МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

СТОЙКО Володимир Іванович

Математичне та програмне забезпечення виконання завдань при розробці програмних продуктів / Mathematical Tools and Software for Tasks Implementation in the Development of Software Products

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
В. І. Стойко

_____

Науковий керівник:
к.е.н., доцент, Л. І. Гончар

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРОЦЕСУ ПЛАНУВАННЯ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 10
1.1. Визначення поняття проекту 10
1.2. Аналіз процесу планування проектів 11
1.3. Аналіз основних методів прогнозування термінів проекту 18
1.4. Огляд та аналіз систем ведення проекту 22
Висновки до першого розділу 27
РОЗДІЛ 2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ ВИКОНАННЯ ЗАВДАНЬ ПРИ РОЗРОБЦІ ПРОГРАМНИХ ПРОДУКТІВ 28
2.1. Модель процесу виконання задач при розробці програмного забезпечення 28
2.2. Метод розрахунку ймовірності виконання задач 31
Висновки до другого розділу 43
РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ ВИКОНАННЯ ЗАВДАНЬ ПРИ РОЗРОБЦІ ПРОГРАМНИХ ПРОДУКТІВ 44
3.1. Загальна архітектура системи 44
3.2. Інструкція користувача по роботі з системою 55
3.3.Прогнозування процесу виконання завдань у проекті розробки системи автоматизованого аналізу текстів 58
3.4.Експериментальні дослідження та оцінка ефективності запропонованого підходу 64
Висновки до третього розділу 69
ВИСНОВКИ 70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72

ДОДАТОК А ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ ..... **Помилка! Закладку не визначено.**

ВСТУП

Актуальність теми. При розробці програмного забезпечення (ПЗ) велике значення має планування проекту. Часто саме через помилки у плануванні програмний продукт не випускається вчасно або не вкладається у виділений бюджет.

З одного боку, це пов'язано зі складністю завдання планування та врахування особливостей продукту, замовника, команди розробки, фінансових засобів тощо, з іншого – з великою роллю людського чинника у питаннях планування.

Інтенсивний розвиток інформаційних та комунікаційних технологій призвів до формування нових вимог до планування та управління проектами у різних галузях. Процес виконання проекту розробки ПЗ динамічний, вимоги та набір завдань можуть часто змінюватися, помилка у плануванні може призвести до значної затримки часу закінчення проекту та/або суттєвого перевитрати ресурсів.

Однією з обов'язкових умов ефективної реалізації проектів стає застосування сучасних засобів та інструментів управління проектами, заснованих на використанні нових інформаційних технологій. Розвиток спеціального програмного забезпечення для планування та управління проектами обумовлено насамперед необхідністю максимальної інтеграції найбільш ефективних методів, засобів та інструментів теорії управління проектами для оперативного доступу до аналітичних даних за станом проекту.

Зв'язок роботи з науковими програмами, планами, темами

Напрямок виконаних досліджень безпосередньо пов'язаний з науково-дослідним напрямком кафедри “комп’ютерних наук” Західноукраїнського національного університету.

Мета і задачі дослідження

Метою магістерської роботи є розробка спеціального математичного та алгоритмічного забезпечення системи аналізу та підтримки прийняття рішень при управлінні проектами розробки програмного забезпечення.

Для вирішення поставленої мети необхідно розв’язати наступні завдання:

- аналіз відомих моделей процесу розробки ПЗ, підходів до планування та існуючих програмних продуктів для проектів розробки ПЗ;

-дослідження інформаційних процесів при виконанні проекту розробки програмного забезпечення з метою проведення моделювання та їх автоматизації;

- розробка математичної моделі процесу виконання завдань під час розробки ПЗ;

- розробка системи підтримки прийняття рішення під час планування на основі математичної моделі;

- проведення експериментальних досліджень перевірки створених моделей та системи підтримки прийняття рішення під час планування проекту розробки ПЗ.

Об’єкт дослідження – процеси розробки програмного забезпечення.

Предмет дослідження – методи та програмні засоби при планувані проектів розробки програмного забезпечення.

Методи дослідження

Методи математичного аналізу, математичного моделювання, чисельні методи; методи теорії ймовірностей та математичної статистики, масового обслуговування, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів

Запропоновано функціонально-інформаційне модель процесу виконання проекту розробки ПЗ з використанням ймовірнісної моделі планування, що відрізняється можливістю перебудови плану проекту залежно від його поточного стану або зміни набору завдань та ресурсів.

РОЗДІЛ 1
АНАЛІЗ ПРОЦЕСУ ПЛАНУВАННЯ ПРОЕКТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Визначення поняття проекту

Проект зазвичай визначається як унікальний набір скоординованих дій із заданим початком та завершенням, що здійснюються індивідуумом або організацією для вирішення специфічних завдань відповідно до наявного розкладу, витрат і параметрів виконання [1-4].

Головна властивість проекту в тому, що він задає обмежене в часі зусилля, створене для створення унікального продукту чи послуги. Унікальність кожного проекту породжує складнощі при його плануванні, оскільки часто важко достовірно припустити, як досягатимуться результати та наскільки можливо вкластися у визначені терміни. Результатом проектної діяльності є виконані завдання та набутий досвід, який враховується під час планування наступних проектів.

Управління проектами складається з двох основних частин: складання плану та контроль ходу робіт по ньому. Якісний план проекту полегшує надалі виконання проектних завдань та підвищує ймовірність вдалого завершення проекту. Є безліч факторів, які позитивно та негативно впливають на перебіг проекту. Наприклад, декомпозиція цілей більш дрібні і зрозумілі зазвичай позитивно впливає перебіг проекту [5-9].

Управління проектами – це сукупність процесів (методик, моделей, програмних і технічних засобів, методологій), які здійснюються при розробці та реалізації проектів, ці процеси мають обмеження в часі, вимагають витрат ресурсів.

Під ресурсами проекту зазвичай розуміють бюджет, час, працю, матеріали, енергію тощо. У галузі розробки програмного забезпечення є суттєвими:

- часові ресурси;
- людські ресурси;
- фінансові ресурси (бюджет).

При плануванні проводиться загальний аналіз робіт та ресурсів, також необхідно враховувати їхню обмеженість та передбачуваний розподіл на основі потреб проектів. Ресурсне планування включає аналіз ресурсів і робіт, розробку системи розподілу ресурсів, контроль над ходом робіт, вибір коригуючих заходів [10-15].

У кожного проекту чітко визначено початок та критерії закінчення. Проект закінчується разом із досягненням усіх його цілей або коли ці цілі не можуть бути досягнуті. Проекти зазвичай мають чітко окреслені часові рамки для створення продукту, які часто встановлюються ринком [16].

1.2. Аналіз процесу планування проектів

Реалізація проекту розробки програмного забезпечення відповідно до висунутих вимог – вкрай актуальне завдання. Проект починається з визначення мети, яку часто неможливо сформулювати на початку проекту. Це призводить до того, що учасники проекту розуміють її по-різному. У результаті мета проекту уточнюється в міру виконання проекту, що в результаті призводить до змін у вимогах проекту [17-22].

Потім зазвичай складається план у традиційному розумінні, де планування полягає у призначенні робіт, визначенні їхньої тривалості та бюджету. Подальший контроль виконання полягає у порівнянні фактичної тривалості робіт та витрат з плановими значеннями. При цьому цілі проекту,

заради досягнення яких проект власне і замислювався, відходять на другий план, і про них згадують тоді, коли проект майже завершено.

На початку кожного проекту стоїть завдання визначення його ресурсоємності, часу реалізації, бюджету та відповідно до здійсненності при обраному поєднанні цих параметрів. Це завдання вирішується приблизно, оскільки існуючі методики пропонують лише приблизні оцінки, що практично не враховують специфіку проекту, а необхідні ресурси визначаються виключно на основі досвіду та суб'єктивної думки людей, які виступають у ролі експертів, не завжди є фахівцями в даній галузі.

У будь-якому проекті доводиться балансувати між вартістю, часом, якістю та обсягом функціональності, що реалізується. Точний розрахунок ресурсів, необхідні реалізації даного продукту за певних вимог до якості, є однією з основних проблем у галузі управління проектами. Особливо, якщо йдеться про проекти розробки програмного забезпечення. При цьому багатьма авторами виділяється насамперед велика творча складова процесу розробки програмного забезпечення. Тому в розрахунках характеристик проектів виникають складності обліку величезної кількості факторів, у тому числі важкоформалізованих, що впливають на життєвий шлях проекту. Як наслідок, сьогодні багато компаній стикаються із серйозними проблемами у разі неправильних розрахунків необхідних термінів:

- при недооцінці – непередбачене витрачання додаткових коштів, невдоволення замовника невиконанням зобов'язань у обумовлений час, переробки співробітників, складність управління неконтрольованим проектом, низька якість кінцевого продукту, недостатні функції системи тощо;

- при переоцінці – перевитрата витрат ресурсів, залучених до проекту, відмова замовника від контракту з цими умовами тощо.

Точні оцінки витрат виробництва програмного забезпечення важливі як розробникам, і замовнику. Виникає потреба у розробці методів та засобів, що дозволяють менеджеру проекту оцінити необхідні часові та людські ресурси

на основі всіх наявних даних: історії попередніх подібних проектів, досвіду та продуктивності праці співробітників, специфіки компанії.

Крім того, потрібно також мати можливість перерахунку та уточнення термінів та ресурсів вже в ході розробки програмних систем з урахуванням розбіжності поточного стану проекту та його плану. Це може допомогти керівнику своєчасно виявити відхилення від встановленого графіка та вжити відповідних заходів в управлінні проектом.

Більшість проектів мають певні дату закінчення, бюджет та обсяг робіт. Час, ресурси та обсяг робіт у зазвичай називають проектним трикутником (рисунок 1.1). Зміни в одному з цих елементів викликають зміни інших. У загальному випадку для проекту важливі всі три складові, але, як правило, тільки один з них, залежно від пріоритетів, має найбільший вплив на інші. У цьому випадку говорять про проекти з фіксованим часом виконання (time-driven, найсуттєвішим параметром є час), з фіксованим обсягом робіт (feature driven, для реалізації основних функцій системи можуть бути збільшені ресурси або терміни) і з фіксованою вартістю (resource driven, пріоритетним є обмеження одного із ресурсів).

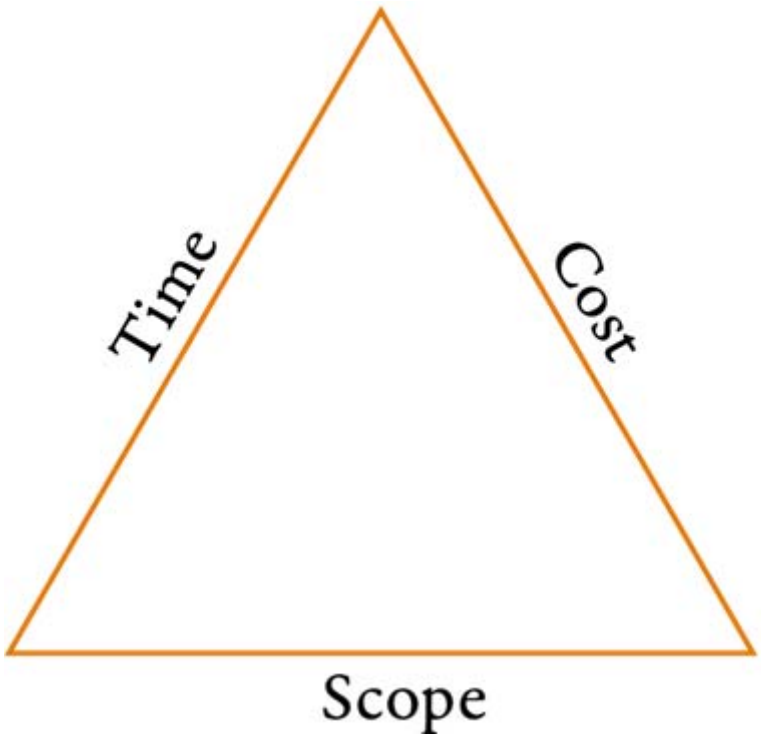

Рис. 1.1. Проектний трикутник

Наприклад, якщо рахунок зменшення термінів окремих робіт змінюється план проекту, то зростає вартість проекту (якщо, наприклад, вирішено залучити додаткових працівників) чи має зменшитися обсяг робіт. Якщо ж змінити план проекту з метою зменшення його бюджету, то може зрости тривалість проекту та зменшитись обсяг робіт. Зрештою, якщо збільшити обсяг робіт, то проект триватиме довше і коштуватиме дорожче. Те, як зміни у плані впливають інші сторони трикутника, залежить від обставин і специфіки проекту.

Якість, четвертий елемент проектного трикутника, знаходиться в його центрі, і зміни, що вносяться в будь-яку сторону трикутника, практично завжди впливають на якість (рисунок 1.2). Якість не є стороною трикутника - це результат дій з часом, вартістю та обсягом робіт.

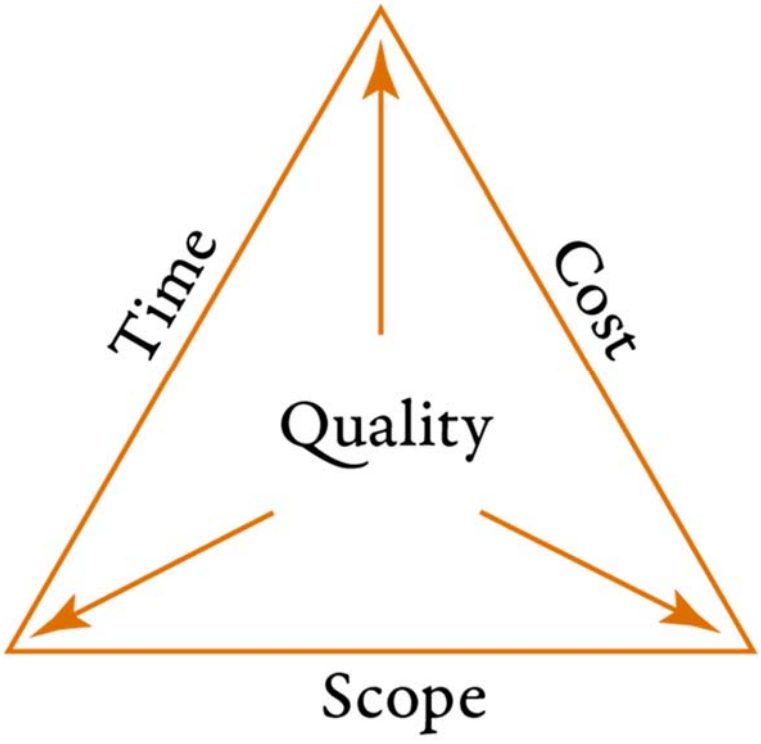

Рис. 1.2. Проектний трикутник із врахуванням якості

У роботах [16-19] розглядається багато сучасних варіантів організації розробки програмного забезпечення, виробляються критерії оцінки та вибору альтернатив, а також їх різні класифікації. Практично кожен проект, часто намагаючись слідувати тим чи іншим принципам, адаптує їх до своїх реалій,

створюючи тим самим новий підхід. При цьому практично завжди команда розробки в чомусь обмежена. Тому у роботі ці підходи класифікуються за способом побудови процесу розробки програмного забезпечення з урахуванням пріоритетного типу обмежень.

Вибір серед варіантів побудови процесу розробки ПЗ є досить складним завданням. Найчастіше такий вибір відбувається без урахування всіх можливих альтернатив, розглядаючи лише обмежений набір методик, у своїй не враховується специфіка проекту.

Підходи до розробки програмного забезпечення, засновані на обмеженнях розмірів проекту, історично виникли першими. Це в першу чергу тим, що обмеження розмірів проекту (як, втім, і ресурсів) логічно випливають із самої суті проекту.

Як тільки вдається визначити та декомпозувати мету проекту відразу виникають рамки проекту (scope), у яких визначено проект та обмеження. Проект, який не має меж, наперед не успішний. Усі підходи цієї групи засновані на чіткому поділі проекту на фази, відрізняючись лише тривалістю та ступенем опрацювання фаз.

Існує набір стандартів, що визначають різні елементи у структурі життєвих циклів програмного забезпечення та програмно-апаратних систем. В якості основних таких елементів виділяють технологічні процеси - структуровані набори діяльностей, що вирішують деяке загальне завдання або пов'язану сукупність завдань, такі як процес супроводу ПЗ, процес забезпечення якості, процес розробки документації та іншими різними видами діяльностей, артефактами та ролями зацікавлених осіб [14]. Найпоширенішими варіантами побудови процесу розробки ПЗ є: каскадний підхід, V-подібний, спіральний та ітераційний підходи.

Слід відзначити, що, незважаючи на численну критику і великий вік каскадного підходу, найчастіше, практично, навіть якщо заявлений інший підхід, застосовується саме каскадний, мабуть, через простоту його застосування.

Знаходження вірного балансу між ресурсами, часом розробки та можливостями – найважливіший момент у побудові рішення, що відповідає потребам замовника.

Наприклад, порівняння важливості обмежень кожного типу для підходів до розробки, дає діаграму, представлену на рисунку 1.3.

Рис. 1.3. Різниця проектних трикутників для різних підходів

Методи побудови процесу розробки ПЗ, засновані на обмежені ресурсів. Обмеження різного виду ресурсів, безумовно, присутні у кожному проекті. Зазвичай вони випливають з обмежень решти типу, коли на стадії планування з'ясовується, яка з складових гратиме найбільше значення для проекту в цілому. Зазвичай, обмеження ресурсів доповнюють обмеження рамок проекту.

Методи побудови процесу розробки, засновані на обмеженні часу. Досвід останніх років показав, що для систем, вимоги до яких досить часто змінюються, необхідно максимально зменшити тривалість одного етапу життєвого циклу. У зв'язку з цим в даний час стали дуже популярними короткі життєві цикли розробки, які були об’єднані загальною назвою гнучкі методи (agile).

Найсуттєвішим обмеженням при такому варіанті побудови процесу розробки є час. Зазвичай на якийсь короткий проміжок часу (1-3 тижні)

фіксується набір функціоналу майбутньої системи, що підлягає реалізації. При цьому проектування, програмування, тестування протягом міні-циклу йдуть майже паралельно. Звичайно, не можна реалізовувати, а тим більше перевіряти модуль, який ще не спроектований.

Основна ідея життєвого циклу – максимальне вкорочування тривалості кожного етапу життєвого циклу та тісна взаємодія із замовником. По суті, на кожному етапі відбувається реалізація та тестування однієї функції системи, після завершення якого система одразу передається замовнику на перевірку чи експлуатацію.

Головною проблемою є здійснення зрозумілої та документованої взаємодії, наприклад, інтерфейсами між модулями, що реалізують якусь функцію. Якщо у всіх попередніх типах життєвого циклу інтерфейси досить чітко визначаються на самому початку розробки, оскільки заздалегідь відомі всі модулі, то при «гнучкому» підході інтерфейси проектуються одночасно з модулями, що зазвичай розробляють негативний вплив на систему в цілому в довгостроковій перспективі.

Ще до появи прототипу прийнято приступати до створення тестового середовища, максимально наближеного до реального середовища виконання програми, і коли з'являються проекти інтерфейсів – вже можна розпочати розробку прототипів функціональних тестів.

Всі ці процеси, притаманні всім моделям життєвого циклу, йдуть майже одночасно, що потребує великої організованості та високого професіоналізму усієї команди розробки. Відсутність чіткої документованості процесів, незамінність кожного в команді, а також найчастіше брак системного погляду на проект – найсерйозніші проблеми цього підходу.

Але, незважаючи на недоліки, Agile-підхід є найпрогресивнішим і найперспективнішим серед усіх існуючих, тому що він дозволяє подолати головну нестачу всіх інших – статичність вимог та проекту в цілому.

Програмне забезпечення має бути гнучким, змінюватись в залежності від зовнішніх умов та часто ніде не заявленим вимогам користувача.

Змішані підходи становлять найбільший інтерес для досліджень та застосування різних методик розрахунку характеристик проектів, оскільки на практиці через безліч причин саме вони знаходять застосування. Головною причиною можна виділити унікальність кожного проекту і неможливість підібрати всі наявні умови під будь-який один шаблон, тому зазвичай говорять про пріоритетну методику, що використовується, і її адаптацію для потреб конкретного проекту.

1.3. Аналіз основних методів прогнозування термінів проекту

План проекту, розроблений і затверджений до початку виконання робіт, є модель спрямованих на досягнення поставлених цілей взаємопов'язаних виробничих процесів. Як і будь-яка модель, календарний план має одночасно відповідати суперечливим умовам простоти та достатньої повноти.

Для розробки календарних планів використовуються методи мережевого моделювання, які дозволяють ув'язати виконання різних робіт і процесів у часі, отримавши в результаті загальну тривалість всього проекту.

В основу сучасного програмного забезпечення управління проектами (Microsoft Office Project (рисунок 1.4), Primavera (рисунок 1.5), Open Plan (рисунок 1.6), Spider Project (рисунок 1.7)) також є мережне моделювання в управлінні проектами.

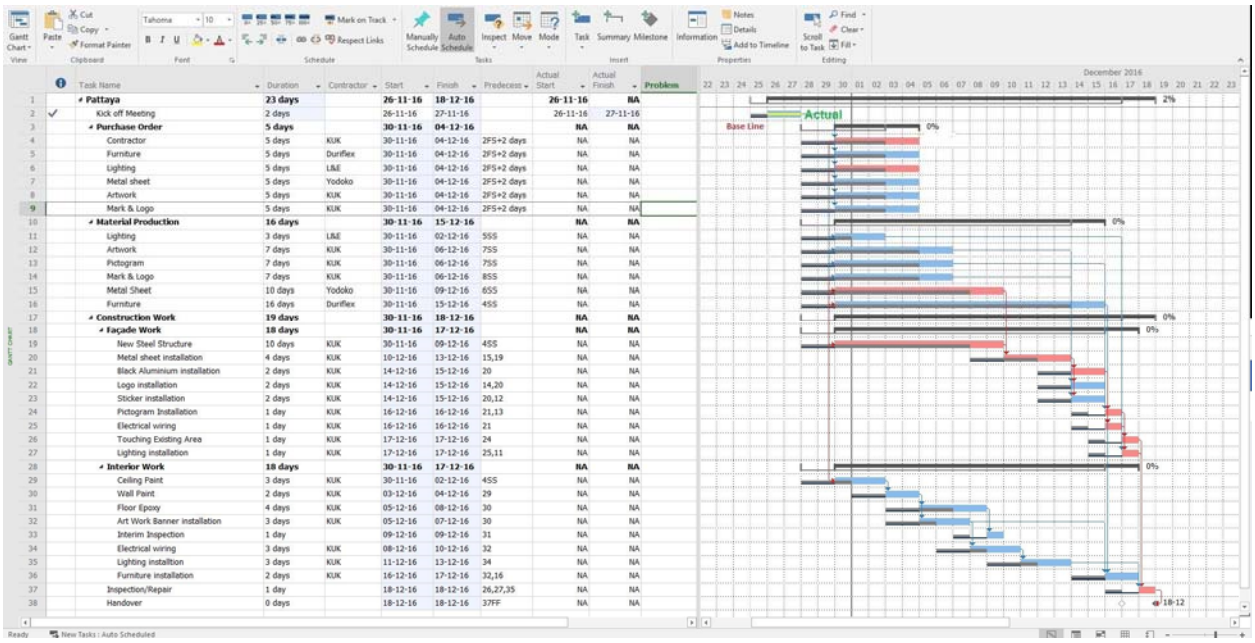

Рис. 1.4. Microsoft Office Project

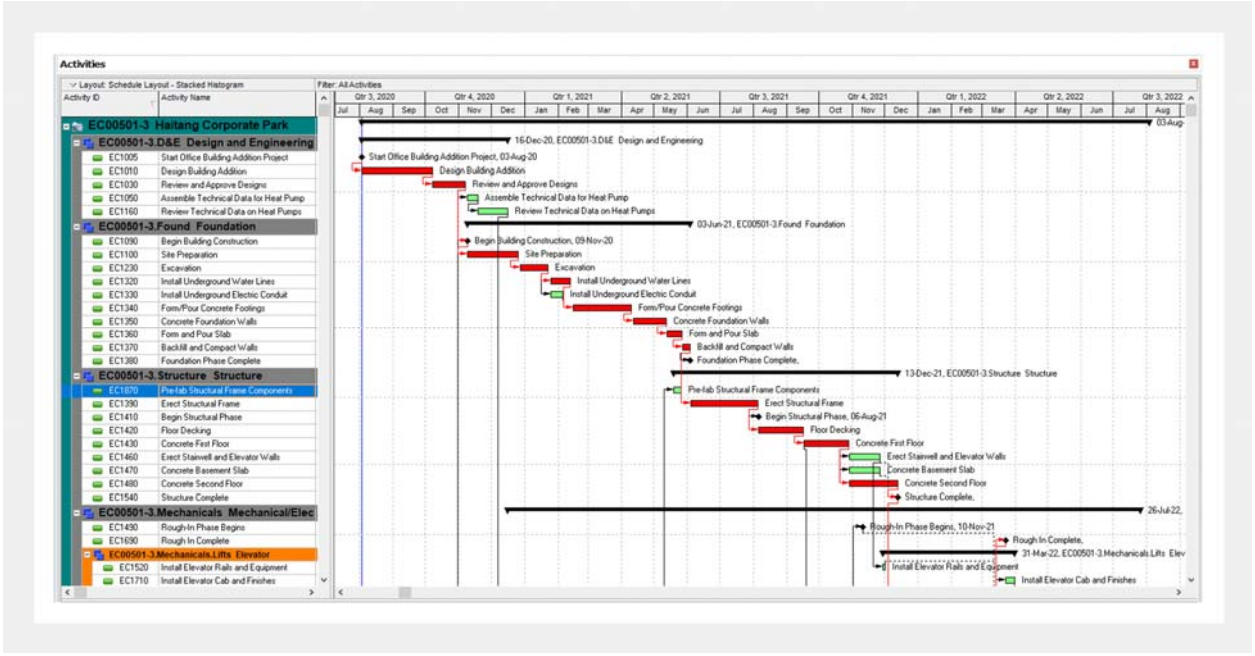

Рис. 1.5. Ситема Primavera

Під час мережного планування загальна тривалість проекту залежить від взаємозалежності робіт, від тривалостей робіт та тимчасових обмежень, встановлених на строки робіт. Зазвичай під час розрахунків розглядаються детерміновані мережеві моделі, взаємна послідовність і тривалість яких задані однозначно; виділяється критичний шлях (роботи від точки старту до

закінчення проекту, які не можуть бути розділені і виконуються максимальний час), з урахуванням оцінок якого оцінюється проект цілком.

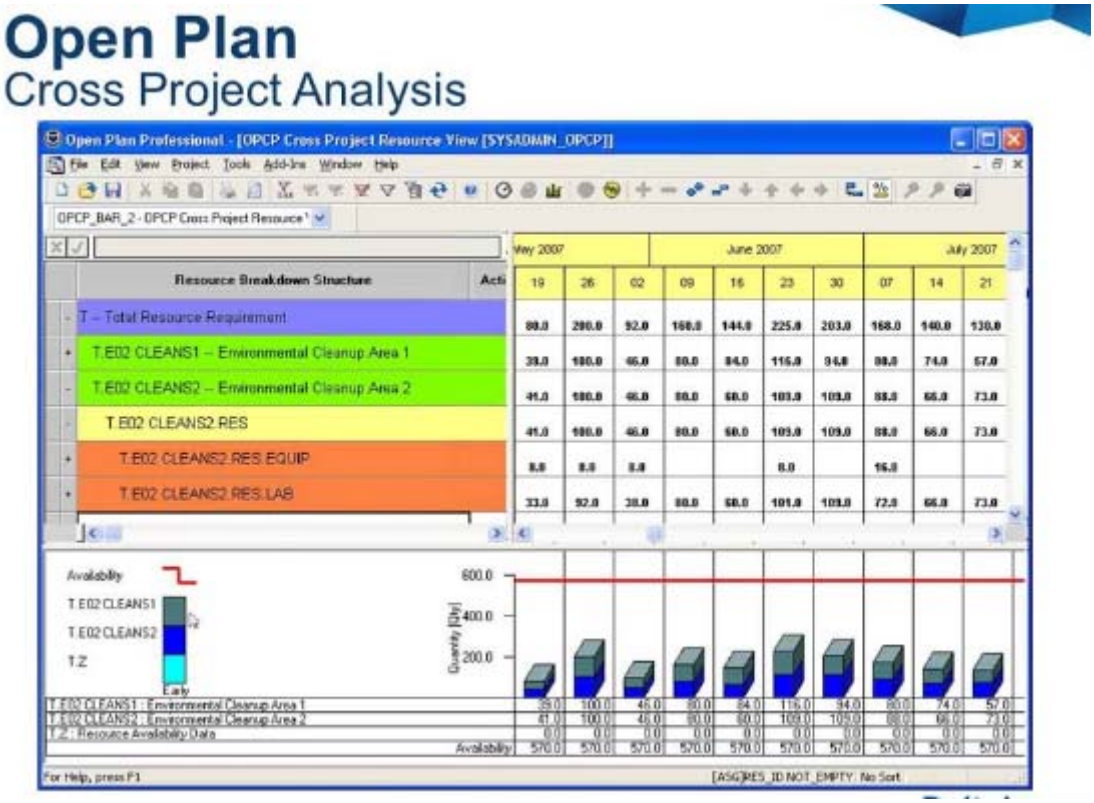

Рис. 1.6. Ситема Open Plan

Збільшення фактичної тривалості робіт, що належать критичному шляху, з якихось причин призводить до відповідного збільшення загальної тривалості проекту, в той час як некритичні роботи мають деякі резерви часу.

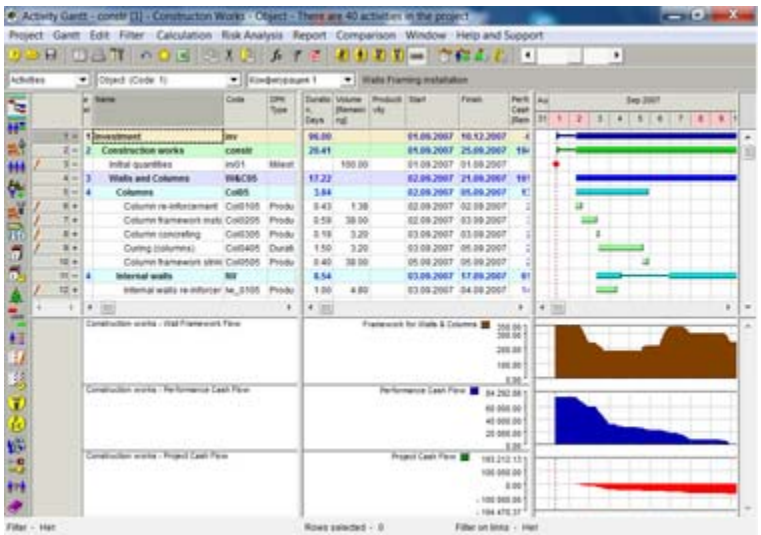

Рис. 1.7. Ситема Spider Project

Проте зі збільшенням фактичних термінів робіт некритичного шляху, варто враховувати можливість зміни критичного шляху як наслідок, поява необхідності повного повторного планування проекту. Одним із методів зниження ризику зсуву термінів закінчення робіт при детермінованому плануванні є виділення так званого страхового резерву часу, тобто плановий термін приймається на деяку кількість відсотків (зазвичай 10–30%) менше договірного, а період часу, що залишився, використовується для завершення невиконаних у строк робіт.

Внаслідок динамічного характеру та величезної кількості факторів, що впливають на успіх здійснення робіт, лише в окремих випадках фактична тривалість та вартість робіт може збігтися з аналогічними параметрами, зазначеними у календарному плані. Невизначеність, що має місце при плануванні, призводить до суттєвих розбіжностей між планом та фактичним виконанням робіт та необхідності постійних перерахунків календарного плану. Також нерідкі ситуації, у яких змінюється запланована послідовність виконання робіт.

Припустимо, що тривалість деякої роботи складно заздалегідь поставити однозначно. Це може бути обумовлено багатьма причинами: низькою кваліфікацією виконавців, новизною робіт, наявністю недокументованих частин чи помилок у сторонньому програмному забезпеченні, недоліками в документації тощо. У разі технічно складних проектів, пов'язаних із дослідженням нових технологій, однозначне визначення тривалостей робіт, а також їх взаємозв'язків стає практично неможливим. Все сказане відноситься не тільки до планування виконання робіт, але і до планування інших стадій проекту розробки програмного забезпечення: розробки документації, підготовки користувачів, настроювання обладнання тощо. У цих випадках необхідно використовувати імовірнісні мережні методи, застосування яких призводить, з одного боку, до ускладнення розрахунків, однак результати, що отримуються, є більш адекватними реальним умовам.

В даний час відомо багато методів імовірнісного мережевого планування, найбільш поширеними з яких є:

- метод оцінки та аналізу програм (Program Evaluation and Review Technique, PERT);

- метод статистичних випробувань або метод Монте-Карло;

- метод графічної оцінки та аналізу програм (Graphic Evaluation and Review Technique, GERT).

1.4. Огляд та аналіз систем ведення проекту

Інтенсивний розвиток інформаційних та комунікаційних технологій в останні роки призвело до формування нових вимог до планування та управління проектами у різних галузях. Однією з обов'язкових умов ефективної реалізації проектів стає застосування сучасних засобів та інструментів управління проектами, заснованих на використанні нових інформаційних технологій.

Розвиток спеціального програмного забезпечення для планування та управління проектами обумовлено насамперед необхідністю максимальної інтеграції найбільш ефективних методів, засобів та інструментів теорії управління проектами.

Діяльність практично будь-якого сучасного проекту, незалежно від тривалості, сфери діяльності, кількості та складу учасників, бюджету, намічених цілей, неможливо уявити без використання сучасного ПЗ. Нижче розглянуто класифікацію та функціональні особливості сучасних систем планування та управління проектами, особливості їх практичного застосування, можливості інтеграції та сумісності з іншими додатками, програмами та системами, підтримка сучасних стандартів та принципів управління проектами.

В даний час на ринку представлена значна кількість універсальних програмних пакетів для персональних комп'ютерів, що автоматизують функції планування та контролю календарного графіка виконання робіт.

Найбільш популярними є такі:

- Primavera Project Planner (P3) (Primavera);
- Microsoft Project (Microsoft);
- Time Line (Time Line Solutions);
- Open Plan (Welcome Software);
- Artemis Views (Artemis Management Systems);
- CA-Super Project (Computer Associates International Inc.);
- Project Scheduler (Scitor Corp.);
- TurboProject (IMSI);
- Project Workbench (Applied Business Technology);

Багато фахівців з розробки та впровадження систем управління проектами поділяють ПЗ на професійні та настільні (непрофесійні). Професійні системи надають гнучкіші засоби реалізації функцій планування та контролю, але вимагають великих витрат часу на підготовку та аналіз даних та високої кваліфікації користувачів. Другий тип пакетів призначений для тих, для кого управління проектами не є основною діяльністю. Від таких програм не потрібні специфічні функції планування або оптимізації розкладів. Для них важливішим є простота використання та швидкість отримання результату.

Однією з найпопулярніших систем ведення проектів є Open Plan Professional (рисунок 1.4). Ця система виділяється потужними засобами ресурсного планування, які дозволяють керувати різними видами ресурсів: людьми, обладнанням, матеріалами, фінансами з огляду на їх взаємозв'язки. В Open Plan передбачені спеціальні процедури планування та контролю витрат, серед яких особливо варто відзначити засоби аналізу та побудови звітів щодо фактичного вироблення. Засоби «запам'ятовування» кількох варіантів реалізації проекту та введення фактичних даних щодо витрат на

роботу та відпрацювання ресурсів дозволяють проводити аналіз витрат за проектом, як прогнозованих, так і реальних. В Open Plan є аналітичний інструмент, що базуються на методі Монте-Карло, що дозволяє визначити можливі допуски в оцінці терміну завершення окремих робіт, цілих етапів і всього проекту, причому істотним обмеженням є те, що необхідна для роботи методу імовірнісна оцінка робіт проводиться вручну.

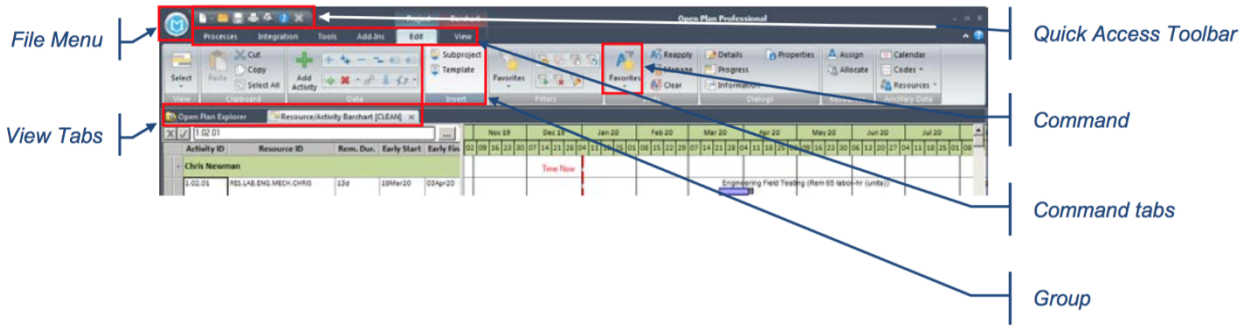

Рис. 1.7. Open Plan Professional

Open Plan надає безліч засобів, призначених для структуризації та візуалізації моделі проекту: ієрархічна структура завдань (WBS – Work Breakdown Structure), мережева модель (PERT–діаграма) та ін.

У Open Plan є можливість використання ймовірнісних оцінок, отриманих в результаті аналізу ризиків проекту. Також підтримується автоматичне розпізнавання критичного шляху проекту, за введеними даними про тривалість завдань та їх оцінки. Основною перевагою Open Plan є розгалужена система контролю над ресурсами та фінансами. Підтримуються багато видів звітів щодо використання різних видів ресурсів, і навіть великі можливості фінансів звітності проектів.

Одним із найпоширеніших програмних продуктів, призначених для управління проектами, є Microsoft Project (рисунок 1.8). На сьогоднішній день остання версія цієї програми – MS Project являє собою широкий набір програмних інструментів для управління проектами, але, як відзначають багато користувачів MS Project, головною його перевагою, крім наявних функцій, є глибока інтеграція з іншими засобами та продуктами Microsoft і

стандартний інтерфейс, звичний користувачам з інших програм з пакета MS Office.

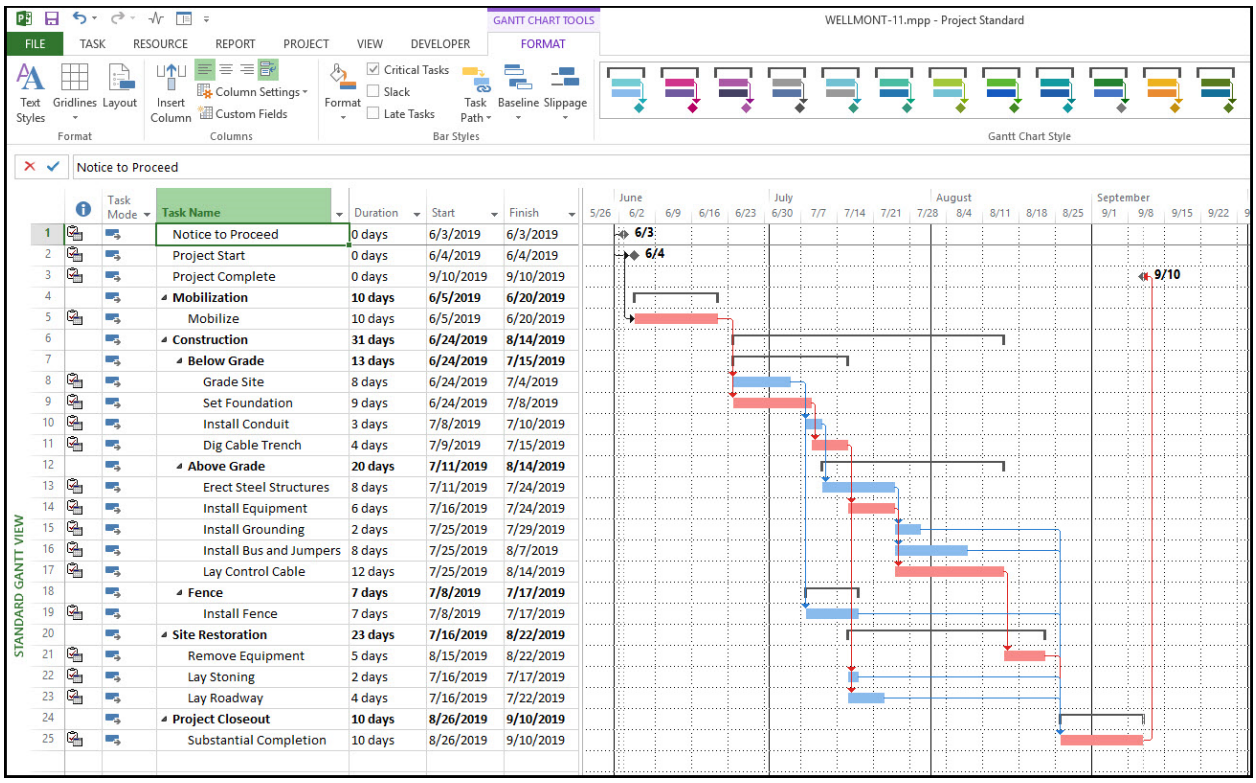

Рис. 1.8. Microsoft Project

Microsoft Project дозволяє керівникам проектів керувати календарними планами та ресурсами, отримувати відомості про стан проекту та аналізувати його виконання. Візуальні засоби планування, що застосовуються, дозволяють керівникам проектів швидко оцінити вплив змін, внесених у завдання або ресурси. Наприклад, якщо завдання відстає від календарного плану, MS Project показує вплив цієї зміни інші завдання і проект у цілому. Такі візуальні представлення даних проекту, як діаграма Ганта, мережевий графік та інші, дозволяють правильно оцінити залежність завдань та стан проекту.

Серед дорогих професійних рішень виділяється система управління проектами Primavera Project Planner (рисунок 1.9). Система застосовується великих проектів, дозволяє вирішувати складні ресурсні конфлікти, має розширені засоби аналізу використання ресурсів. Ще однією особливістю є

підтримка різних рівнів доступу та різних автоматизованих робочих місць. Дозволяє проводити введення та оновлення даних безпосередньо з PERT-подання, календарно-мережевого плану та тимчасової логічної діаграми. Також має функцію аналізу проекту за допомогою методу PERT.

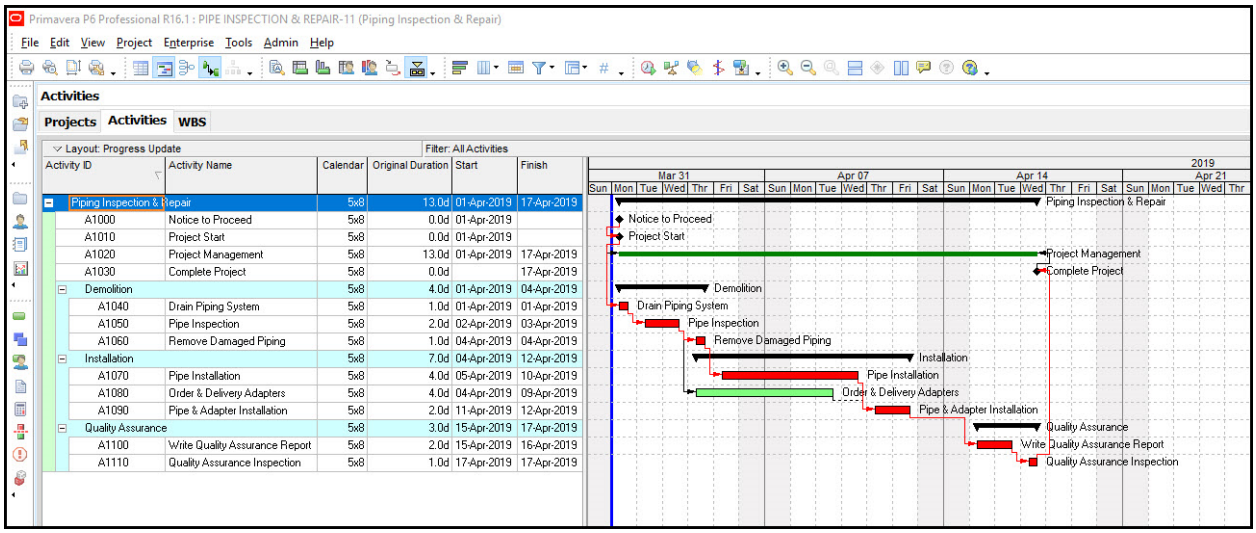

Рис. 1.9. Primavera Project Planner

Таким чином, сучасні системи календарного планування надають основний набір функціональних можливостей:

1. Засоби, призначені для опису робіт проекту, зв'язків між роботами та їх тимчасових характеристик, побудова мережевого графіка проекту. До них відносяться підтримка календаря проекту, обмеження, що накладаються на роботи проекту, можливості призначення часових характеристик та обмежень, зв'язки між завданнями та допустимі витрати.

2. Засоби підтримки інформації про ресурси та витрати за проектом та призначення ресурсів та витрат окремих робіт проекту. До них відносяться різні інструменти, призначені для ресурсного планування та оптимізації навантаження на ресурси проекту.

3. Засоби контролю за ходом виконання проекту та використанням ресурсів, що включають інструменти відстеження стану завдань проекту, відсоток завершення.

4. Зручні графічні засоби подання структури, візуальної побудови діаграм, засоби складання звітів. Важливими для користувача є простота вивчення та використання системи та якість додаткової консультаційної підтримки даної системи на ринку.

Висновки до першого розділу

1. Проаналізовано різні визначення проекту та дано визначення проекту та проектної діяльності з точки зору розробки ПЗ. Виявлено основні фактори, що перешкоджають точному розрахунку часу закінчення проекту.

2. Проведено аналіз обмежень процесу розробки програмного забезпечення. У ході його було проаналізовано та класифіковано основні підходи до розробки ПЗ; виділено їх особливості, які необхідно враховувати під час створення методів планування.

3. Було проаналізовано відомі методи розрахунку термінів виконання проектів, визначено їх переваги та недоліки. Аналіз ходу реальних проектів показав, що більшість методів не надають користувачам необхідну точність при плануванні проектів розробки програмного забезпечення. Серед причин цього можна виділити насамперед їхню орієнтацію на проекти «в цілому», без зазначення конкретної галузі застосування даної методики. Тому розробка методів, що дають змогу більш точно планувати час закінчення проекту розробки програмного забезпечення, є актуальним завданням.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ ВИКОНАННЯ ЗАВДАНЬ ПРИ РОЗРОБЦІ ПРОГРАМНИХ ПРОДУКТІВ

2.1. Модель процесу виконання задач при розробці програмного забезпечення

Проектну команду у процесі розробки програмного забезпечення можна розглядати як систему масового обслуговування (СМО), на вхід якої надходить потік завдань, на виході виходить потік виконаних завдань.

Через аналіз графа завдань у проекті, будується загальний перелік завдань із інформацією зв'язки з-поміж них і пріоритетах, який сортується за пріоритетами упорядкування плану ітерацій, тобто на кожну ітерацію завдання відбираються із загальної черги завдань, які максимально деталізовані для спрощення їхньої оцінки та планування.

Єдинис вважатимемо завдання, яке один фахівець може виконати за один робочий день. Для визначення кількості завдань ітерацію основним критерієм є їх трудомісткість, тобто облік того, скільки ресурсів необхідне для їх виконання. Виходячи з цього, завдання може характеризуватись деякою кількістю поодиноких завдань, тобто складність завдання визначається кількістю поодиноких завдань, на які вона може бути умовно розбита.

Система масового обслуговування, що представляє команду розробки, містить N каналів обслуговування, які відповідають N членам колективу. Характер випадкового процесу виконання завдань членами команди такий, що ймовірність будь-якого стану системи у майбутньому залежить тільки від поточного стану системи та не залежить від того, коли і яким чином система прийшла до цього стану. З якою ймовірністю і за який час будуть виконані наступні завдання, не залежить від того, як було виконано попередні. Виходячи з цього, процес, що моделюється, можна вважати марківським.

До закінчення ітерації мають бути виконані всі завдання. Це частково аналогічно до вирішення деяких відомих завдань, наприклад, моделювання відображення нальоту групи літаків системою ППО, високоорганізованого бою, задачі про ескадру, тобто завданням, в основі вирішення яких лежить у деякому вигляді модель загибелі та розмноження або модель загибелі. На відміну від описаних у літературі моделей, у разі прогнозування вирішення завдань при розробці програмного забезпечення необхідно враховувати, по-перше, можливості одночасного виконання кількох завдань, по-друге, необхідність виконання командою як мінімум кількості поставлених завдань на ітерацію. Тобто команда має за час ітерації виконати щонайменше поставлені завдання. Якщо команда достроково завершила всі роботи з поточних завдань, їх набір на ітерацію може бути розширений.

Важливо, що використання граничного режиму роботи СМО, як у вирішенні аналогічних завдань, у разі, неможливо. Необхідно визначити залежність ймовірностей виконання заданої кількості завдань від часу, а не розрахувати стаціонарний режим роботи СМО, здійснивши граничний перехід при t до нескінченності, тим самим замінивши систему диференціальних рівнянь системою лінійних рівнянь.

Математична модель процесу розробки програмного забезпечення, як модель будь-якої системи масового обслуговування, включає такі основні елементи: потік повідомлень, що надходять, систему обслуговування і дисципліну обслуговування.

Як потік вхідних повідомлень у разі розглядається черга одиничних завдань. Системою обслуговування в даному випадку є команда розробки, яка включає N приблизно однакових за рівнем фахівців: програмістів, інженерів з тестування, аналітиків і т.д. У багатьох компаніях завдання різних фахівців часто виконує одна людина, зазвичай програміст, тому моделі не поділяються на види фахівців. На основі даних про раніше виконані проекти розраховується середня інтенсивність виконання завдань командою – λ. Значення λ перераховується після завершення кожної ітерації

проекту та дозволяє адаптувати систему до поточного стану команди розробки, відображати зміну кількості та рівня підготовки фахівців. Цей параметр дозволяє враховувати індивідуальні професійні можливості фахівців. У разі появи нового члена команди середня інтенсивність виконання завдань командою може бути розрахована за такою формулою:

$$\lambda = (\lambda_{old} + \lambda_{newcomer})/(N_{old} + 1) \quad (2.1)$$

де λ_{old} – поточна інтенсивність виконання завдань командою, $\lambda_{newcomer}$ – інтенсивність праці нового члена команди, N_{old} – кількість розробників у проекті.

Після завершення чергової ітерації чи всього проекту:

$$\lambda = k/t \quad (2.2)$$

де k – кількість виконаних завдань під час ітерації, t – кількість днів у ітерації.

На її основі виходять $\lambda_1, \lambda_2, \dots, \lambda_N$– інтенсивності переходів між станами залежно кількості членів команди.

Процес розробки є СМО з очікуванням без втрат, тобто всі поставлені завдання у результаті мають бути виконані. Заявки обслуговуються в порядку черги, яка може бути отримана під час упорядкування плану ітерацій з урахуванням вихідних завдань, тобто всі зв'язки між завданнями, їх складність та пріоритети враховуються при створенні черги поодиноких завдань.

Головною характеристикою якості обслуговування у такому разі є можливість виконання під час ітерації всіх завдань, запланованих на ітерацію. Додатковими характеристиками можуть бути:

1. Імовірність виконання k завдань у s–й день, де k – будь-яка кількість завдань із черги, s – будь-який день у межах термінів ітерації або за ним.

2. Оптимальна кількість завдань на ітерацію – k_{opt}, вибрана при заздалегідь заданій ймовірності успішного виконання ітерації.

3. Оптимальна кількість днів для вирішення вибраної кількості задач t_{iter_opt}, вибрана з урахуванням цієї ймовірності.

Основним інструментом дослідження є метод рівнянь імовірності станів. У системі масового обслуговування стан характеризується кількістю виконаних завдань. За виконання чергового завдання система змінює свій стан. Інтенсивності переходу з одного стану до іншого визначаються середньою інтенсивністю виконання завдань командою та кількістю членів команди.

Так як команда розробників складається з N спеціалістів, то якщо n фахівців виконають свої завдання одночасно (тобто протягом одного дня виконають поставлені перед ними поодинокі завдання), система може перейти з поточного стану S_k не тільки стан S_{k+1}, але і в будь-який із станів S_{k+n}, де $n \leq N$. Тому в даному випадку необхідно розглядати не тільки ймовірності переходів між сусідніми станами («виконано лише одне завдання»), але й усі ймовірності можливих переходів у сусідні стани («виконано n задач »). Це дозволяє побудувати розмічений граф станів та скласти систему рівнянь, що пов'язують між собою ймовірність сусідніх станів.

У результаті розв'язання системи рівнянь визначаються функції залежності ймовірності знаходження системи у кожному стані від часу, що дозволяють обчислити необхідні характеристики системи.

2.2. Метод розрахунку ймовірності виконання задач

Для побудови загального графа станів системи спочатку розглянемо графи станів одного розробника, двох, трьох і N. У цьому основою роботи

модуля покладено динамічна модель СМО, переходи між станами у якій визначаються залежно від вхідних параметрів.

Незалежно від кількості фахівців у команді система може перебувати в одному з $K + 1$ станів:

S_0 - початок ітерації, не виконано жодного завдання.

S_1 – виконано одне завдання.

S_2 – виконано дві задачі.

...

S_m – виконаний m завдань, тобто виконано всі обов'язкові завдання, заплановані на проект чи ітерацію.

S_{m+1} – виконано $m + 1$ завдання, тобто виконані всі обов'язкові завдання, заплановані на проект або ітерацію та ще одне необов'язкове, крім цього.

...

S_k – виконано k завдань, де k – будь-яке невід'ємне число, оскільки черга завдань не обмежена.

K – загальна кількість усіх завдань на проекті чи ітерації.

На початку ітерації система перебуває у стані S_0. У будь-який момент часу кожен член команди може виконати завдання, над яким він працює, у цьому випадку система переходить у наступний стан – S_{k+1}, або ще не завершити її, тоді система залишається в поточному стані (рисунок 3.1).

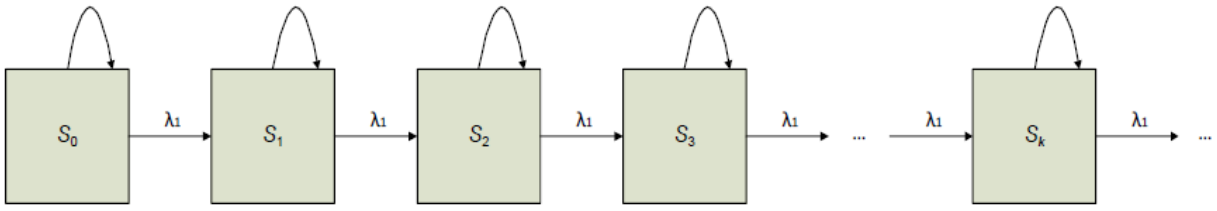

Рис. 3.1. Граф станів системи з одним фахівцем

λ_1 – інтенсивність виконання одного завдання командою, яка в даному разі дорівнює середньому значенню інтенсивності виконання завдань командою λ.

Якщо в команді розробки більше одного спеціаліста, то людина може завершити завдання одночасно, у цьому випадку кількість виконаних завдань збільшиться не на одну, а на кількість осіб, які завершили свої завдання. У такій системі можливі переходи не тільки в наступне за поточним стан, а $i + j, j \leq N$, де i – поточний стан системи, j – кількість фахівців, які одночасно завершили завдання.

На рисунках 2.2 та 2.3 представлені графи станів системи у разі наявності двох та трьох членів у команді розробки.

У разі наявності двох, трьох та більше фахівців у команді розробки інтенсивність переходу між сусідніми станами, тобто виконання командою одного завдання, $\lambda_1 = \lambda_1/\lambda$, аналогічно випадку з одним членом команди, і дорівнює середній інтенсивності виконання завдань усією командою.

Інтенсивність переходу зі стану S_0 у стан S_2, тобто виконання командою двох завдань, $\lambda_2 = \lambda/2$ аналогічно і для переходу зі стану 0 у стан 3 у разі наявності трьох спеціалістів у команді, $\lambda_3 = \lambda/3$. Тобто λ_2 для випадку з двома фахівцями таλ_3 для випадку з трьома фахівцями відповідають середній інтенсивності виконання завдань одним фахівцем.

У випадку для N фахівців у команді розробки граф станів має вигляд, наведений на рисунку 3.2.

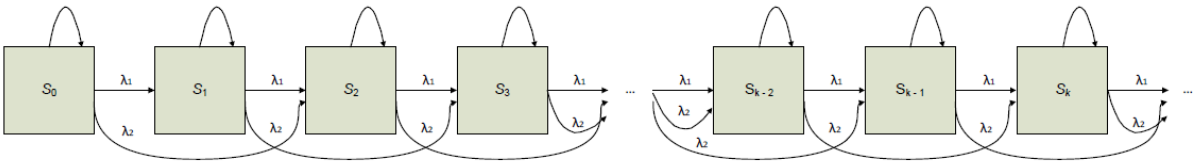

Рис. 3.2. Граф станів системи з двома фахівцями

За час Δt система обслуговування у кожний момент часу може виконати від 0 до N завдань, де N – кількість фахівців у команді, отже перейти в один із станів від i, в якому вона знаходиться, до $i + N$.

При цьому інтенсивності $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_N$ розраховуються за формулою:

$$\lambda_i = \lambda / i \tag{2.3}$$

На основі графа станів для всіх випадків можуть бути складені рівняння ймовірностей перебування у кожному стані.

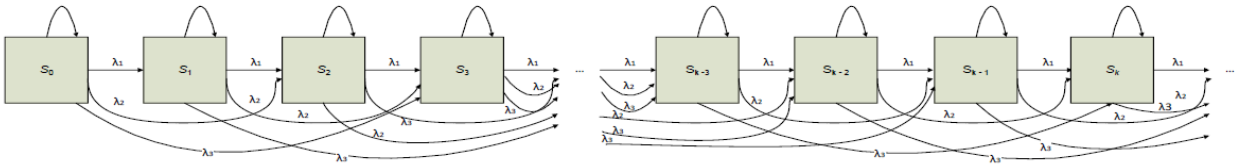

Рис. 3.3. Граф станів системи з трьома фахівцями

З графів станів, можна отримати рівняння станів. Так для випадку, коли команда розробки складається з одного фахівця (рисунок 3.1), ймовірність того, що система через час Δt перебуватиме в стані S_0, визначається як ймовірність того, що система не перейде зі стану S_0 у стан S_1, тобто

$$p_0(t + \Delta t) = p_0(t)(1 - \lambda_1 \Delta t) \tag{2.4}$$

Імовірність того, що через час Δt система перебуватиме в стані S_1 складається з ймовірності переходу зі стану S_0 в стани S_1 і ймовірність того, що система вже знаходилася в стані S_1 і воно не змінилося:

$$p_1(t + \Delta t) = p_1(t)(1 - \lambda_1 \Delta t) + p_0(t)\lambda_1 \Delta t \tag{2.5}$$

Аналогічно

$$p_k(t+\Delta t) = p_k(t)(1-\lambda_1 \Delta t) + p_{k-1}(t)\lambda_1 \Delta t \tag{2.6}$$

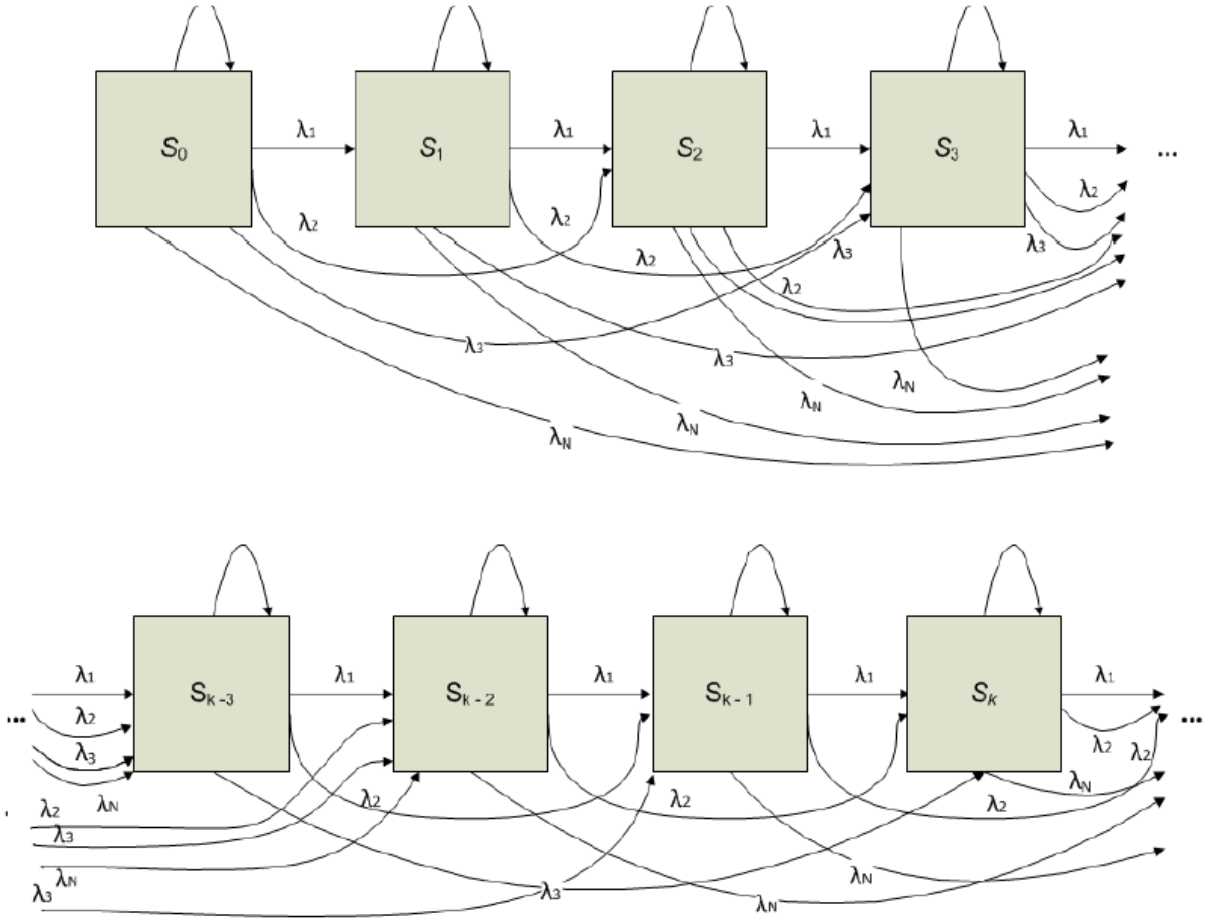

Рис. 3.4. Граф станів системи з N фахівцями

Таким чином, для всіх станів системи виходить система рівнянь:

$$\begin{cases} p_0(t+\Delta t) = p_0(t)(1-\lambda_1 \Delta t) \\ p_1(t+\Delta t) = p_1(t)(1-\lambda_1 \Delta t) + p_0(t)\lambda_1 \Delta t \\ \dots \\ p_k(t+\Delta t) = p_k(t)(1-\lambda_1 \Delta t) + p_{k-1}(t)\lambda_1 \Delta t \end{cases} \tag{2.7}$$

Після перетворень система рівнянь набуває вигляду:

$$\begin{cases} \frac{p_0(t+\Delta t)-p_0(t)}{\Delta t} == -p_0(t)\lambda_1 \\ \frac{p_1(t+\Delta t)-p_1(t)}{\Delta t} = -p_1(t)\lambda_1 + p_0(t)\lambda_1 \\ \dots \\ \frac{p_k(t+\Delta t)-p_k(t)}{\Delta t} = -p_k(t)\lambda_1 + p_{k-1}(t)\lambda_1 \end{cases} \quad (2.8)$$

При $\Delta t \to 0$

$$\begin{cases} \frac{dp_{0(t)}}{d\Delta t} == -p_0(t)\lambda_1 \\ \frac{dp_1(t)}{d\Delta t} = -p_1(t)\lambda_1 + p_0(t)\lambda_1 \\ \dots \\ \frac{dp_k(t)}{d\Delta t} = -p_k(t)\lambda_1 + p_{k-1}(t)\lambda_1 \end{cases} \quad (2.9)$$

Для випадку коли команда розробки складається з двох фахівців (рисунок 3.2), ймовірність того, що система через час Δt перебуватиме в стані S_0, визначається як ймовірність того, що система не перейде з стану S_0 ні стан S_1, ні S_2, тобто

$$p_0(t+\Delta t) = p_0(t)(1-\lambda_1\Delta t - \lambda_2\Delta t) \quad (2.10)$$

Імовірність того, що через час Δt система перебуватиме в стані S_1 складається з ймовірності переходу зі стану S_0 в стани S_1 і ймовірність того, що система вже знаходилася в стані S_1 і воно не змінилося:

$$p_1(t+\Delta t) = p_1(t)(1-\lambda_1\Delta t - \lambda_2\Delta t) + p_0(t)\lambda_1\Delta t \quad (2.11)$$

Аналогічно

$$p_2(t+\Delta t) = p_2(t)(1-\lambda_1\Delta t - \lambda_2\Delta t) + p_1(t)\lambda_1\Delta t + p_0(t)\lambda_2\Delta t \quad (2.12)$$

$$p_k(t+\Delta t) = p_k(t)(1-\lambda_1\Delta t - \lambda_2\Delta t) + p_{k-1}(t)\lambda_1\Delta t + p_{k-1}(t)\lambda_2\Delta t \quad (2.13)$$

Таким чином, для всіх станів системи виходить система рівнянь:

$$\begin{cases} p_0(t+\Delta t) = p_0(t)(1-\lambda_1\Delta t-\lambda_2\Delta t) \\ p_1(t+\Delta t) = p_1(t)(1-\lambda_1\Delta t-\lambda_2\Delta t)+p_0(t)\lambda_1\Delta t \\ p_2(t+\Delta t) = p_2(t)(1-\lambda_1\Delta t-\lambda_2\Delta t)+p_1(t)\lambda_1\Delta t+p_0(t)\lambda_2\Delta t \\ \dots \\ p_k(t+\Delta t) = p_k(t)(1-\lambda_1\Delta t-\lambda_2\Delta t)+p_{k-1}(t)\lambda_1\Delta t+p_{k-1}(t)\lambda_2\Delta t \end{cases} \quad (2.14)$$

Система рівнянь Колмогорова для цього випадку має вигляд:

$$\begin{cases} \frac{dp_{0(t)}}{d\Delta t} == p_0(t)(\lambda_1+\lambda_2) \\ \frac{dp_1(t)}{d\Delta t} = p_1(t)(\lambda_1+\lambda_2)+p_0(t)\lambda_1 \\ \frac{dp_2(t)}{d\Delta t} = p_2(t)(\lambda_1+\lambda_2)+p_1(t)\lambda_1+p_0(t)\lambda_2 \\ \dots \\ \frac{dp_k(t)}{d\Delta t} = p_k(t)(\lambda_1+\lambda_2)+p_{k-1}(t)\lambda_1+p_{k-2}(t)\lambda_2 \end{cases} \quad (2.15)$$

Для загального випадку, коли команда розробки складається з фахівців N (рисунок 3.4), ймовірність того, що система через час Δt перебуватиме в стані S_0, визначається як ймовірність того, що система не перейде з стану S_0 ні стан S_1, …, S_n, тобто

$$p_0(t+\Delta t) = p_0(t)(1-\lambda_1\Delta t-\lambda_2\Delta t-\cdots\lambda_N\Delta t) \quad (2.16)$$

Імовірність того, що через час Δt система перебуватиме в стані S_1 складається з ймовірності переходу зі стануS_0 в стани S_1 і ймовірність того, що система вже знаходилася в стані S_1 і воно не змінилося:

$$p_1(t+\Delta t) = p_1(t)(1-\lambda_1\Delta t-\lambda_2\Delta t-\cdots-\lambda_N\Delta t)+p_0(t)\lambda_1\Delta t \quad (2.17)$$

$$p_2(t+\Delta t) = p_2(t)(1-\lambda_1\Delta t-\lambda_2\Delta t-\cdots-\lambda_N\Delta t)+$$

$$+p_1(t)\lambda_1\Delta t + p_0(t)\lambda_2\Delta t \tag{2.18}$$

$$p_N(t+\Delta t) = p_N(t)(1-\lambda_1\Delta t - \lambda_2\Delta t - \cdots - \lambda_N\Delta t) + p_0(t)\lambda_1\Delta t + \\ +p_1(t)\lambda_{N-1}\Delta t + \cdots + p_{N-1}(t)\lambda_1\Delta t \tag{2.19}$$

$$p_k(t+\Delta t) = p_k(t)(1-\lambda_1\Delta t - \lambda_2\Delta t - \cdots - \lambda_N\Delta t) + p_{k-1}(t)\lambda_N\Delta t + \\ +p_1(t)\lambda_{k-N-1}\Delta t + \cdots + p_{k-1}(t)\lambda_1\Delta t \tag{2.20}$$

Виходячи з цих рівнянь, система рівнянь Колмогорова для загального випадку має вигляд:

$$\begin{cases} \frac{dp_{0(t)}}{d\Delta t} = -p_0(t)(\lambda_1+\lambda_2+\cdots+\lambda_N) \\ \frac{dp_1(t)}{d\Delta t} = -p_1(t)(\lambda_1+\lambda_2+\cdots+\lambda_N) + p_0(t)\lambda_1 \\ \frac{dp_2(t)}{d\Delta t} = -p_2(t)(\lambda_1+\lambda_2+\cdots+\lambda_N) + p_1(t)\lambda_1 + p_0(t)\lambda_2 \\ \cdots \\ \frac{dp_k(t)}{d\Delta t} = -p_k(t)(\lambda_1+\lambda_2+\cdots+\lambda_N) + \\ p_{k-N}(t)\lambda_N + p_1(t)\lambda_{k-N-1} + \cdots p_{k-1}(t)\lambda_1 \end{cases} \tag{2.21}$$

Привівши рівняння до загального вигляду, отримуємо систему рівнянь Колмогорова для загального випадку в залежності від кількості фахівців в команді розробки:

$$\begin{cases} \frac{dp_{0(t)}}{d\Delta t} = -p_0(t)\sum_{i=1}^{N}\lambda_i \\ \frac{dp_1(t)}{d\Delta t} = -p_1(t)\sum_{i=1}^{N}\lambda_i + p_0(t)\lambda_0 \\ \cdots \\ \frac{dp_k(t)}{d\Delta t} = -p_k(t)\sum_{i=1}^{N}\lambda_i + \sum_{i=1}^{N}p_{k-1}(t)\lambda_1; \\ \sum_{k=0}^{K}p_k = 1 \end{cases} \tag{2.22}$$

Функції ймовірностей станів системи. Розв’язок системи рівнянь Колмогорова є функції залежності ймовірності знаходження системи у вказаному стані від часу. Рішенням отриманої системи диференціальних рівнянь для випадку з одним членом команди є наступні функції, отримані засобами Maple:

$$\begin{cases} p_0(t) = e^{-\lambda_1 t} \\ p_1(t) = \lambda_1 t e^{-\lambda_1 t} \\ p_2(t) = \dfrac{{\lambda_1}^2 t^2 e^{-\lambda_1 t}}{2} \\ \dots \\ p_k(t) = \dfrac{(\lambda_1 t)^k e^{-\lambda_1 t}}{k!} \end{cases}$$

Виходячи з цього, для найпростішого випадку з одним фахівцем залежність ймовірності розв'язання заданої кількості завдань від часу можливо визначити за формулою:

$$p_i(t) = \frac{(\lambda_1 t)^i e^{-\lambda_1 t}}{i!}$$

У складніших випадках для системи з великою кількістю фахівців у команді, а також у разі великої кількості завдань на ітерацію в загальному випадку для визначення ймовірностей знаходження системи в i–ому стані доцільно використовувати чисельний метод для вирішення системи рівнянь Колмогорова, методом Рунге-Кути в середовищі Maple 16.

Після вирішення системи рівнянь та визначення ймовірностей знаходження системи у заданий час t у кожному зі станів, тобто ймовірностей вирішення командою нуля, однієї, двох, трьох, ..., N задач, необхідно визначити ймовірність виконання не менш заданої кількості

завдань у вибраний момент часу $P_i(t)$ – саме це є результатом успішного завершення ітерації чи проекту. Тобто

$$P_i(t) = P_i(t) + P_{i+1}(t) + P_{i+2}(t)$$

або

$$P_i(t) = 1 - \sum_{j=0}^{i-1} P_j(t)$$

Таким чином можуть бути отримані значення ймовірностей для будь-якої кількості завдань у будь-який момент часу і, відповідно визначено ймовірність виконання у строк усіх N завдань, запланованих на ітерацію.

У разі незадовільного прогнозу, системою можуть бути запропоновані оптимальна кількість завдань на ітерацію для заданих параметрів системи k_{opt} та оптимальна кількість днів для вирішення обраної кількості задач d_{opt}.

Порогове значення визначення прийнятності плану на ітерацію за замовчанням дорівнює 0.9. Якщо ймовірність виконання завдань на ітерацію в строк менше цього значення, то може бути визначено оптимальну кількість задач, при якому ймовірність успішного завершення ітерації буде не меншою за порогове значення. Нижче наводиться код для обчислення цього параметра:

```
for k from 6 by -1 to 1
do
     if Ps[k] > .9 then
          print(k);
          print(Ps[k]);
          break
     end if
end do;
```

Також прогнозується час, через який буде з достатньою ймовірністю (більше порогового значення) виконано всі заплановані на ітерацію завдання:

```
for k to 5
do
     if Ds[k] > .9 then
          print(k);
          print(Ds[k]);
          break
     end if
end do;
```

Для визначення ймовірності виконання k завдань під час ітерації d необхідно вирішити систему $k+1$ диференціального рівняння. Через рішення складеної системи рівнянь методом Рунге–Кути в середовищі Maple отримуємо значення ймовірностей знаходження системи у кожному зі станів, тобто розв'язання рівно заданої кількості завдань, у кожен із днів ітерації, які наводяться в таблиці 2.1.

На основі отриманих значень ймовірностей знаходження системи у кожному стані розраховуються ймовірності виконання не менше заданої кількості завдань у кожен із днів ітерації. Результати розрахунку наводяться у таблиці 2.1.

Таблиця 2.1

Результати розрахунку ймовірностей знаходження системи у кожному стані

t, дн.	$p(t)$						
	0	1	2	3	4	5	6
1	0.1599	0.1599	0.1599	0.1599	0.1199	0.0879	0.0613
2	0.0256	0.0511	0.0767	0.1022	0.1150	0.1175	0.1115
3	0.0041	0.0123	0.0245	0.0409	0.0582	0.0742	0.0866
4	0.0007	0.2614	0.0065	0.0131	0.0222	0.0335	0.0458

Таблиця 2.2

Результати розрахунку ймовірностей знаходження системи у кожному стані

t, дн.	*P*(*t*)					
	1	2	3	4	5	6
1	0.6802	0.5203	0.3604	0.2405	0.1526	0.0931
2	0.9234	0.8467	0.7445	0.6295	0.5120	0.4005
3	0.9836	0.9591	0.9183	0.8600	0.7858	0.6992
4	0.9967	0.9902	0.9771	0.9549	0.9214	**0.8756**

З даних, наведених у таблиці 3.2 видно, що ймовірність виконання вчасно всіх запланованих завдань незадовільна, оскільки менше порогового значення (0.8765 < 0.9). Тому проводиться розрахунок оптимальної кількості завдань, які можуть бути виконані за відведений на ітерацію час із достатньою ймовірністю.

```
for k from 6 by -1 to 1
do
     if Ps[k] > .9 then
          print(k);
          print(Ps[k]);
          break
     end if
end do;
```

На практиці часто необхідно вирішити всі поставлені завдання, тому важливим є отримання прогнозу, за який час можуть бути виконані всі поставлені задачі із задовільною ймовірністю.

```
for k to 5 do
     if Ds[k] > .9 then
          print(k);
          print(Ds[k]);
          break
     end if
end do;
```

Таким чином, для даної команди розробників виконання всіх поставлених завдань на ітерацію можливе з ймовірністю 87,7%, з достатньою ймовірністю 91,8% можливе виконання 3 завдань за відведений час ітерації

(3 дні) або виконання всіх поставлених завдань за 5 днів з ймовірністю 95,5%.

Висновки до другого розділу

1. Команда розробки програмного забезпечення сприймається як система масового обслуговування. Як основні характеристики такої системи масового обслуговування вибрано ймовірність вирішення заданої кількості завдань у вибраний день, допустиму кількість завдань на ітерацію з урахуванням порогового значення ймовірності успішного її виконання, а також кількість днів для вирішення обраної кількості задач.

2. Запропоновано динамічну модель процесу виконання завдань командою розробки програмного забезпечення, що дозволяє в залежності від кількості членів команди автоматично змінювати модель системи масового обслуговування та чисельним методом Рунге-Кути перераховувати залежність ймовірностей виконання заданої кількості завдань від поточного дня ітерації.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ ВИКОНАННЯ ЗАВДАНЬ ПРИ РОЗРОБЦІ ПРОГРАМНИХ ПРОДУКТІВ

3.1. Загальна архітектура системи

Для розробки програмного засобу, що задовольняє вимогам, потрібно врахувати ряд особливостей проектів розробки ПЗ, тому система підтримки прийняття рішень при плануванні проекту створення ПЗ включає (рисунок 3.1):

Модуль постановки завдань. Представлений інтерфейсом користувача для ведення проекту, створення та підтримки списку завдань, визначення їх трудомісткості, взаємозв'язків та пріоритетів, завдання команди розробки, її характеристик.

Модуль аналізу задач. Включає модуль побудови графа завдань і модуль створення черги. На основі заведених завдань щодо проекту, їх зв'язків та пріоритетів у модулі будується та аналізується граф задач.

Після цього в модулі створення черги на основі побудованого графа задач створюється черга завдань, яка згодом перетворюється на чергу одиночних завдань.

Модуль прогнозування ітерацій проекту. Включає модуль складання плану ітерацій, в якому на основі даних про проект і команду розробки, визначених в модулі постановки задачі, і отриманої черги одиничних завдань автоматично складається план ітерацій.

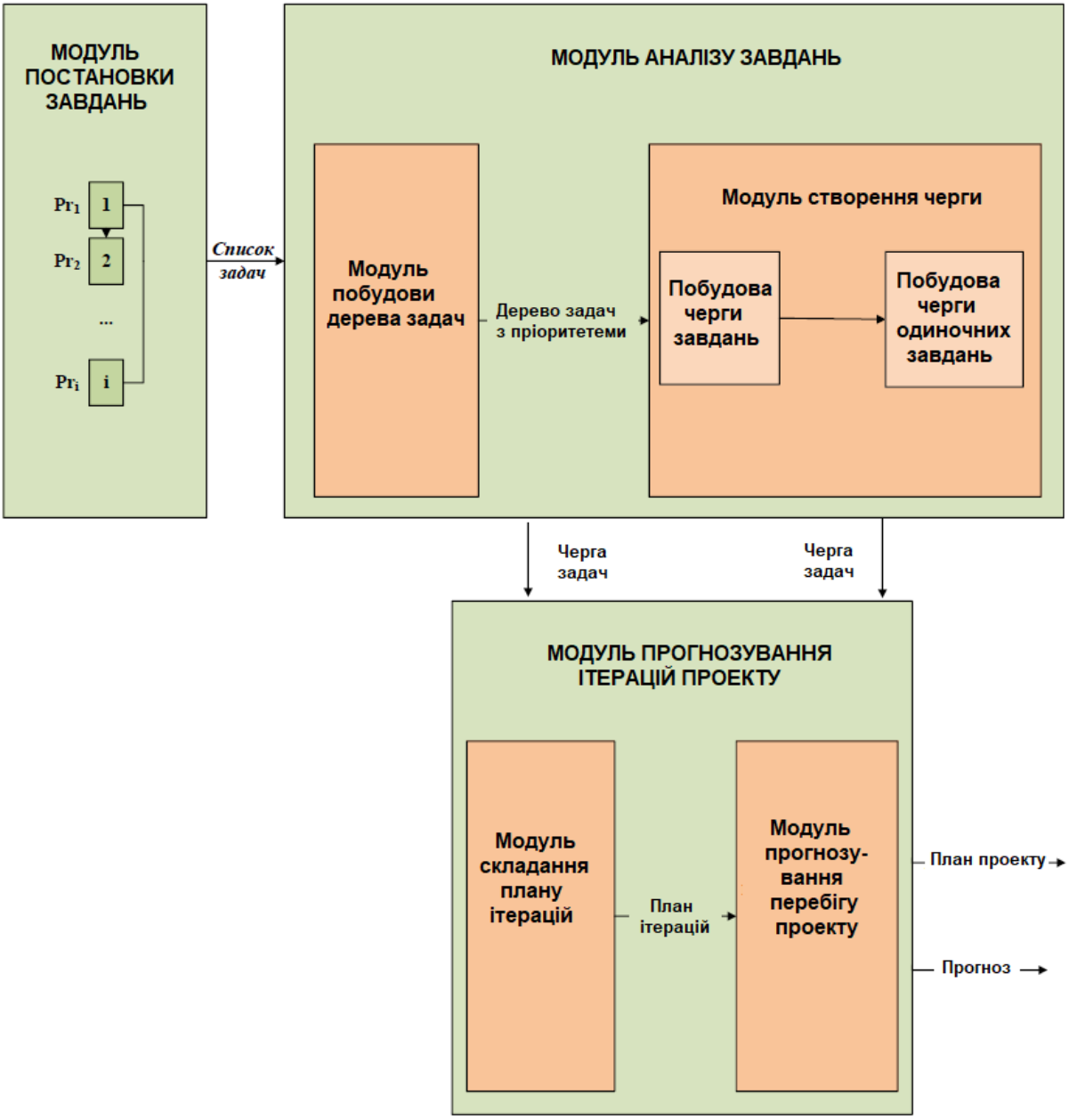

Рис. 3.1. Архітектура програмної системи

На його основі в модулі прогнозування ходу виконання проекту складається прогноз успішності виконання поставлених завдань у задані терміни, у разі незадовільного прогнозу видаються рекомендації щодо оптимальної кількості завдань та часу виконання поставлених завдань.

Далі докладно розглянуто основні модулі.

Модуль постановки завдань. Після визначення загальних вимог робота над проектом починається з уточнення вимог та постановки завдань.

Модуль постановки завдань надає користувачеві можливість ведення списку завдань за проектами та списком команд розробки. При додаванні нового проекту користувач задає його назву, характеристики, вимоги, терміни, основну документацію та переходить до додавання завдань щодо проекту.

На цьому етапі користувачів може бути кілька, тобто, як і в будь-якій системі управління проектами, вносити до списку нові завдання можуть різні люди.

Для створення нового завдання необхідно, окрім її назви, опису та вказівки, до якого проекту вона належить, задати її пріоритет та трудомісткість, що вимірюється в поодиноких завданнях. Тобто складність завдання визначається кількістю поодиноких завдань, на які вона може бути умовно розбита.

Крім цього, користувач може вказати пов'язані між собою завдання, що виконуються послідовно, або підзавдання. Можливе також ведення списку команд розробки, завдання команди кожного проекту, управління складом команди, створення нових команд.

При додаванні нового фахівця до команди необхідно вказати йому інтенсивність виконання завдань. Ця характеристика може бути отримана після аналізу результатів попередніх проектів, у яких брав участь цей фахівець, або встановлена експертним шляхом. Згодом вона може коригуватися за результатами завершення чергової ітерації чи проекту.

Згодом такі характеристики можуть коригуватися за результатами завершення чергової ітерації чи проекту. Таким чином, модуль постановки завдань забезпечує можливість додавання всієї необхідної інформації щодо проекту, включаючи документацію, завдання та команду розробки.

Модуль аналізу завдань. При плануванні проекту зазвичай на основі загального списку завдань з урахуванням їх зв'язків та пріоритетів керівник проекту складає план виконання завдань. У розробленій системі підтримки

прийняття рішень під час управління проектами цей процес має автоматизуватись.

Список завдань, отриманих від джерел завдань (замовників), аналізується та перетворюється на вид графа, який відповідає списку завдань за одним проектом з урахуванням пов'язаності завдань, підзадач та їх пріоритетів. Причому в системі передбачено використання пріоритетів двох видів: базовий пріоритет задачі та пріоритет джерела.

Пріоритет джерела визначається кожному користувачеві системи залежно від його впливу на проект. Тому пріоритет задачі складається з цих двох характеристик, при цьому в першу чергу враховується базовий пріоритет задачі, а серед рівних за базовим пріоритетом завдань черга будується з урахуванням пріоритету джерела. Через роботу модуля формується попередній граф завдань проекту.

Після побудови він аналізується з точки зору узгодженості завдань щодо їх зв'язків та пріоритетів. Так, наприклад, якщо одна з підзадач має більш високий пріоритет, ніж батьківська або попередні, то їх пріоритет автоматично змінюється на більш високий пріоритет підзадачі. Результатом роботи модуля є сформований графік завдань.

Модуль створення черги завдань. Модуль створення черги призначений для оцінки завдань, декомпозиції їх на більш дрібні зручні для вирішення задачі. У розроблюваної системі декомпозиція проводиться до рівня одиночних задач, у яких вказується трудомісткість завдання під час її створення.

Після проходження етапу оцінки та обробки на вхід модуля створення черги завдань надходить розмічений за вагами та впорядкований по пріоритетах граф завдань проекту.

Даний модуль вирішує задачу рекурсивного обходу графа задач відповідно до пріоритетів та зв'язків. На початок черги потрапляють завдання з вищим базовим пріоритетом, що забезпечує їхнє першочергове виконання.

Усередині списку завдань з однаковим пріоритетом здійснюється сортування за пріоритетом джерела завдання. Залежно від зв'язків між завданнями для кожної з них встановлюється статус, що визначає чи доступне це завдання для вирішення на даний момент часу.

В результаті формується черга завдань щодо проекту з урахуванням усіх пріоритетів та взаємозв'язків (рисунок 3.2).

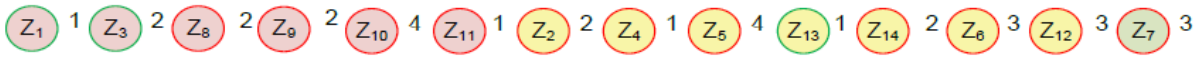

Рис. 3.2. Черга задач проекту

Ще однією функцією даного модуля є перебудова наявної черги під час вступу нових завдань, як із боку замовників проекту, і в результаті тестування проміжних версій ПЗ. Перебудова черги відбувається за таким правилом: знаходиться останній елемент потрібного пріоритету, потім виконується вставка у вже наявну чергу, або початок черги, якщо елементів із цим пріоритетом у черзі не знаходилося раніше.

Після створення черги завдань щодо проекту автоматично формується відповідна їй черга поодиноких завдань (рисунок 3.3), тобто кожне завдання замінюється на кілька одиничних завдань, еквівалентних їй за заданим джерелом трудомісткості, тобто завдання з трудомісткістю M розпадається на M поодиноких завдань, що йдуть поспіль у черзі. Для кожного одиничного завдання зберігається інформація про те, частиною якого завдання вона є.

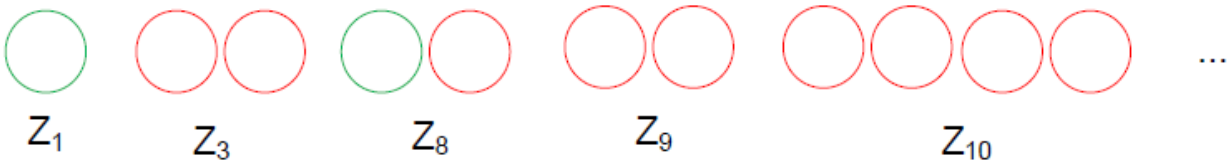

Рис. 3.3. Черга одиночних завдань щодо проекту

При цьому керівник проекту може змінити трудомісткість та порядок виконання завдань, скоригувавши відповідні пріоритети та характеристики трудомісткості.

Модуль прогнозування ітерацій проекту. Модуль складання плану ітерацій. В результаті на вхід модуля прогнозування результатів виконання завдань надходить велика, порівняно з розміром одного етапу виконання проекту, черга поодиноких завдань. Далі в залежності від типу проекту може бути визначено:

• приблизна кількість ітерацій, необхідних для виконання N завдань, при фіксованих рамках проекту. Ця кількість оцінюється як відношення N завдань до середньої кількості, яку колектив може виконати за одну ітерацію.

• приблизна кількість завдань, які необхідно виконати за одну ітерацію при фіксованому часу. Кількість завдань можна оцінити як відношення кількості завдань до кількості допустимих ітерацій проекту.

При цьому розмір ітерації відповідно до обраної ідеології ведення проекту зазвичай фіксується для стандартизації процесів усередині ітерації. Однак у загальному випадку зустрічаються ітерації змінної довжини, тому система дозволяє враховувати їх, оскільки за основу взято завдання одиничного обсягу.

Модуль прогнозування ходу виконання проекту. В основу роботи модуля прогнозування ходу виконання проекту покладено модель процесу виконання задач з розробки ПЗ, в якій команда розробки розглядається як система масового обслуговування.

Вихідними даними для складання прогнозу є черга поодиноких завдань, план ітерацій для одиничних завдань та характеристики команди розробки: інтенсивність виконання завдань та кількість спеціалістів.

План ітерацій включає час і розмір кожної з ітерацій для підтримки роботи із нерівними ітераціями.

У модулі прогнозування результатів виконання завдань, що становлять проект, описана динамічна модель системи масового обслуговування, яка використовується для отримання ймовірностей виконання списку запланованих на ітерацію задач, відображення прогнозу ходу проекту, а також визначення необхідних для ітерації проекту ресурсів.

На вхід модуля прогнозування результатів виконання завдань подаються такі параметри:

• N – кількість спеціалістів у команді;

• k – кількість завдань на ітерацію;

• d – кількість днів в ітерації;

• λ – середня продуктивність команди розробки, завдань на день.

Система може використовувати для прогнозування параметри N, k, d за замовчуванням, вихідні значення яких: $N = 5, k = 20, d = 10$. Користувач може встановити свої значення параметрів, як для конкретного прогнозу, і як значень за умовчанням.

Блок-схема роботи модуля представлена рисунку 3.4. На початку роботи модуля перевіряються вхідні параметри. Якщо N не має значення за умовчанням і не збігається з характеристиками останньої модельованої системи, то залежно від вхідних параметрів визначаються стани системи та формується система відповідних диференціальних рівнянь.

Після розв'язання системи рівнянь визначаються функції залежності ймовірності знаходження системи в i −му стані від часу t.

Якщо N має значення за замовчуванням, то граф і рівняння станів та функції ймовірностей були раніше визначені та зберігаються в пам'яті системи.

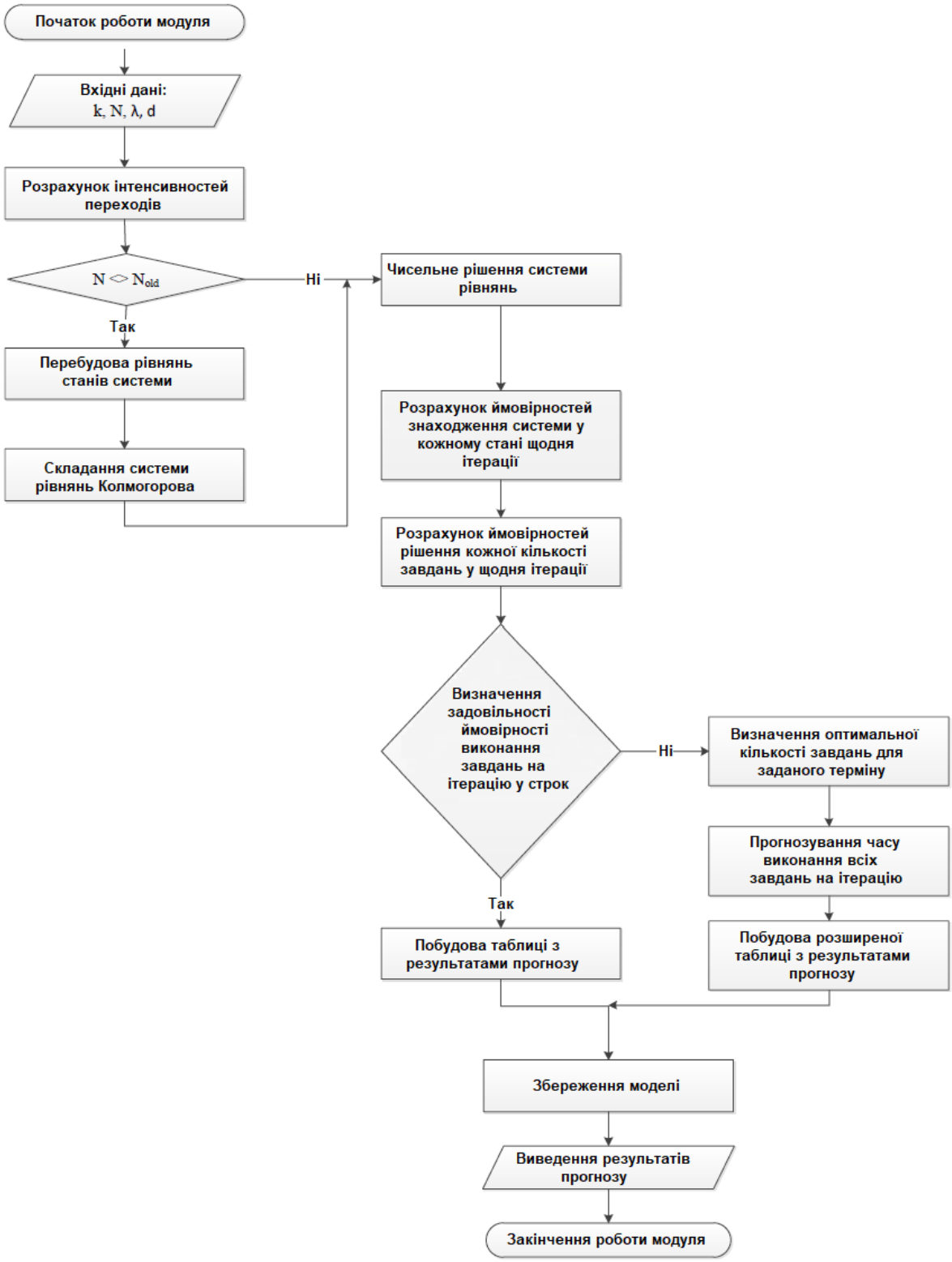

Рис. 3.4. Блок-схема роботи модуля прогнозування виконання завдань

За допомогою отриманих залежностей визначається ймовірність виконання всіх запланованих завдань за ітерацію та ймовірності виконання кожного завдання по всіх днях в ітерації. Якщо ймовірність менше заданого в

системі порогового значення, визначається оптимальна кількість завдань для заданих вхідних параметрів і час закінчення проекту.

На основі цих даних формується загальний прогноз на ітерацію, загальний вигляд якого представлений у таблиці 3.1.

Таблиця 3.1

Загальний вид прогнозу виконання завдань щодо проекту на ітерацію

№ задачі	1	2	3	4	…	K	План на ітерацію
1	p11	p12	p13	p14	…	p1k	**p1**
2	p21	p22	p23	p24	…	p2k	**p2**
3	p31	p32	p33	p34	…	p3k	**p3**
4	p41	p42	p43	p44	…	p4k	**p4**
…	…	…	…	…	…	…	…
d	**pd1**	**pd2**	**pd3**	**pd4**		**pd6**	**pd**
d+1	p(d+1)1	p(d+1)2	p(d+1)3	p(d+1)4	…	p(d+1)k	**p(d+1)**
d+2	p(d+2)1	p(d+2)2	p(d+2)3	p(d+2)4	…	p(d+2)k	**p(d+2)**
…	…	…	…	…	…	…	**…**
$d+d_{доп}$	**$p(d+d_{доп})_1$**	**$p(d+d_{доп})_2$**	**$p(d+d_{доп})_3$**	**$p(d+d_{доп})_4$**			**$p(d+d_{доп})$**

Після цього зберігається модель системи для поточних параметрів та виводяться результати прогнозу. Використовуючи побудовану модель, складається прогноз ходу виконання проекту протягом часу ітерації та визначається ймовірність успішного виконання всіх поставлених завдань за цей термін.

Задовільність прогнозу визначається порівнянням з граничним значенням ймовірності успішного виконання проекту, яке в системі встановлено за умовчанням і дорівнює 0.9 (90%).

Модуль прогнозування ходу виконання проекту може працювати у двох режимах: прогнозування та аналізу. У разі роботи в режимі система робить прогноз (рисунок 3.5), який включає в себе набір поодиноких завдань на ітерацію та кількість днів відповідно до плану ітерацій для вирішення кожної кількості завдань у кожен день ітерації розраховується ймовірність їх виконання, визначається ймовірність виконання всіх поставлених завдань за

термін ітерації та оцінюється чи прийнятний результат прогнозу з урахуванням граничного значення.

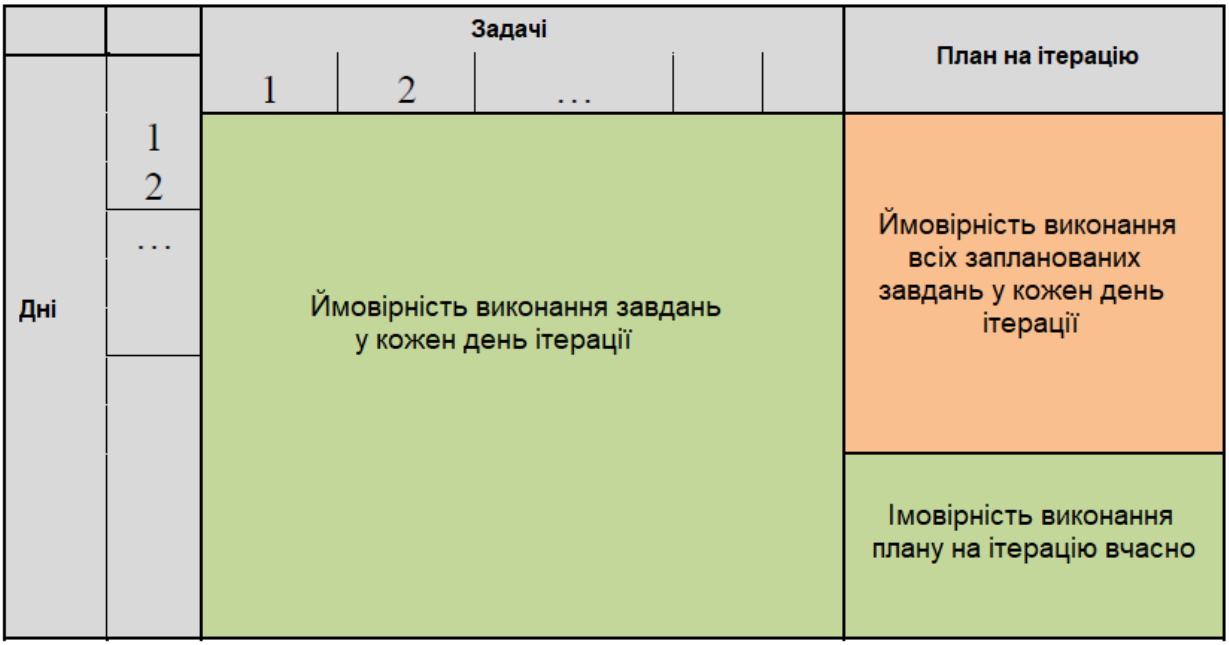

Рис. 3.5. Результат роботи у режимі прогнозування

У разі роботи в режимі аналізу крім складання прогнозу після оцінки задовільності результату прогнозу система пропонує користувачеві рішення щодо оптимізації плану ітерацій.

Якщо ймовірність успішного виконання завдань більша за порогове значення, то система складає розширений прогноз для позитивного випадку та визначає кількість додаткових одиничних завдань, які можна додати в аналізовану ітерацію, за умови, що ймовірність їх успішного виконання залишиться у допуску.

Якщо ймовірність успішного виконання завдань менша за порогове значення, то система будує розширений прогноз для негативного випадку та пропонує змінити можливі параметри:

• збільшити розмір ітерації (кількість днів, необхідних на виконання певної кількості завдань);

• зменшити кількість завдань, залишивши розмір ітерації незмінним. Така методика підтримується більшістю методологій, що належать до сімейства «гнучких»;

• змінити склад команди розробки, додаючи до неї нових спеціалістів. При цьому в системі буде новий розрахунок середньої продуктивності команди та на її основі перебудується модель процесу виконання завдань та буде розрахована скоригована ймовірність виконання N відібраних завдань під час d.

Оптимальна кількість завдань, яку можна з допустимою ймовірністю виконати за час ітерації та оптимальний час виконання поставлених завдань автоматично розраховуються системою. За підсумками них складається розширений варіант прогнозу.

Таким чином, в результаті роботи модуля прогнозування результатів виконання завдань користувачеві системи надається прогноз, складений на основі плану ітерацій, із зазначенням ймовірності виконання кожної їх у термін.

Для врахування того факту, що завдання не завжди відповідають своїй початковій оцінці, а також через появу будь-яких можливих додаткових деталей, не врахованих при початковій оцінці, модуль прогнозування результатів виконання завдань показує поточний статус ітерації, ймовірності виконання завдань до кінця ітерації, а також дозволяє зробити їх перерахунок під час коригування кількості завдань, термінів або розміру команди, аналогічно тому, як це відбувалося перед початком ітерації.

На основі отриманих даних керівник проекту може прийняти отриманий план проекту і перейти до його реалізації або змінювати його доти, доки результати не задовольнять його.

3.2. Інструкція користувача по роботі з системою

На основі розробленого математичного та алгоритмічного забезпечення для зручного перерахунку даних та практичної перевірки запропонованої методики планування при розробці ПЗ була розроблена система, що дозволяє керівникам проектних команд проводити оцінку ймовірності виконання проектів до заданого терміну.

Крім безпосередньо практичної реалізації моделі системи масового обслуговування, відповідно до типової схеми ведення проекту, представленої у другому розділі, реалізовані також такі функції:

Модуль формування команди (розділ Команда, рисунок 3.6).

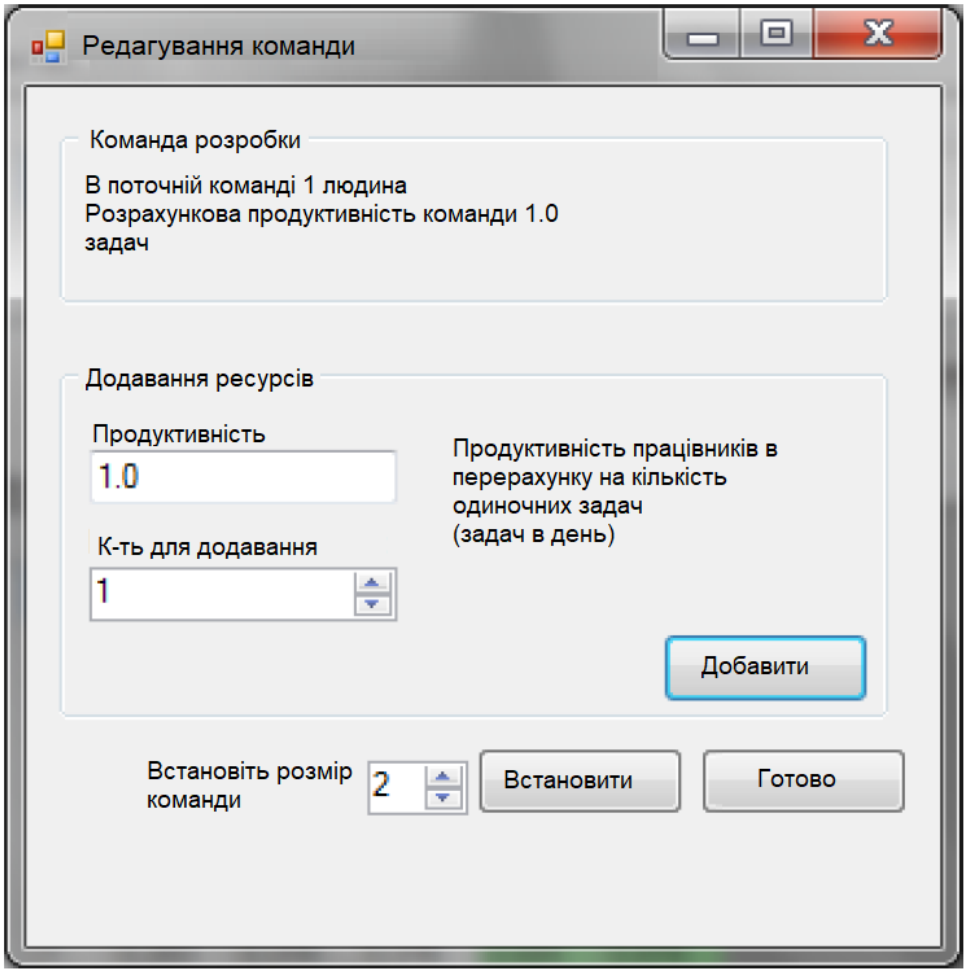

Рис. 3.6. Розділ управління командою

Модуль призначений для завдання інформації про команду розробки, автоматичне коригування відомостей про продуктивність команди, залежно від виконаних завдань, на основі якої перераховується ймовірність виконання інших завдань. А також для ручного коригування складу та продуктивності розробників.

Модуль формування переліку завдань (рисунок 3.7). Як було сказано у другому розділі роботи, завдання проекту є графом. Модуль дозволяє побудувати граф задач, а потім зробити його «обхід», сформувавши список завдань з урахуванням пріоритетів.

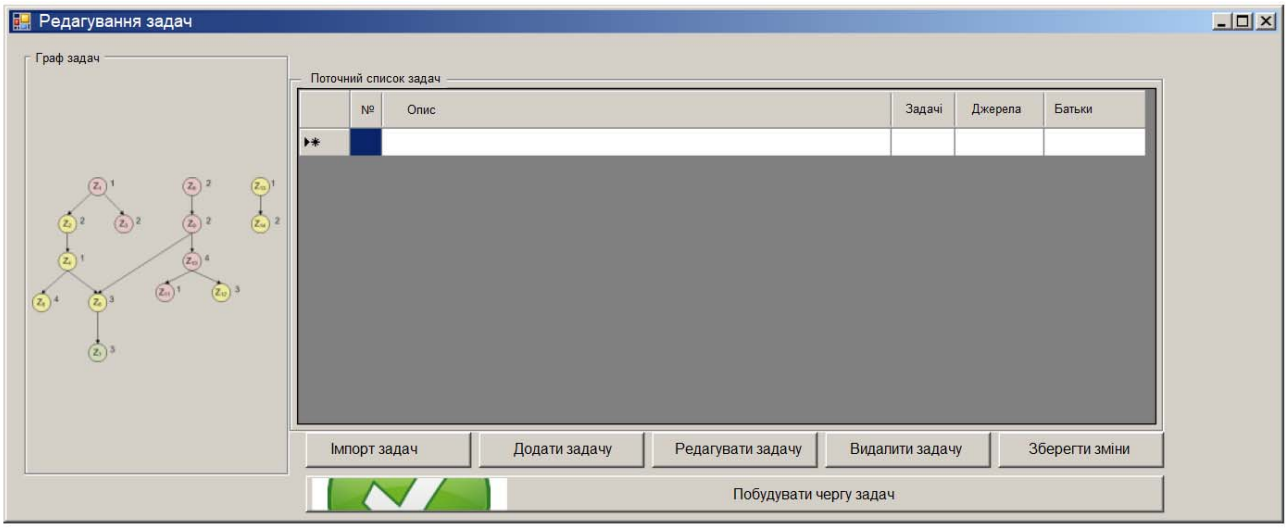

Рис. 3.7. Формування списку завдань

Модуль прогнозування. (Рисунок 3.8) Основний модуль програми. У модулі реалізована пропонована математична модель, за допомогою якої програма розраховує ймовірність виконання заданої кількості завдань за ітерацію, а також пропонує керівнику проекту можливі варіанти підвищення цієї ймовірності, якщо вона нижча від порогової.

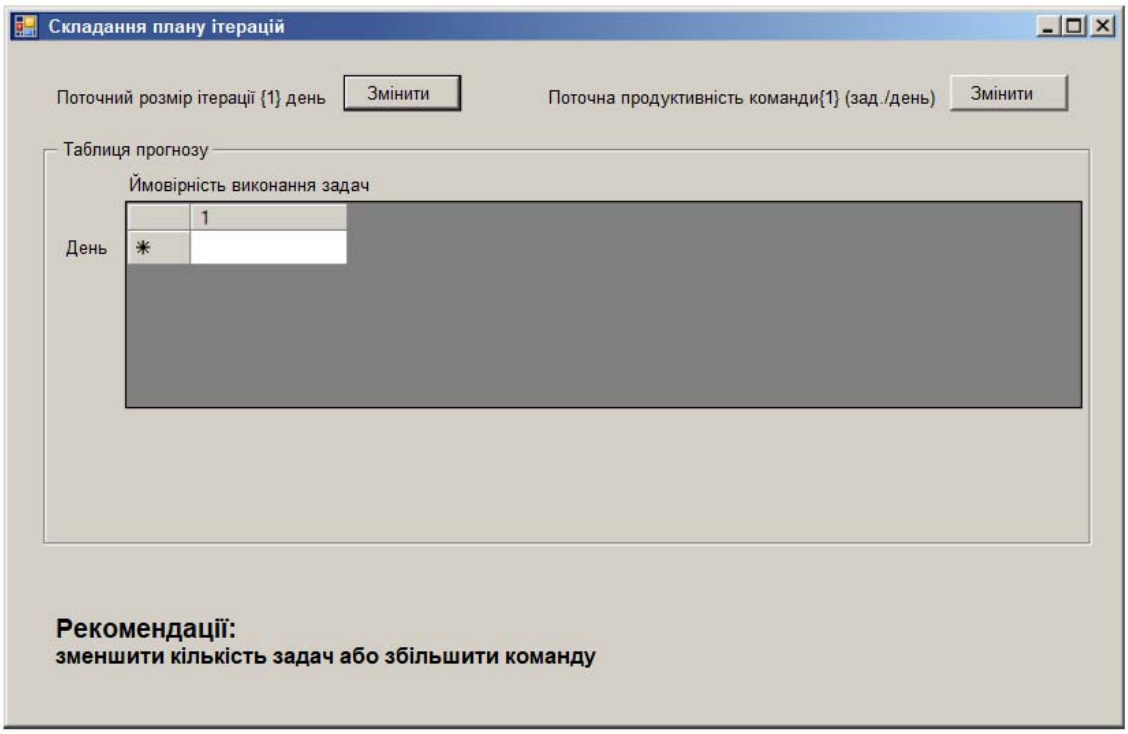

Рис. 3.8. Модуль прогнозування

У найуспішнішому випадку, після внесення завдань за проектом, користувачеві необхідно лише фіксувати виконання завдань, така функція доступна у меню головного вікна системи (рисунок 3.9).

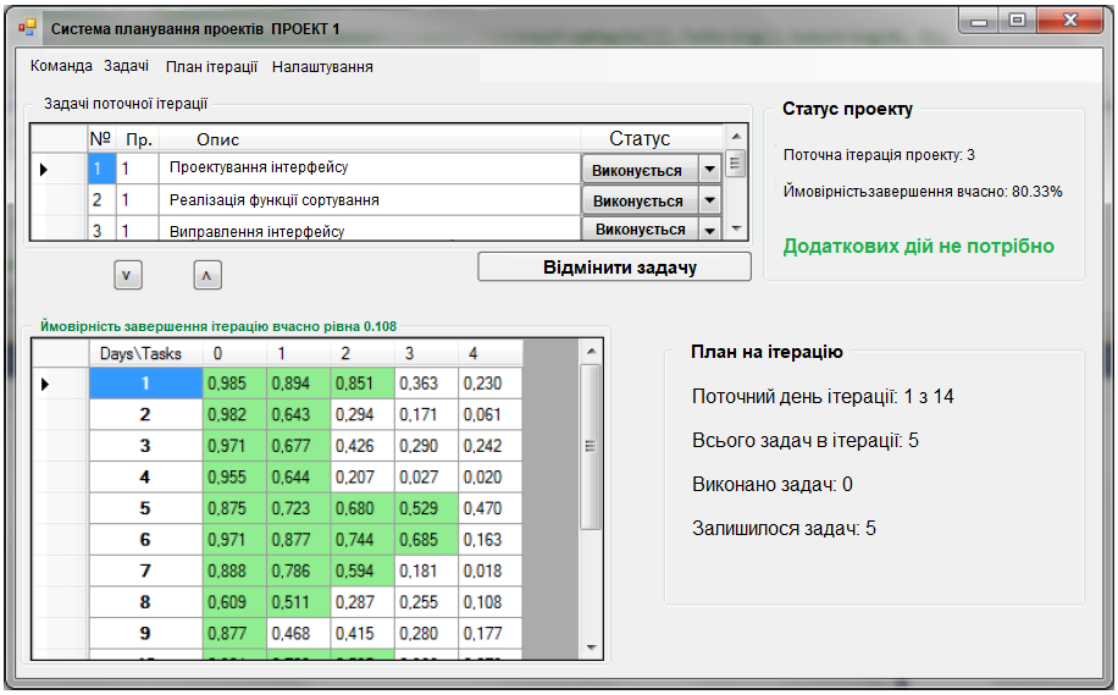

Рис. 3.9. Головне вікно системи

3.3. Прогнозування процесу виконання завдань у проекті розробки системи автоматизованого аналізу текстів

У ході експерименту було виконано планування та реалізація завдань ітерації розробки у проекті створення системи автоматизованого аналізу тексту. Вибір обумовлений тим що розміри проекту дозволяють мати достатній ступінь свободи щодо розбиття завдань на ітерації та відбору завдань;

• проект ведеться вже кілька років, у ньому накопичилося безліч завдань, пов'язаних з розробкою функціоналу, яким вже практично не потрібно проводити додаткові дослідження. Дослідницькі роботи мають суттєву творчу складову, та не можуть бути достовірно оцінені;

• команда розробки проекту складається із спеціалістів різної кваліфікації, що дозволяє простежити залежності можливості виконання завдань при різних конфігураціях команди розробки;

• важлива також відкритість розробки цього проекту, на відміну від більшості комерційних проектів, розкриття деталей яких неможливе через юридичні обмеження.

Система аналізу тексту призначена для обробки текстової інформації, використовує останні напрацювання в галузі програмних засобів, алгоритмів та методів автоматизованого аналізу тексту.

Система має постійно поповнюваний набір гнучких інструментів та можливостей впливу на алгоритми опрацювання текстів. Реалізація системи аналізу тексту та розробка інструментальних засобів інтегрального аналізу тексту дозволять фахівцям різних галузей накопичувати та узагальнювати знання з аналізу тексту, вирішувати різні практичні завдання (автоматичне реферування, класифікація текстів, визначення авторства і т.д.), а також використовувати пропонований програмний продукт для наукових досліджень.

Система аналізу розвивається вже кілька років, за цей час значно збільшився її розмір. Збільшилася кількість завдань, необхідні реалізації, знадобилося залучення

більшого кількості розробників. Також виникла потреба у застосуванні якогось із відомих підходів до розробки програмного забезпечення. Гнучкий підхід дуже добре підходить для інформаційних систем, де вже є базова частина, що включає ядро системи та основні інтерфейси взаємодії між модулями, оскільки в цьому випадку вдається подолати, мабуть, основний недолік «гнучких» підходів, а саме відсутність добре продуманої та документованої архітектури систем, що розробляються. На кожну ітерацію виноситься той чи інший набір функцій, відповідно до потреб користувача, а система цілком залишається «за рамками», і після якоїсь певної точки стає занадто важкою, повною різних і навіть неузгоджених елементів, нездатної для подальшого розвитку. Однак, коли архітектура вже опрацьована, гнучкі підходи показують більшу ефективність за рахунок більш тісної взаємодії всередині команди та із замовником. Тому для подальшого розвитку системи аналізу тексту було прийнято рішення застосовувати гнучкий підхід до розробки, а розраховувати та відбирати завдання кожної ітерації за допомогою запропонованої моделі.

Щоб визначити загальний список завдань у проекті, необхідно спочатку виділити джерела завдань. У результаті аналізу було виділено такі групи людей, які впливають розвиток системи аналізу тексту:

• учасники визначають нові можливості системи з погляду потреб комп'ютерної лінгвістики, ці завдання складають розширення основних функцій системи;

• користувачі, найчастіше такими є працівники, які використовують систему у різних наукових цілях, від них крім завдань щодо функціональності системи надходить великий потік побажань, пропозицій щодо графічного інтерфейсу та зручності використання;

• студенти, задіяні переважно у дослідній експлуатації системи, тестуванні нових функцій.

• архітектори системи, завданням яких є побудова ефективної моделі розрахованої на багато користувачів веб-системи.

Для простоти було обрано лише дві пріоритети груп: до першого пріоритету було віднесено архітектори, до другого - всі інші групи.

Для побудови завдань для системи було проведено збір вимог та побажань кожної групи. Складання та ведення таких списків саме по собі є складним і кропітким. завданням. Для великих комерційних проектів, які також брали участь в апробації моделі, такі списки досягали кількох тисячі пунктів (з урахуванням приблизно рівної деталізації функцій).

На наступному кроці були встановлені залежності задач та пріоритети кожної конкретної задачі чи покращення.

Після цього етапу виходить неоцінений чорновий варіант графа завдань проекту. По ньому можна провести коригування пріоритетів завдань, з урахуванням того факту, що ніяка підпорядкована задача не може бути пріоритетнішою за батьківське завдання. Таке коригування виконується за кнопкою «Скоригувати». У програмі реалізований рекурсивний алгоритм обходу графа завдань проекту, список завдань є двозв'язним списком, у кожного елемента у випадку може бути N батьків і K залежних елементів. Також є можливість автоматичного проставлення пріоритету батьківського чи підлеглого завдання, рівного пріоритету пов'язаної з нею завданням, якщо завдання пріоритету не має.

Найскладніший і найбільш викликаючий розбіжності крок – оцінка вимог. Пропонована модель працює з уже готовими оціненими вимогами, і велика перевага моделі в тому, що вона не вимагає точних часових оцінок для великих завдань, розрахункові величини коригуються в ході виконання проекту. У моделі істотною є необхідність коректної взаємної оцінки різних вимог, тобто якщо завдання більше і складніше будь-якої з наявних завдань, то вона повинна бути оцінена в потрібну кількість разів більшим значенням. Незважаючи на те, що в системі вказано трудодні, насправді можна застосовувати будь-які відомі одиниці часу. Наприклад, за одиницю часу може бути прийнято час виконання будь-якого еталонного завдання. Згодом (протягом 1–2 ітерацій) у системі продуктивність праці буде вирівняно під цей стандарт.

Зробивши оцінку завдань, можна отримати список завдань із розставленими пріоритетами. Цей список може бути придатний до початку виконання, при необмежених ресурсах і вартості розробки, однак для отримання оцінок потрібно

скласти план завдань і виділити ітерації, що задовольняють обраним критеріям успішності.

Для побудови плану ітерацій, проведення розрахунків ймовірностей виконання завдань потрібно з цього списку завдань із пріоритетами скласти чергу поодиноких завдань. Будь-яке завдання може бути декомпоновано більш дрібні завдання, до сукупності кількох одиничних завдань, наприклад, реалізують окремі методи. У програмі одинично вважатимемо завдання, яке один фахівець може виконати за один робочий день.

Програма виконує цю операцію автоматично після завершення попереднього етапу при переході на форму «планування ітерації». Важливим моментом є збереження зв'язків поодиноких завдань між собою, програма враховує, що не можна в одну ітерацію запланувати цілу кількість завдань із початкового списку.

Форма планування ітерації представлена на рисунку 3.10. За замовчуванням на ітерацію вибираються верхні (найпріоритетніші) завдання відповідно до заданої довжини ітерації і прийнятої ймовірністю успішного завершення, що розраховується автоматично при відкритті форми. Передбачається можливість надалі коригувати автоматично згенерований план проекту, змінюючи завдання будь-якої з ітерацій.

Таблиця з ймовірностями виконання завдань відображається на формі складання плану ітерацій проекту.

Для кожної ітерації автоматично визначається кількість виконуваних завдань і за допомогою запропонованої моделі обчислюється можливість виконання їх до закінчення ітерації. Для керівника проекту є можливість у разі потреби вносити корективи до плану ітерації, змінюючи одне або відразу два обмеження: за часом або за кількістю завдань для кожної ітерації. Така гнучкість необхідна, оскільки на практиці існують важливі завдання проекту, наприклад реалізація пріоритетних бізнес сценаріїв, виконання яких не повинно бути зірвано.

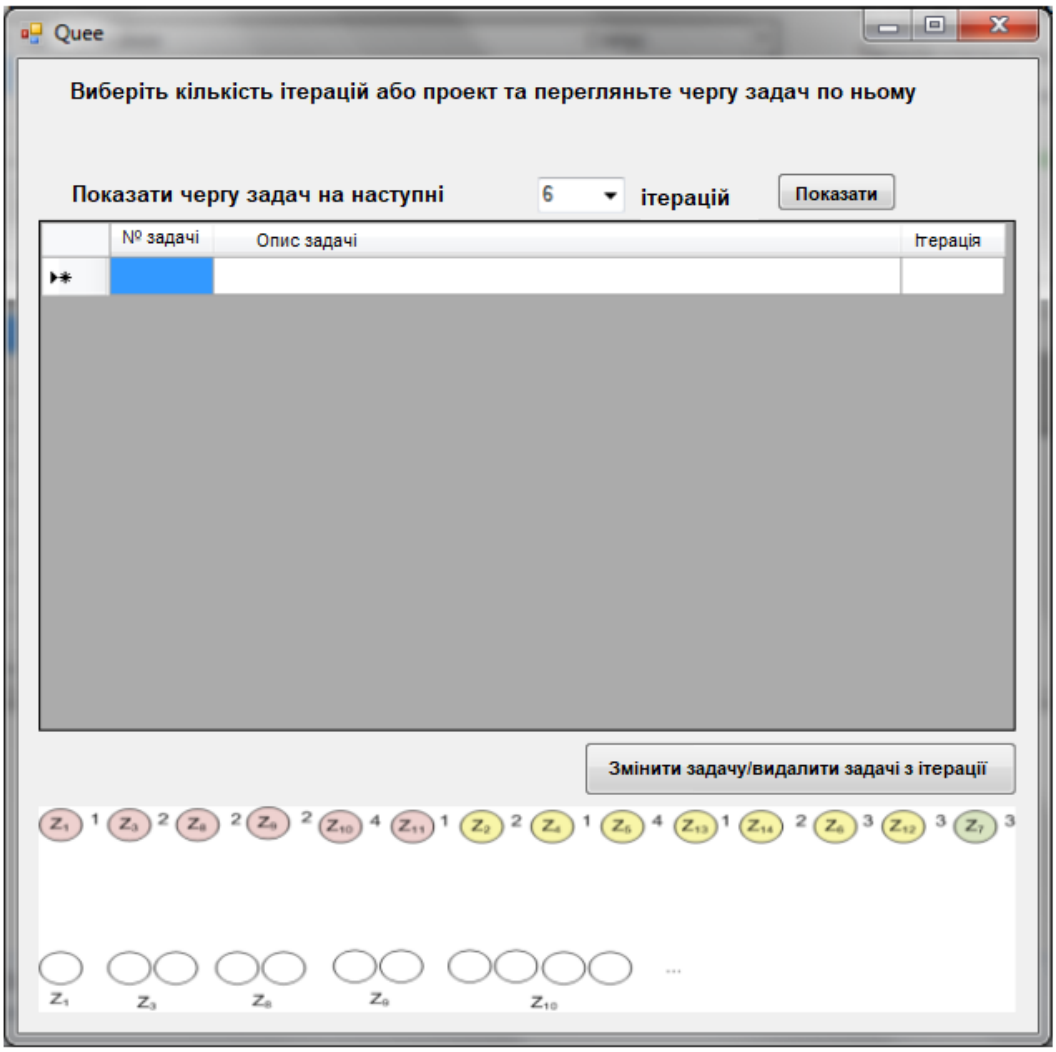

Рис. 3.10. Черга задач

Форма поточного статусу проекту разом із іншими формами може бути використання моделювання впливу різних нестандартних ситуацій на проект. Необхідно перевірити ситуації, пов'язані з впливом на план проекту розширення наявних вимог замовника. Залежно від своїх масштабів, таке розширення може призвести як до незначного зменшення ймовірності виконання проекту в строк, так і до суттєвого порушення поточного календарного плану.

Система для швидкого моделювання ступеня впливу на проект та ймовірності його закінчення у намічений термін дозволяє з достатньою точністю перевірити, наскільки ті чи інші зміни в обсягах робіт можливі без перегляду всього проекту. З його допомогою можна оперативно провести

попереднє планування оновленого проекту для подальшого рішення про доцільність внесення змін.

Друга потреба перегляду термінів і плану проекту, що часто зустрічається, з'являється у зв'язку зі збільшенням розміру завдань щодо початкового плану. Практично будь-яке завдання розробки навіть невеликої функції, тим більше якщо початкова оцінка була зроблена для більших вузлів системи, може містити додаткові складності, які неможливо врахувати на початкових етапах. Система реагує на подібні ситуації із двох сторін. По-перше, наявність невиконаних завдань за минулий період з невеликими застереженнями може бути розглянута як поява нових та впливає на зміну ймовірності планового завершення поточної ітерації та всього проекту. По-друге, невиконані завдання свідчать про зменшення продуктивності праці команди (інтенсивності виконання завдань), на основі якої здійснюється розрахунок ймовірностей у запропонованій моделі.

Поряд із неповною оцінкою завдань також необхідно розглядати ситуації, пов'язані з втратою розробника в команді та, навпаки, вплив посилення команди на проект. Зменшення кількості працюючих над проектом людей, окрім причин хвороби чи звільнення співробітників, може бути викликано переведенням (або частковим перекладом) на інші проекти. З іншого боку, збільшення кількості людей через підвищення важливості проекту за рахунок залучення людей з інших проектів також не рідкість. Це характерно насамперед великих компаній, де одночасно ведеться розробка кількох різних проектів, й у залежність від потреб, термінів, пріоритетів кожного проекту наявні ресурси може бути перерозподілені. Розроблена програма за допомогою форми «Команда розробки» дозволяє оперативно прорахувати вплив на календарний план проекту збільшення та зменшення кількості членів команди розробки.

Будь-якій системі ведення проектів потрібні максимально точні оцінки завдань, наприклад, при використанні методу PERT, залежно від набору оцінених завдань, може бути різна форма критичного шляху, а отже, всі

оцінки проекту можуть змінитися через одну або кілька неточних оцінок. У розробленій системі цей недолік не виявляється, неточні оцінки конкретних завдань впливають лише на виконання цих завдань, а систематичні неточні оцінки призводять до зміни інтенсивності розв'язання задач командою, яка автоматично перераховується залежно від поточних виконаних завдань, а прогноз на наступну ітерацію будується з використанням оновленої продуктивність.

Як було сказано вище, в ході виконання проекту програма автоматично щодня перераховує ймовірності рішення різного числа завдань до кінця ітерації Ймовірність виконання вчасно всіх запланованих завдань менше заданого порогового значення (0.8765<0.9). Тому проводиться розрахунок максимальної кількості завдань, які можуть бути виконані за відведений на ітерацію час із достатньою ймовірністю.

На практиці часто необхідно вирішити всі поставлені завдання, тому актуальним завданням є отримання прогнозу, за який час можуть бути виконані всі поставлені завдання із задовільною ймовірністю.

Таким чином, для даної команди розробки виконання всіх поставлених завдань на ітерацію можливе з ймовірністю 87.7%, з достатньою ймовірністю 91,8% можливе виконання трьох завдань за відведений час ітерації (3 дні) або виконання всіх поставлених завдань за 5 днів з ймовірністю 95.5% .

3.4. Експериментальні дослідження та оцінка ефективності запропонованого підходу

Було проведено порівняння оцінок реалізації запропонованої методики з використанням методів PERT та Монте-Карло для ітерацій кількох проектів, оцінено трудомісткість отримання таких оцінок та їх порядок. Оскільки оцінки є наближеними, являючи собою можливість успішності

виконання завдань, то можливе лише їх порівняння з реальними даними, отриманими після завершення даних проектів.

Порівняння методів проведено на прикладі планування та реалізації завдань ітерацій розробки у проекті створення системи автоматизованого аналізу тексту. Зі всього списку завдань проекту (близько 250) для складання прогнозу були обрані завдання для виконання в ході однієї ітерації.

В разі розробленої методики додаткова гнучкість виявляється в тому, що кількість завдань ітерації попередньо ставити не потрібно, оскільки воно визначається заданою ймовірністю виконання ітерації вчасно. Для методу PERT додатково було визначено критичний шлях (таблиця 3.1), а для методу Монте-Карло підготовлено додаток для генерації бета-розподілу.

Таблиця 3.1

Завдання ітерації проекту та критичний шлях

Задача	Мінімальна оцінка	Середня оцінка	Максимальна оцінка	Батьківська задача	Середньо-озважена
Доопрацювання мови сценаріїв					
Реалізація логіки операцій мови сценаріїв					
Реалізація скриптів для операцій	4	5	9		5,5
Реалізація операції визначення розміру словника	**2**	**4**	**5**	**3**	**3,83**
Передача параметрів операції	2	2	5	3	2,5
Виконання всіх операцій	3	5	7	5	5
Написання скрипта для повного поєднання	**2**	**3**	**3**		2,83
Створення системи зберігання даних мови сценаріїв					
Зберігання службових даних з операцій у БД	3	4	5		4
Створення системи зберігання сценаріїв	**4**	**4**	**8**	**9**	**4,67**
Перевірка існування тимчасової таблиці	**1**	**1**	**2**	**10**	**1,17**
Додавання можливості перезапису до існуючої структури	**1**	**1**	**2**	**4**	**1,17**
Перевірка на існування сценарію з таким ім'ям	**1**	**1**	**2**	**12**	**1,17**

Потім у відповідності з методом PERT було отримано час T_E – виконання завдань критичного шляху та стандартне відхилення для завдань

проекту. Тривалість завдань критичного шляху: 125 днів. Стандартне відхилення ітерації проекту: 2,17 днів.

Для визначення ймовірності виконання ітерації за 14 днів (T_S) необхідно обчислити величину Z – кількість стандартних відхилень від середньої величини.

$$Z = \frac{T_S - T_E}{\sum \sigma_{te}^2} \tag{3.1}$$

Потім за статистичними таблицями визначається відповідна ймовірність – 76%. Результат роботи методу Монте-Карло представлено на рисунку малюнку 3.11.

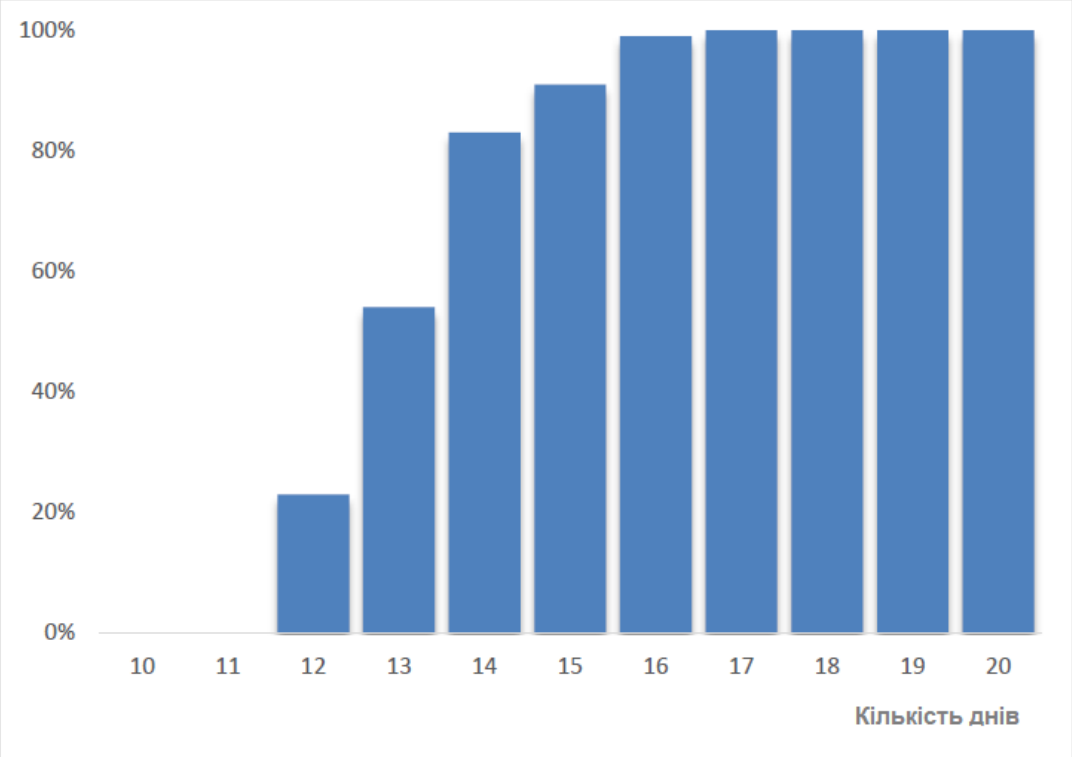

Рис. 3.11. Ймовірність успішного закінчення ітерації проекту

У таблиці 3.2 наведено аналогічні оцінки для 5 ітерацій проекту.

Таблиця 3.2

Результати порівняння методів прогнозування

№ ітерації	PERT	Монте-Карло	Запропонований метод	Кількість виконаних завдань (за планом – 14)	Успішність операції
1.	76%	86%	90%	14	Так
2.	37%	18%	35%	11	Ні
3.	70%	74%	70%	14	Так
4.	84%	79%	90%	16	Так
5.	77%	91%	90%	16	Так

Для 1-3 ітерацій кількість завдань було вручну, для 4 і далі кількість завдань ітерації було визначено автоматично, виходячи з умови 90% ймовірності успіху, що відповідає двом режимам роботи системи планування. Для 1–3 ітерації кількість завдань було встановлено вручну, для 4 і далі кількість завдань ітерації було визначено автоматично, виходячи з умови 90% ймовірності успіху, що відповідає двом режимам роботи системи планування.

Порівняння відносної помилки кожного методу було проведено, виходячи з припущення, що за планом ймовірність успіху ітерації має бути понад 90%. За допомогою кожного з методів було отримано прогноз кількості днів, необхідних для виконання ітерації, за умови заданої ймовірності успіху та фіксованої кількості завдань, оскільки методи PERT та Монте-Карло не дозволяють змінювати кількість завдань без перебудови моделі повністю. Потім середня відносна помилка прогнозу обчислюється за такою формулою:

$$\varepsilon = \frac{\sum_{i=1}^{N} \frac{|y_i - y_i'|}{y_i}}{n} \tag{3.2}$$

де: y_i – фактична кількість днів, яка була потрібна для виконання ітерації; y_i' - кількість днів в ітерації згідно з прогнозом; n – кількість ітерацій проекту

Результати порівняння одного з проектів наведено у таблиці 3.3.

Таблиця 3.3

Порівняння фактичного та прогнозованого часу ітерацій проекту

Фактичний час ітерації (днів)	Прогноз методом PERT	Прогноз методу Монте-Карло	Прогноз за пропонованим методом
14	15	15	15
18	19	20	17
14	16	15	15
14	15	15	15
14	15	15	13

Потім обчислюється відносна помилка кожного з методів кожної ітерації і визначається відносна похибка (таблиця 3.4).

Таблиця 3.4

Розрахунок відносної помилки прогнозу

Номер ітерації	Помилка прогнозу за методами:		
	PERT	Монте Карло	Пропонований метод
1	0,0715	0,0714	0,0714
2	0,0556	0,1111	0,0556
3	0,1429	0,0714	0,0714
4	0,0714	0,0714	0,0714
5	0,0714	0,0714	0,0714
Відносна похибка (ε)	0,0825	0,0794	0,0683

Порівняння отриманої відносної похибки дає такі значення для перерахованих методів (PERT, Монте-Карло, Запропонований метод): відносна похибка для запропонованого методу (0,0683) менше на 17%, ніж відносна похибка методу PERT (0,0825) та на 14% менше, ніж відносна похибка методу Монте-Карло (0,0794).

Розроблена система підтримки прийняття рішень була застосована для прогнозування термінів виконання завдань під час планування низки проектів розробки прикладного програмного забезпечення для масового використання.

Пропонований метод дає в середньому на 7% менше відхилення від фактичного часу порівняно з PERT і на 5% менше - щодо методу Монте-Карло. Крім того, запропонований метод дозволяє зробити розрахунки швидше, без серйозної залежності від виду критичного шляху та кількості розробників проекту.

Висновки до третього розділу

1. Розроблено архітектуру системи підтримки прийняття рішень при управлінні проектами створення ПЗ та оцінки термінів успішного завершення проектів.

2. Розроблено архітектуру, описано вхідні та вихідні дані та принципи функціонування модуля побудови та аналізу графа задач та створення на його основі черги завдань.

3. Розроблено систему аналізу та прийняття рішень при управлінні проектами розробки програмного забезпечення.

4. Наведено приклади роботи створеної системи, що використовує запропоновану модель під час планування та виконання прикладного проекту.

ВИСНОВКИ

В процесі виконання магістерської роботи отримано наступні наукові та практичні результати:

1. Проаналізовано різні визначення проекту та дано визначення проекту та проектної діяльності з точки зору розробки ПЗ. Виявлено основні фактори, що перешкоджають точному розрахунку часу закінчення проекту.

2. Проведено аналіз обмежень процесу розробки програмного забезпечення. У ході його було проаналізовано та класифіковано основні підходи до розробки ПЗ; виділено їх особливості, які необхідно враховувати під час створення методів планування.

3. Було проаналізовано відомі методи розрахунку термінів виконання проектів, визначено їх переваги та недоліки. Аналіз ходу реальних проектів показав, що більшість методів не надають користувачам необхідну точність при плануванні проектів розробки програмного забезпечення. Серед причин цього можна виділити насамперед їхню орієнтацію на проекти «в цілому», без зазначення конкретної галузі застосування даної методики. Тому розробка методів, що дають змогу більш точно планувати час закінчення проекту розробки програмного забезпечення, є актуальним завданням.

4. Команда розробки програмного забезпечення сприймається як система масового обслуговування. Як основні характеристики такої системи масового обслуговування вибрано ймовірність вирішення заданої кількості завдань у вибраний день, допустиму кількість завдань на ітерацію з урахуванням порогового значення ймовірності успішного її виконання, а також кількість днів для вирішення обраної кількості задач.

5. Запропоновано динамічну модель процесу виконання завдань командою розробки програмного забезпечення, що дозволяє в залежності від кількості членів команди автоматично змінювати модель системи масового обслуговування та чисельним методом Рунге-Кути перераховувати

залежність ймовірностей виконання заданої кількості завдань від поточного дня ітерації.

6. Розроблено архітектуру системи підтримки прийняття рішень при управлінні проектами створення ПЗ та оцінки термінів успішного завершення проектів.

7. Розроблено архітектуру, описано вхідні та вихідні дані та принципи функціонування модуля побудови та аналізу графа задач та створення на його основі черги завдань.

8. Розроблено систему аналізу та прийняття рішень при управлінні проектами розробки програмного забезпечення. Наведено приклади роботи створеної системи, що використовує запропоновану модель під час планування та виконання прикладного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sliger M. Agile project management and the PMBOK® guide. PMI® Global Congress 2008. Newtown Square, PA: Project Management Institute, 2008. URL: https://www.pmi.org/leaming/library/agile-project-management-pmbok-waterfall-7042.
2. Schwalbe K. Managing a Project Using an Agile Approach and the PMBOK® Guide. 2012. URL: https://kathyschwalbe.files.wordpress.com/2013/06/managing-a-project-using-an-agileapproach-and-the-pmbokc2ae-guide.pdf.
3. Project Management Institute. Agile Practice Guide. Newtown Square, PA: Project Management Institute. 2017.210 p.
4. Leffmgwell D. SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering. Addison-Wesley Professional, 2016. 569 p.
5. Gorakavi P. K. Build Your Project Using Feature Driven Development. 2009. URL: http://www.ipma-usa.org/articles/A4_AboutFDD.pdf
6. Agile Business Consortium. The DSDMAgile Project Framework. 2014. URL : https ://www. agile-business. org/page/TheDSDMAgileProj ectF ramework.
7. Poppendieck M., Poppendieck T. Lean Software Development: An Agile Toolkit. Addison-Wesley, 2003. 203 p.
8. 2018 IT Project Success Rates S urvey Results. URL: http://www.ambysoft.com/surveys/success2018.html
9. Tamanini L, Pinheiro P. R., Sampaio Machado T. C., Albuquerque A. B. Hybrid Approaches of Verbal Decision Analysis in the Selection of Project Management Approaches. Procedia Computer Science. 2015. No. 55. P. 1183 – 1192