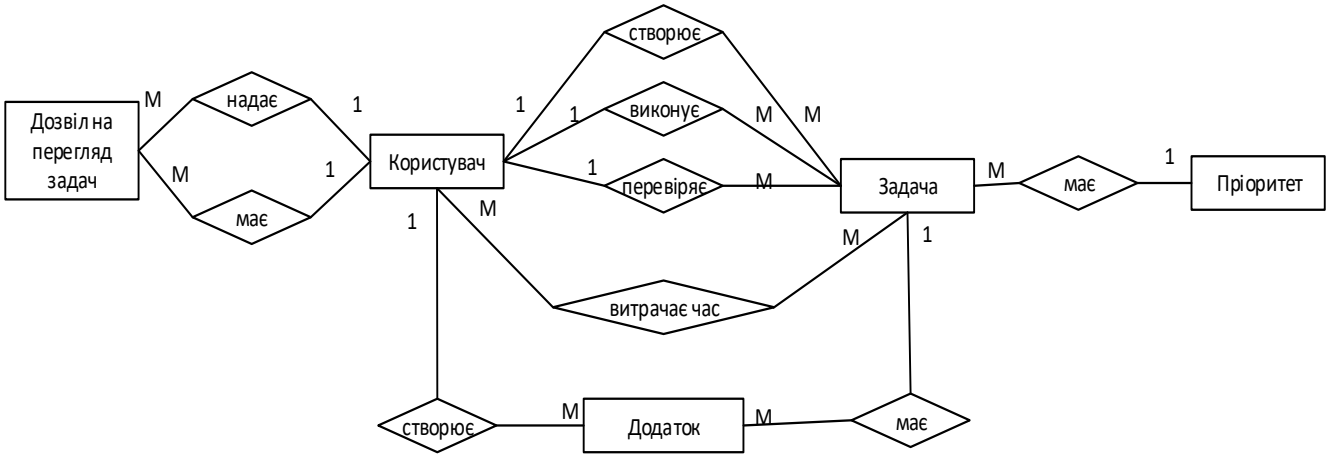

Рисунок 3.4 – Концептуальна модель бази даних

3.2 Технічний проект

Першим етапом технічного проектування є побудова діаграми компонентів системи. Діаграма компонентів необхідна для відображенні основних компонентів та відносин між ними. Наступним етапом є уточнення діаграми компонентів набором діаграм класів. Діаграма класів необхідна для відображенні основних класів та типів відносин між ними. На рисунку 3.5 відображено діаграму класів для модулю розподілення вхідного запиту.

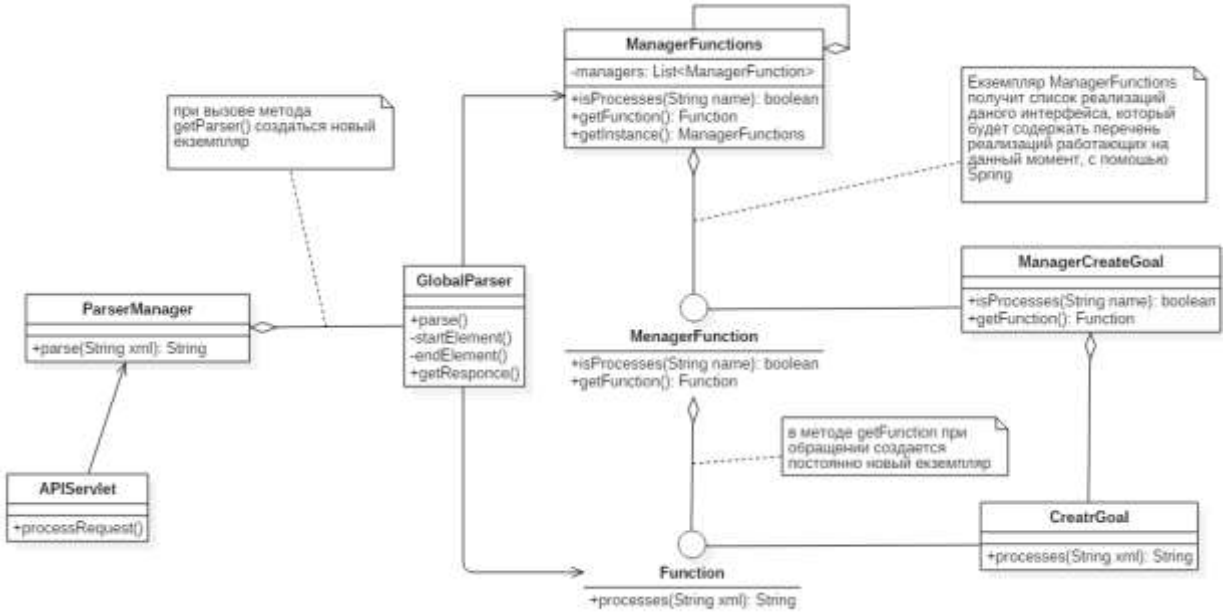

Рисунок 3.5 – Діаграма класів модулю розподілення вхідних запитів

Далі проводиться вибір типу сховища даних і перетворення концептуальної моделі в модель даних відповідно до вибраного типу сховища. Із-за відносної легкості в проектуванні, обсягу інформації, та можливостями, що надають реляційні БД, було вибрано саму цей тип сховища даних, відповідно до якого була розроблена логічна модель БД.

3.3 Робочій проект

Вибір мови програмування було виконано на етапі написання технічного завдання – скриптова мова PHP. Вибір обумовлено тим, що система TaskRing, для якої розробляється підсистема для оцінки тривалості виконання задач , написано мовою PHP версії 5.6.

Середовищем розробки було обрано PHPStorm від JetBrains, тому що ця середа розробки була створена спеціально для мови PHP, та має велику кількість корисних функцій, які прискорюють процес розробки програмного забезпечення. Крім того, PHPStorm входить до множини програмних продуктів компанії IntelliJ, які розповсюджуються за студентською ліцензією, як, наприклад IntelliJ Idea, яка неодноразово використовувалася в процесі навчання для написання програмних продуктів.

При розробці було використано СУБД MySQL. Це обумовлено можливостями, що надає дана СУБД, популярністю та доступністю інформації.

Проектування БД виконувалось в середовищі адміністрування БД dbForge Studio for MySQL – це програмне забезпечення, в якому можна спроектувавши фізичну модель даних та отримати код для конкретної СУБД.

Під час випробувань було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу. Всі виявлені недоліки ПЗ були зафіксовані в протоколах і усунуті до моменту впровадження ПЗ у дію. Випробування було проведено за стратегією «чорного ящика».

ВИСНОВКИ

В даній роботі було вирішене науково-практичне завдання підвищення достовірності оцінювання тривалості виконання задач з розробки ПЗ з використанням Agile методології.

В процесі проведення досліджень одержані наступні результати:

1. Проаналізовано особливості, переваги та недоліки існуючих моделей та методів для оцінювання тривалості виконання задач з розробки ПЗ з використанням Agile методології; визначено задачі, які необхідно вирішити для досягнення мети дослідження. Аналіз показав, що для отримання достовірних результатів необхідно використовувати нелінійні регресійні моделі та нормалізуюче перетворення Джонсона для емпіричних даних трудомісткості задач з розробки ПЗ.

2. Удосконалено нелінійну регресійну модель для оцінювання тривалості виконання задач з розробки ПЗ з використанням Agile методології в залежності від трудомісткості цих задач. Також побудовано рівняння границь довірчих інтервалів та інтервалів прогнозування нелінійної регресії, що дозволяє підвищити достовірність оцінювання тривалості виконання задач з розробки ПЗ з використанням Agile методології.

3. На основі удосконаленої регресійної моделі було розроблено програму «TaskRing» для оцінювання тривалості виконання задач з розробки ПЗ з використанням Agile методології. Був розроблений проект та програмна документація: технічне завдання, опис програми, інструкція користувача, програма та методика випробувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Голованова М.А. Оценка трудоемкости работ на ранних стадиях создания программного обеспечения [Текст] / М.А. Голованова, Е.В. Надин // Системи обробки інформації. – Харків, 2014. – № 8 (124). – С.151-156.

2. Lazic L. Challenges in Estimating Software Testing Effort [Текст] / L. Lazic, I. Đokic, S. Milinkovic // INFOTEH-JAHORINA. – 2014. – №13. – P.637-642.

3. Software Test Estimation – 9 General Tips on How to Estimate Testing Time Accurately [Електронний ресурс] / Режим доступу: http://www.softwaretestinghelp.com/software-test-estimation-how-to-estimate-testing-time-accurately/ – 25.09.2019 р. – Загол. з екрану.

4. Стадии цикла разработки ПО [Електронний ресурс] – Режим доступу: http://qalight.com.ua/baza-znaniy/stadii-tsikla-razrabotki-po/ – 25.09.2019 р. – Загол. з екрану.

5. Сокращение затрат на обеспечение качества программных продуктов. Решение IBM Rational Quality Management [Електронний ресурс] – Режим доступу: https://www-01.ibm.com/software/ru/smb/rational/quality/pdf/reducequalitycost_rrc_ru2.pdf. – 25.09.2019 р. – Загол. з екрану.

6. Agile Testing Process [Електронний ресурс] – Режим доступу: http://www.slideshare.net/ExigenServices/agile-testing-process-26580032 – 25.09.2019 р. – Загол. з екрану.

7. Полезные метрики в Agile: Метрики производительности [Електронний ресурс] – Режим доступу: http://it-study.by/philippov-agile-metrics – 25.09.2019 р. – Загол. з екрану.

8. Дрейпер Н. Прикладной регрессионный анализ: В 2-х кн. Кн. 2 / Пер. с англ. – 2-е изд., перераб. и доп. [Текст] / Н. Дрейпер, Г. Смит – М.: Финансы и статистика, 1987. – 351 с.

9. Айвазян С.А. Прикладная статистика. Основы эконометрики: Учебник для вузов: В 2 т. 2-е изд., испр. – Т. 1: Теория вероятностей и прикладная статистика [Текст] / С.А. Айвазян, В.С. Мхитарян – М.: ЮНИТИ-ДАНА, 2001. – 656 с.

10. Кобзарь А.И. Прикладная математическая статистика. Для инженеров и научных работников [Текст] / А.И. Кобзарь – М.: ФИЗМАТЛИТ, 2006. – 816 с.

11. Chatterjee Samprit. Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Simonoff. – Wiley, 2012. – 240 p.

12. Yan Xin. Linear regression analysis: theory and computing [Text] / Xin Yan, Xiao Gang Su – Singapore: World Scientific Publishing Co. Pte. Ltd., 2009. – 328 p.

13. Приходько С.Б. Інтервальне оцінювання статистичних моментів часу затримок виконання програмних проектів на основі перетворення Джонсона [Текст] / С.Б. Приходько, А.В. Пухалевич // Збірник наукових праць НУК. – Миколаїв: НУК, 2010. – № 2 (431). – С.118-124. – ISSN 2313-0415.

14. Приходько С.Б. Розробка нелінійної регресійної моделі трудомісткості програмних проектів на основі нормалізуючого перетворення Джонсона [Текст] / С.Б. Приходько, А.В. Пухалевич // Радіоелектронні і комп’ютерні системи. – Харків: ХАІ, 2012. – № 4 (56) – С.90-93.

15. Приходько С.Б. Розробка нелінійних регресійних моделей трудомісткості програмних проектів на основі перетворення Джонсона [Текст] / С.Б. Приходько, А.В. Пухалевич // Збірник наукових праць НУК. – Миколаїв: НУК, 2014. – № 2 (2014). – С.76-80.

16. Кендалл М. Теория распределений [Текст] / М. Кендалл, А. Стюарт. – Пер. с англ. под ред. А.Н. Колмогорова. – М.: Наука. Гл. ред. физ.-мат. лит., 1966. – 588 с.

17. Приходько С.Б. Аналитическая зависимость для выбора семейства распределений Джонсона [Текст] / С.Б. Приходько, Л.Н. Макарова, А.С. Приходько // Проблеми інформаційних технологій. – Херсон: ХНТУ, 2016. – №02 (020). – С.105-110.

Додаток А - Текст програми

Лістинг А.1 – Запити на створення таблиць БД відповідно до розробленої фізичної моделі

```
CREATE TABLE user (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  confirmation VARCHAR(50) DEFAULT NULL,
  dateRegistration VARCHAR(50) DEFAULT NULL,
  hashpass VARCHAR(50) NOT NULL,
  mail VARCHAR(150) NOT NULL,
  name VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
)
ENGINE = INNODB
AUTO_INCREMENT = 8
AVG_ROW_LENGTH = 8192
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE acount (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  annitation LONGTEXT NOT NULL,
  name VARCHAR(150) NOT NULL,
  user_id BIGINT(20) NOT NULL,
  type TINYINT(1) NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_e57un1oftohad5mkh5nclybmf FOREIGN KEY (user_id)
    REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 1
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE fixing_acount (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  date VARCHAR(255) NOT NULL,
  user_id BIGINT(20) NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_t4kybkkecx9mk2we1h72r4iww FOREIGN KEY (user_id)
    REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
```

```
ENGINE = INNODB
AUTO_INCREMENT = 1
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE goal (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  annotation VARCHAR(255) DEFAULT NULL,
  endDate VARCHAR(255) DEFAULT NULL,
  name VARCHAR(255) DEFAULT NULL,
  startDate VARCHAR(255) DEFAULT NULL,
  user_id BIGINT(20) DEFAULT NULL,
  state VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_7b7j83l6dquot72lsg25y8323 FOREIGN KEY (user_id)
    REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 21
AVG_ROW_LENGTH = 4096
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE plan_to_day (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  date VARCHAR(255) DEFAULT NULL,
  user_id BIGINT(20) DEFAULT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_2fcaxdfpxb6300y9b0sqmyf4h FOREIGN KEY (user_id)
    REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 18
AVG_ROW_LENGTH = 1820
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE fixing_hash_acount (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  value DOUBLE NOT NULL,
  acount_id BIGINT(20) NOT NULL,
  fixing_acout_id BIGINT(20) NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_26e396o8t0bshlgwsldpikq37 FOREIGN KEY (acount_id)
    REFERENCES acount(id) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT FK_ltlj0n2ot83vfgxoggtx5rxem FOREIGN KEY (fixing_acout_id)
```

```
    REFERENCES fixing_acount(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 1
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE item_goal_plan (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  anotation VARCHAR(255) DEFAULT NULL,
  end_date VARCHAR(255) NOT NULL,
  name VARCHAR(255) NOT NULL,
  start_date VARCHAR(255) NOT NULL,
  acount_id BIGINT(20) DEFAULT NULL,
  goal_id BIGINT(20) DEFAULT NULL,
  state VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_b4gdxpqap53w6ngdxwru48ahh FOREIGN KEY (goal_id)
    REFERENCES goal(id) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT FK_bfacaqha2kfjqrwg6o08m8mre FOREIGN KEY (acount_id)
    REFERENCES acount(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 24
AVG_ROW_LENGTH = 2340
CHARACTER SET utf8
COLLATE utf8_general_ci;
CREATE TABLE item_plan_to_day (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) DEFAULT NULL,
  status VARCHAR(255) DEFAULT NULL,
  time_notifice VARCHAR(255) DEFAULT NULL,
  item_goal_plan_id BIGINT(20) DEFAULT NULL,
  plan_to_day_id BIGINT(20) DEFAULT NULL,
  anotation VARCHAR(255) DEFAULT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_5sce36yuykcr4gdv33elbovvs FOREIGN KEY (item_goal_plan_id)
    REFERENCES item_goal_plan(id) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT FK_heohxmesm3ou68j85wfla5xtn FOREIGN KEY (plan_to_day_id)
    REFERENCES plan_to_day(id) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
AUTO_INCREMENT = 46
AVG_ROW_LENGTH = 1638
```

```
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Лістинг А.2 – Код модулю виконуючого функцію реєстрації користувач

```
public class AuthorizationFunction implements Function {

    @Override
    public String processes(String xml) throws Exception {
        ParserAuthorization handler = new ParserAuthorization();
        try {
            InputSource source = new InputSource(new StringReader(xml));
            SAXParserUtil.getParser().parse(source, handler);
        } catch (Exception e) {
            e.printStackTrace();
        }
        User user = handler.getUser();

        SessionManager sessionManager = SessionManagerImpl.getInstance();
        Object tempObj =
SpringUtil.getInstance().getBean("sessionManager");
        if (tempObj instanceof SessionManager) {
            sessionManager = (SessionManager) tempObj;
        } else {
            throw new ExceptionInInitializerError("Не соответствие типов
при инициализации");
        }
        org.hibernate.Session sessiondao =
HibernateUtil.getSessionFactory().openSession();
        List<User> users
=sessiondao.createCriteria(User.class).add(Example.create(user)).list();
        sessiondao.close();
        if (users.size()>0) {
            user = users.get(0);
            Session session = sessionManager.createSession();
            session.setAttribute("user", user.getId());
            return "<authorization><sessionId>" + session.getIdSession() +
"</sessionId></authorization>";
        } else {
            return "<error>User not found</error>";
        }
    }
}
```

Лістинг А.3 – структура запитів до веб сервісу

```
<?xml version="1.0" encoding="UTF-8"?>

<request
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='APIRequest.xsd'>
    <!--Пример запроса на регистрацию-->
    <registration>
        <name>YuriPanasenco</name>
        <mail>Panasenco@outlook.com</mail>
        <hashPass>12345</hashPass>
    </registration>
    <!--Пример запроса на авторизацию-->
    <authorization>
        <name>ebtb</name>
        <hashPass>sdfvrf</hashPass>
    </authorization>
    <!--пример запроса на создание цели-->
    <createGoal>
        <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
        <name>Название цели</name>
        <anotation>Краткое описание цели</anotation>
        <beginDate>07.09.2016</beginDate>
        <endDate>01.07.2018</endDate>
        <plan>
            <item>
                <number>1</number>
                <name>Получить высшее образование (Уровень магистр)</name>
                <beginDate>01.09.2016</beginDate>
                <endDate>01.07.2018</endDate>
            </item>
            <item>
                <number>2</number>
                <name>Накопить 5 тыс. доларов</name>
                <beginDate>01.09.2016</beginDate>
                <endDate>01.07.2018</endDate>
                <idAcount>4</idAcount>
            </item>
        </plan>
    </createGoal>
```

```
<!--пример запроса на закрытие сесии-->
<closeSession>
    <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
</closeSession>
<!--пример запроса на получение информации о пользователе-->
<userState>
    <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
</userState>
<!--пример запроса на получение списка пользовательских целей-->
<userGoals>
    <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
</userGoals>
<!--пример запроса на добавление задачи на завтра-->
<addTaskOnTommorow>
    <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
    <task>
        <name>название</name>
    </task>
    <task>
        <name>название </name>
        <notification>15:30</notification>
        <idItemGoalPlan>35</idItemGoalPlan>
    </task>
</addTaskOnTommorow>
<!--пример запроса на изменения или удаления задачи-->
<updateTaskOnDay>
    <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
    <task>
        <operation>update</operation>
        <id>345</id>
        <name>название задачи</name>
        <notification>15:30</notification>
        <idItemGoalPlan>35</idItemGoalPlan>
    </task>
    <task>
        <operation>delete</operation>
        <id>3435</id>
    </task>
    <task>
        <operation>update</operation>
        <id>632</id>
        <state>2</state>
    </task>
```

```
            </updateTaskOnDay>
            <!--пример запроса на получения задач пользователя-->
            <globalplan>
                <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
            </globalplan>
            <!--пример запроса на получение информации по конкретной задачи-->
            <goalInfo>
                <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
                <golaID>234</golaID>
            </goalInfo>
            <!--Возвращает список дел выполненых для достижения поставленой
задачи->
            <goalReport>
                <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
                <goalId>234</goalId>
            </goalReport>
            <!--пример запроса на получение статистики пользователя-->
            <statistics>
                <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
            </statistics>
            <acountReport>
                <sessionId>rgthryujtikyt</sessionId>
                <acountId>45</acountId>
            </acountReport>
            <!--Пример запроса для подтверждения регистрации-->
            <confirmation>
                <sessionId>tnlekmgregreko54ymn6oyh5k4my</sessionId>
                <key>uijonuhibjonot6f5ft7</key>
            </confirmation>
        </request>
```

Додаток Б - Інструація користувача

Розроблена програма необхідна для збору аналізу та прогнозування трудомісткості виконання задач з розробки ПЗ за Agile методологією. Програма має клієнт-серверну архітектуру та виконується на сервері за адресою https://app.taskring.com.

Програма написана мовою програмування PHP з використанням елементів JavaScript та дозволяє виконувати наступні дії.

Для користувача:

- створення задачі – введення інформацію про задачу яку необхідно виконати;
- відправлення діалогу задачі іншому учаснику;
- перегляд списку задач;
- редагування задачі;
- відправлення задачі в іншу компанію;
- перегляд звіту з прогнозом на кінець місяця;
- встановлення пріоритету задачі.

Для адміністратора компанії:

- створення резервної копії БД компанії;
- ведення списку користувачів;
- перегляд списку видалених задач.

Для адміністратора сервісу (Супер адміністратора):

- створення резервної копії всіх даних системи;
- перегляд списку компаній;
- створення компанії.

Для початку використання програми необхідно перейти за посиланням https://app.taskring.com/login (див. рис. Б.1), де система запропонує ввести назву компанії.

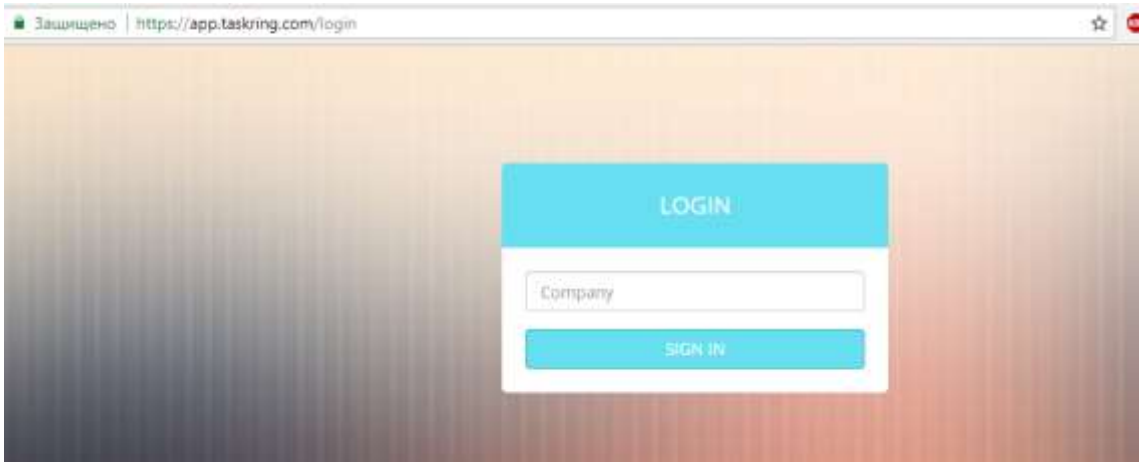

Рисунок Б.1 – Форма 1-го етапу авторизації

Після чого система запропонує ввести логін та пароль користувача (див. рис. Б.2).

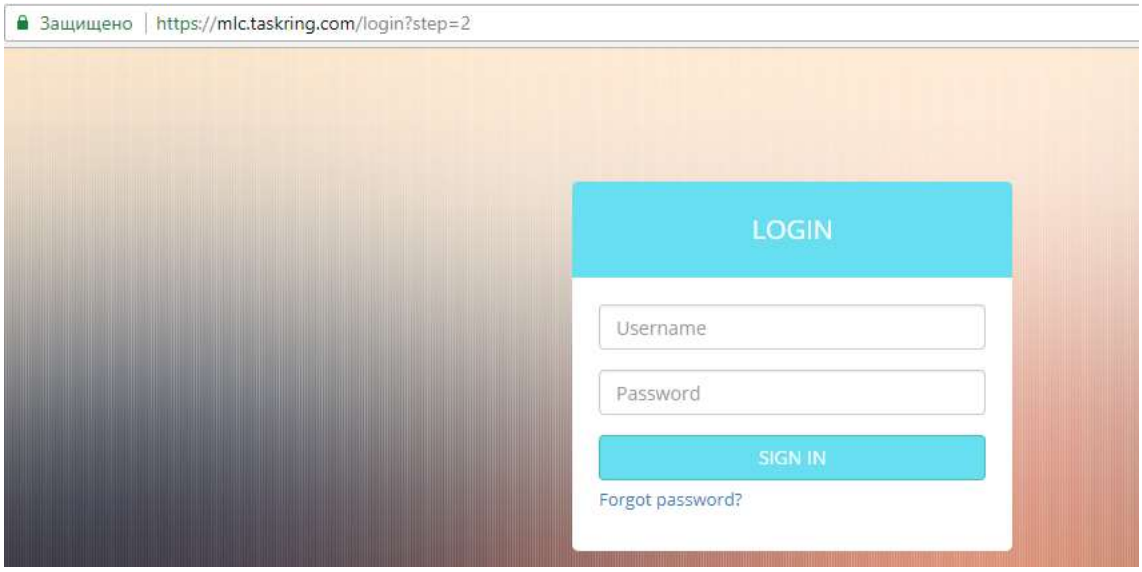

Рисунок Б.2 – Форма другого етапу авторизації

Після успішного проведення авторизації система направить користувача до головної сторінки (див. рис. Б.3).

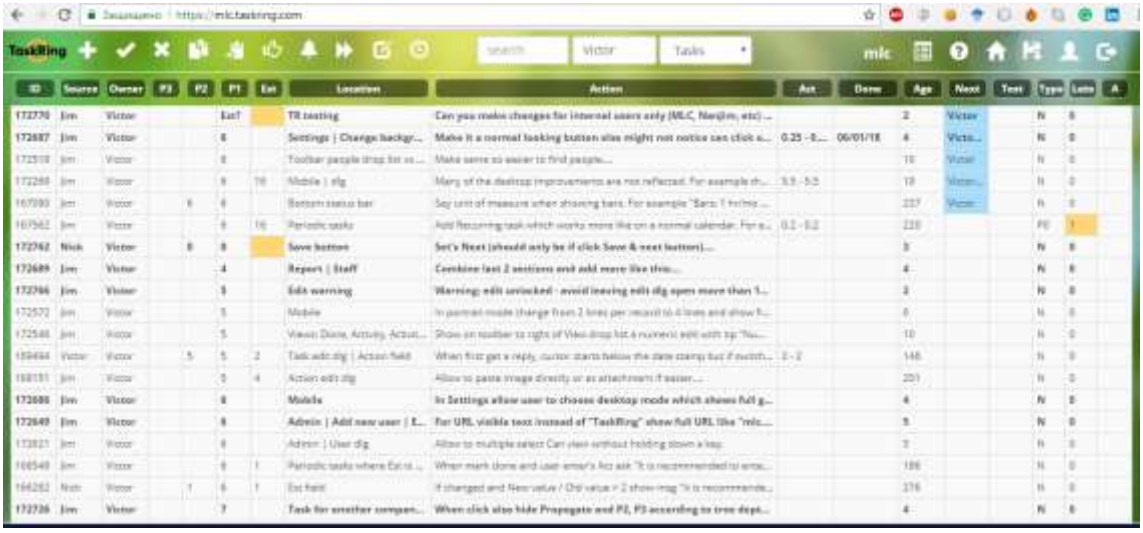

Рисунок Б.3 – Головна сторінка користувача.

На головній сторінці користувача представлено перелік задач які необхідно виконати користувачеві. Для перегляду задачі необхідно вибрати відповідний рядок, та натиснути кнопку редагування на верхній панелі керування. Після чого системі відобразить форму редагування задачі (див. рис. Б.4).

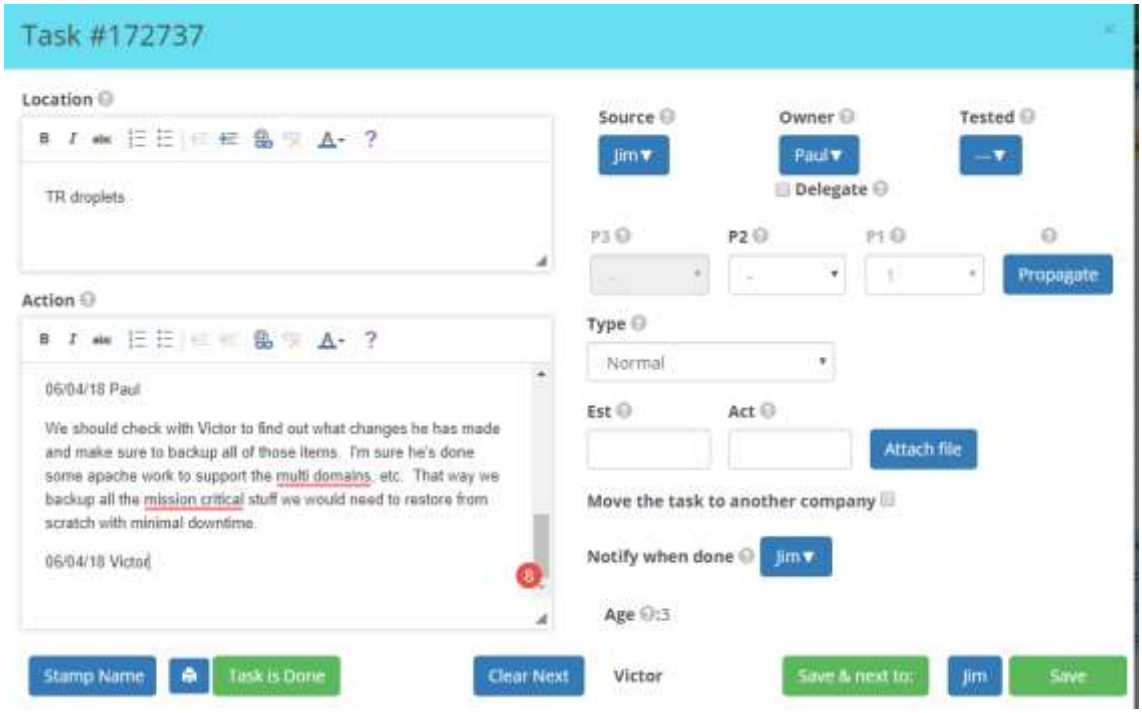

Рисунок Б.4 – Форма редагування задачі

Ні формі редагування задачі користувач має можливість ввести інформацію про те, чого стосується дана задача, що саме необхідно зробити, вибрати відповідального за виконання задачі, вказати пріоритет та час, що відводиться на виконання задачі.

Також користувач може вибрати один з декількох виглядів списків задач (див. рис. Б.5). Кожний вигляд списку задач представляє собою інформацію про статуси в яких знаходиться задача на даний момент.

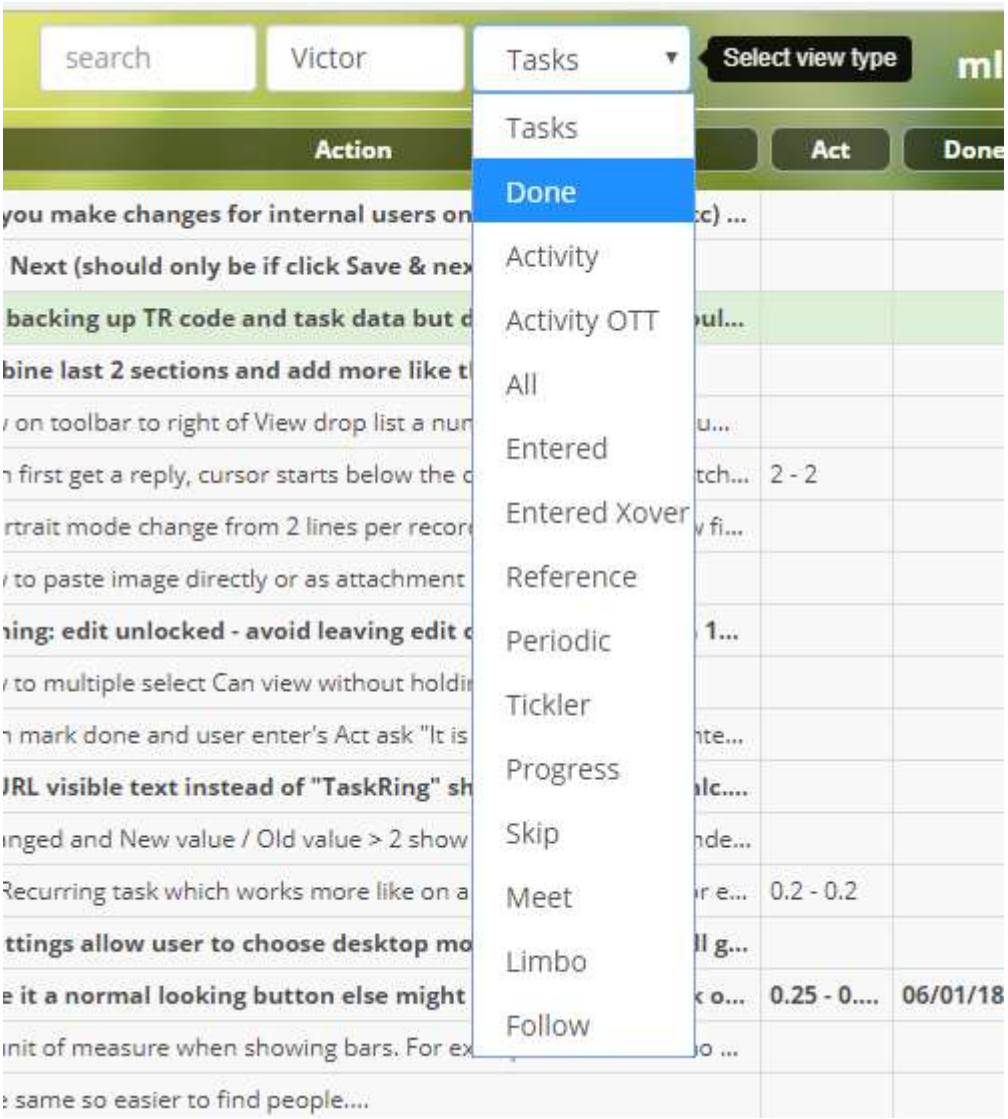

Рисунок Б.5 – Вигляд видів списку задач

Також в системі присутня можливість перегляду звіту користувача, в якому відображено активність користувача за вказаний період у вигляді

кількості витраченого часу, кількості виконаних завдань та прогнозного значення (див. рис. Б.6).

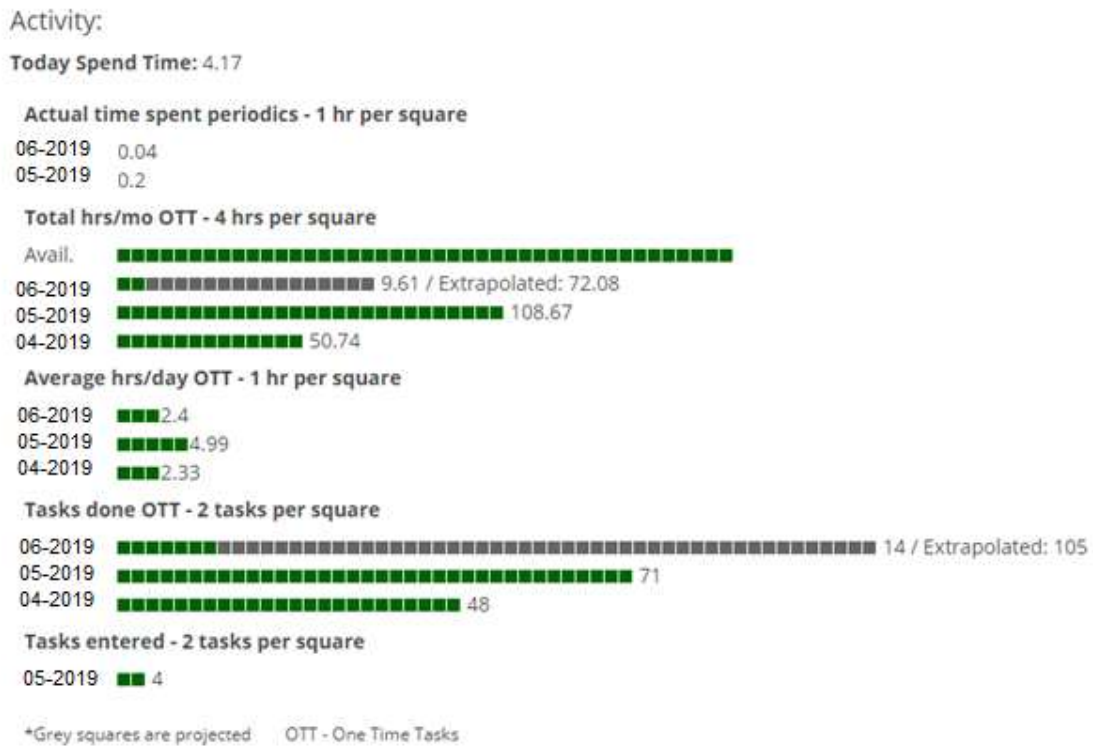

Рисунок Б.6 – Звіт активності користувача за 3 місяці