Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Системотехніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

ГЮИК.502900.009
(позначення документа)

Розробка оптимізаційної моделі формування проектних команд для створення програмного забезпечення
(тема роботи)

Виконав: студент групи СПРм-18-2
спеціальності 122 – Комп'ютерні науки
(код і повна назва спеціальності)

освітньо-професійна програма Системне проектування
(повна назва)

Мироненко М.С.
(прізвище, ініціали)

Керівник проф. каф. СТ Мінухін С.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри системотехніки _____ Гребеннік І.В.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти *другий (магістерський)*
(освітньо-кваліфікаційний рівень)

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

«_____»_____2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЕКТ)

студентові Мироненко Максиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) *Розробка оптимізаційної моделі формування проектних команд для створення програмного забезпечення*
затверджена наказом по університету від «____» _____ . _____
2. Термін подання студентом роботи (проекту) _____
3. Вихідні дані до роботи (проекту) *Функція: розробка програмної системи для знаходження оптимального складу команд для створення програмного забезпечення. Організація вихідних даних: система повинна відображати результати роботи в інтерфейсі користувача. Форма діалогу: графічний інтерфейс. Перелік використаних програмних засобів: ОС Microsoft Windows 10, середовище програмування Visual Studio 2019. Технічне забезпечення: IBM-сумісний ПК з МП Pentium IV та вище.*
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) *Вступ. Огляд та аналіз сучасного стану проблеми формування проектних команд. Процес формування та класифікації проектних команд. Життєві стадії команди проекту. Постановка задачі. Формування математичної моделі. Математичні моделі формування команд. Задача про призначення. Теоретико-ігрові моделі. Експериментальні дослідницькі. Рефлексивні моделі. Математична модель організаційного управління складом виконавців комплексу робіт з невизначеним ступенем складності. Модель та метод оптимізації складу виконавців проектів з розроблення програмного забезпечення згідно з методологією MSF. Метод вирішення задачі. Програмна реалізація. Вибір середи та мови розробки. Розробка програмного засобу. Опис принципів роботи програми. Висновки.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, плакатів) *Взаємозв'язок завдань формування складу команд, розподілу функцій і розподілу обсягів робіт. Діаграма варіантів використання програмного засобу. Схема алгоритму роботи функції GenerateSystem1. Схема алгоритму роботи функції SolveEquation. Схема алгоритму роботи функції GenerateNewNumber. Початкове вікно програми. Результати роботи програми за сценарієм 1. Вікно результатів роботи програми за сценарієм 2. Вікно програми для виконання сценарію 3. Результати роботи програми за сценарієм 3.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання на дипломне проектування	30.03.2020	
2.	Аналіз завдання, пошук літератури та аналогів з теми дипломної роботи	31.03-05.04.2020	
3.	Опрацювання літератури та аналіз об'єкту дослідження	06.04 -11.04.2020	
4.	Формування постановки задачі	12.04.2020	
5.	Проектування програмного засобу	13.04-19.04.2020	
6.	Розробка та тестування програмного засобу	20.04-29.04.2020	
7.	Аналіз результатів, отриманих за допомогою програмного засобу	30.04.2020	
8.	Оформлення пояснювальної записки	01.05.2020-19.05.2020	
9.	Представлення на рецензування	25.05.2020	
10.	Представлення дипломної роботи	28.05.2020	

Студент _____ (підпис)

Керівник роботи _____ (підпис) *проф. Мінухін С.В.* (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 101 арк., 10 рис., 5 додатки, 34 джерел; графічний матеріал – 9 аркушів.

Об’єкт дослідження – процеси (фази) формування складу проектних команд для виконання IT-проектів згідно з методологією розроблення програмного забезпечення MSF.

Мета дипломної роботи - підвищення ефективності прийняття управлінських рішень шляхом оптимізації складу команд в рамках методології розроблення програмного забезпечення MSF.

Методи дослідження – методи організаційного управління; метод оптимізації; методологія MSF. Дослідження проводилося на IBM-сумісному персональному комп'ютері (тактова частота процесора 2.5 GHz, обсяг оперативної пам'яті – 6 ГБ) за допомогою Visual Studio 2019.

Результати дипломної роботи – модель, метод та програмний засіб, що дозволяють знайти ефективний склад команди проекту в умовах неоднорідності учасників проекту, різної ступені складності фаз проекту та обмеженнях на їхню вартість для методології MSF.

Область застосування – розроблені методи та програмний засіб можна використовувати при формуванні проектних команд для розроблення складних програмних систем та дослідженні їхньої ефективності.

КОМАНДА ПРОЕКТУ, C#, ФОРМУВАННЯ КОМАНД, ОРГАНІЗАЦІЙНА СИСТЕМА, МЕТОДОЛОГІЯ MSF, ЗАДАЧА ПРО ПРИЗНАЧЕННЯ

ABSTRACT

Explanatory note: 101 sheets, 10 figures, 5 annexes, 34 sources; graphic material - 9 sheets.

The object of research is the processes (phases) of forming the composition of project teams for the implementation of IT projects in accordance with the methodology of MSF software development.

The purpose of the thesis is to increase the efficiency of management decisions by optimizing the composition of teams within the methodology of MSF software development.

Research methods - methods of organizational management; optimization method; MSF methodology. The study was conducted on an IBM-compatible personal computer (processor clock speed 2.5 GHz, RAM - 6 GB) using Visual Studio 2019.

The results of the thesis - a model, method and software that allows you to find an effective composition of the project team in terms of heterogeneity of project participants, varying degrees of complexity of the project phases and limitations on their cost for the MSF methodology.

Scope - the developed methods and software can be used in the formation of project teams to develop complex software systems and study their effectiveness.

PROJECT TEAM, C #, TEAM FORMATION, ORGANIZATIONAL SYSTEM, MSF METHODOLOGY, ASSIGNMENT TASK

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – Організаційні системи

MSF – Microsoft Solution Framework

PM – Product Manager

TM – Team Model

WF – Windows Forms

MFC – Microsoft Foundation Classes

CLR – Common Language Runtime

API – Application Programming Interface

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій супроводжується багатьма організаційно-технічними інноваціями, зокрема пов’язаними з поліпшенням технологій програмування та організації бізнес-процесів розроблення програмних продуктів. Однією з таких технологій є технологія, що розроблена компанією Microsoft – Microsoft Solutions Framework. Головною відмінністю цієї технології є використання так званих ролевих кластерів виконавців проекту – особливих класифікаційних ознак, що визначають функції виконавців IT-проектів за різними технологічними етапами розробки програмних продуктів. Однією з ідей, покладених в основу дослідження, що пропонується, є визначення такого складу проектних команд, який враховує умови на обмеження: загальну кількість виконавців проекту, загальний час виконання проекту (проектів) та його фаз, загальну вартість проекту (проектів) та його фаз. Ясно, що за таких умов розв’язок завдання щодо оптимізації складу виконавців при наявності вказаних обмежень стає досить складним внаслідок його комбінаторного характеру. Тому вирішення проблеми що розглядається є актуальним та потребує створення моделі та методу для ефективного розв’язання поставленої задачі.

Метою даного дослідження є розроблення моделі, що враховує особливості технології розроблення програмних продуктів згідно з методологією MSF та методу, що дозволяють за наявності різних умов (обмежень) на виконання IT-проекту (проектів) забезпечити ефективність прийняття рішень щодо визначення складу проектних команд за прийнятний проміжок часу.

Для досягнення поставленої мети пропонується вирішити такі задачі:

а) розробити модель ролевих кластерів згідно з методологією MSF;

б) розробити метод оптимізації складу виконавців за цією методологією;

в) розробити програмний продукт для реалізації методу;

г) провести всебічні експериментальні дослідження в різних сценаріях – тестуванні та аналізу результатів роботи програми в режимі статистичного моделювання;

д) сценарії з вхідними величинами параметрів, які задаються користувачем.

В результаті виконання атестаційної роботи були опубліковані тези у VI Міжнародній науково-практичній конференції "SCIENCE, SOCIETY, EDUCATION: TOPICAL ISSUES AND DEVELOPMENT PROSPECTS", що проходила 10-12 травня 2020 р., м. Харків, Україна.

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ ФОРМУВАННЯ ПРОЕКТНИХ КОМАНД

Сьогодні в світі бізнесу найбільшої популярності набула командна форма діяльності [4], бо вона не тільки сприяє скорішому процесу виконання поставлених завдань, а й збільшує експертну оцінку такої групи, допомагає членам команди не втрачати мотивації до виконання свого процесу, знаходити вихід з будь-якої ситуації, адже завжди можна звернутися за порадою до колеги. Така команда є автономною одиницею – гравцем цілого підприємства, що досягає поставленої мети за мінімального контролюючого фактору, але повинна сама мати високий рівень самоконтролю.

Під командою розуміється колектив (об'єднання людей, що здійснюють спільну діяльність і володіють спільними інтересами), здатний досягати мети автономно й узгоджено, при мінімальних керуючих впливах [18]. Команди отримали широку поширеність в наш час. З одного боку вже в середині 90-х років XX століття більш ніж в 50% американських фірм існували «виробничі» команди, з іншого боку, команди тривалий час існують у багатьох областях діяльності, вже звичними стали терміни «команда проекту», «управлінська команда», «творча команда».

Суттєвими в наведеному визначенні команди є два аспекти. Перший - досягнення мети, тобто кінцевий результат спільної діяльності є для команди системоутворюючим фактором. Другий аспект - автономність і узгодженість діяльності - означає, що кожен з членів команди демонструє поведінку, яка вимагається в даних умовах (що дозволяє досягти поставленої мети), тобто ту поведінку, якого від нього очікують інші члени команди.

За формою команда проекту відображає існуючу організаційну структуру управління проектом, розділення функцій, обов'язків і відповідальності за рішення, що приймаються в процесі його реалізації. На верхньому рівні структури

знаходиться менеджер проекту, а на нижніх – виконавці, відділи і фахівці, що відповідають за окремі функціональні сфери.

За змістом команда проекту є групою фахівців високої кваліфікації, які володіють знаннями і навичками, необхідними для ефективного досягнення цілей проекту.

Основним інтегруючим чинником створення і діяльності команди виступає стратегічна мета реалізації проекту. У процесі досягнення цілей проекту команда набуває своїх меж, використовує організаційні можливості учасників і ресурси проекту. Команда проекту виступає як соціальний організм, що має свій початок, здійснює процес життєдіяльності (управління проектом) і завершує своє існування розформуванням або трансформацією в іншу управлінську команду.

Причини зростання популярності команд: зростання конкуренції, технологічні досягнення, необхідність вирішення складних проблем і відповідності швидким темпам змін, плинність кадрів. Але команди володіють і недоліками: висока концентрація фахівців на вузькому фронті робіт, підвищений фонд стимулювання і інтенсивний ритм роботи служб забезпечення, необхідність навчання і тренінгу членів команди, обмеженість розміру, можливість саморозпаду.

З одного боку, команда проекту впливає на створення певного організаційного середовища проекту, формуючи цінності, принципи і норми поведінки персоналу. З іншого боку, діє в ній, підкоряючись єдиній меті та філософії управління проектом. Тому проблеми формування і діяльність команди проекту доцільно розглядати в логічній послідовності: мета проекту – система управління – команда проекту.

З огляду на те, що результат діяльності команди залежить від дій кожного її учасника, то для того, щоб вибрати свою поведінку суб'єкт повинен «передбачити», які дії виберуть інші члени команди, а для цього необхідно мати уявлення про ту інформацію, якою володіють інші члени команди. Це можливо на основі підходу, що враховує «рефлексивні» аспекти прийняття рішень членами команди, в яких автономність і злагодженість спільної діяльності членів команди

(по досягненню загальних цілей) забезпечується тим, що їх дії узгоджені з ієрархією взаємних уявлень один про одного.

Таким чином, актуальність теми визначається необхідністю розробки моделей формування складу проектних команд для створення програмного забезпечення з урахуванням об’ємів робіт та їхньої вартості.

1.1 Процес формування та класифікації проектних команд

В процесі формування команд основною ознакою є класифікаційна, бо саме від неї залежить які характеристики будуть опорними для майбутньої команди, як буде проходити процес роботи команди та що буде запорукою її успіху у в професійній діяльності.

Існує безліч більш-менш детальних класифікацій команд з різних підстав. Так, наприклад, виділяють[4] постійні і тимчасові, формальні і неформальні команди. Існує теорія [3], що потрібно виділити два основних типи команд: функціональний та творчій. В функціональній команді усі її члени виконують певну функцію, в них однакові права, можливості та обов’язки, та вони підпорядковані одній керуючій ланці. Творча ж команда складається з людей різних спрямувань та навичок для досягнення спільної мети.

Залежно від специфіки, розміру та типу проекту в його реалізації можуть брати участь від однієї до кількох десятків (іноді - сотень) організацій та окремих фахівців. У кожної з них свої функції, ступінь участі в проекті й міра відповідальності за його реалізацію. Фахівців та організації, залежно від виконуваних ними функцій, прийнято об'єднувати в абсолютно конкретні групи (категорії) учасників проекту.

Нарешті, існує команда проекту, очолювана керівником проекту – менеджером проекту (проект-менеджер), а також, залежно від специфіки проекту, в проекті можуть бути й інші учасники. Команди класифікують і за їх розміром [8], зазвичай, це маленькі команди (до 4 осіб), середні команди (від 5 до 9 осіб) та великі команди (більше 10 осіб).

Слід зазначити, що учасники проекту – категорія більш широка, ніж команда проекту. Команда проекту – одне з головних понять управління проектами. Це група співробітників, які безпосередньо працюють над здійсненням проекту і підлеглих керівникові останнього, основний елемент його структури, оскільки саме команда проекту забезпечує реалізацію його задуму. Ця група створюється на період реалізації проекту і після його завершення розпускається. Як правило, лідери (менеджери) функціонально і (або) предметно орієнтованих груп фахівців і складають команду управління проектом. Лідери груп – це керівники, координатори зусиль всіх членів групи. Члени групи – безпосередні виконавці, які мають можливість зосереджуватися на конкретній роботі. При необхідності деякі ролі членів команди можуть поєднуватися.

Організаційна структура команди проекту розкриває взаємини учасників проекту всередині команди.

Існує два основних принципи формування команди для управління проектом:

Перший принцип – провідні учасники проекту: замовник і підрядник (Крім них можуть бути і інші учасники) створюють власні групи, які очолюють керівники проекту, відповідно, від замовника і підрядника. Ці керівники підпорядковуються єдиному керівнику проекту. Залежно від організаційної форми реалізації проекту, керівник від замовника або від підрядника може бути керівником всього проекту. Керівник проекту в усіх випадках має власний апарат співробітників, які здійснюють координацію діяльності всіх учасників проекту.

Другий принцип – для управління проектом створюється єдина команда на чолі з керівником проекту. В команду входять повноважні представники всіх учасників проекту для здійснення функцій відповідно до прийнятого розподілу зон відповідальності.

Система управління командою проекту включає:

- організаційне планування;
- кадрове забезпечення проекту;
- створення команди проекту;

– здійснення функції контролю і мотивації трудових ресурсів проекту для ефективного ходу робіт і завершення проекту.

Система націлена на керівництво і координацію діяльності команди проекту, використовує стилі керівництва, методи мотивації, адміністративні методи, підвищення кваліфікації кадрів на всіх фазах життєвого циклу проекту.

Суть команди – в усвідомленні значущості мети, загальної для всіх її членів, якої переймаються всі члени команди, вірять в її досяжність. В досяжність її місії, яка для проекту полягає в його ефективній реалізації.

Для команди проекту необхідна наявність у її членів комбінації взаємодоповнюючих навичок (компетенцій), що складають три категорії:

– технічні і / або функціональні, тобто професійні навички;
– навички щодо вирішення проблем та прийняттю рішень;
– навички міжособистісного спілкування (прийняття ризику, корисна критика, активне слухання і т.д.).

Команда володіє такими суттєвими ознаками, як:

– внутрішня організація, що складається з органів управління, контролю і санкцій;
– групові цінності, на основі яких формується почуття спільності в команді і створюється громадська думка;
– власний принцип відокремлення, який вирізняє її від інших команд;
– груповий тиск, тобто вплив на поведінку членів команди загальними цілями і завданнями діяльності;
– прагнення до стійкості завдяки механізму відносин, що виникають між людьми в ході вирішення спільних завдань;
– закріплення певних традицій.

Команда – це самостійний суб'єкт діяльності, який може бути розглянутий з точки зору властивостей, процесів, параметрів, характерних для соціальної групи.

До основних факторів, що визначають принципи формування команди відносяться [9]:

Специфіка проекту. Команда проекту організовується для його реалізації, тому така характеристика, як специфіка проекту - одна з головних в утворенні команди.

Специфіка проекту визначає:

- формальну структуру команди, яка затверджується керівництвом;
- рольовий склад;
- перелік знань, умінь і навичок, якими повинні володіти члени команди;
- терміни, етапи, види робіт по проекту.

Організаційно-культурне середовище. Це середовище команди проекту ділиться на зовнішню і внутрішню.

Зовнішнє середовище включає в себе оточення проекту у всіх аспектах.

Внутрішнє середовище, або організаційна культура самої команди, включає такі характеристики:

- прийняті і розділені усіма учасниками норми команди;
- способи розподілу влади;
- згуртованість і зв'язаність членів команди;
- характерні способи організації та протікання командної взаємодії (командних процесів - координації, комунікації, діяльності з вирішення конфліктів та прийняття рішень, налагодженню зовнішніх зв'язків);
- організація рольового розподілу.

Управління командою проекту пов'язано з необхідністю створення раціональної структури, забезпечення високого ступеня професіоналізму співробітників, складністю досягнення оптимального співвідношення зовнішнього контролю і незалежності команди. Менеджер повинен бути гнучким, впевненим в собі і в своїх співробітниках. Вплив в команді ґрунтується не на статусі або положенні, а на професіоналізм та компетентність.

При реалізації проекту специфіка управління командою полягає в тому, що вона, як правило, не є традиційною самостійною організацією. Відповідна організаційна форма повинна бути індивідуально підібрана під конкретний проект. В цілому організаційні структури управління проектами належать до

гнучких, органічних структур управління. Для них характерною є відсутність детального розподілу обов’язків по видах робіт, невелика кількість рівнів управління, децентралізація прийняття рішень, індивідуальна відповідальність кожного члена команди за результати діяльності.

При формуванні команди можуть виникнути два варіанти:

Варіант 1. Проект реалізується в рамках підприємства (організації). При цьому є три можливості:

а) Робота над проектом як додаткове завдання в рамках повсякденної діяльності. Це означає включення управління проектом в звичайний ритм роботи. Керівництво організації визначає власного керівника проекту, який в рамках організаційної схеми одночасно виконує і свої звичайні обов'язки, і при цьому додатково керує проектною командою і має професійний доступ до значних співробітників (незалежно від кордонів відділів). Він також планує ресурси і координує всю діяльність по проекту.

б) Класична організація проекту (окрема оргструктура в рамках оргструктури підприємства). У такій моделі, яка вибирається при комплексних і об'ємних завданнях, особливо сильно підкреслено значення роботи над проектом в організаційній структурі підприємства. Робота в команді проекту має однозначний пріоритет перед ієрархічними і дисциплінарними відносинами підпорядкування класичної структури підрозділів підприємства.

в) Змішані форми – призначається звільнений від інших видів діяльності досвідчений менеджер проекту і, в залежності від проекту, залучаються досвідчені спеціалізовані співробітники, які, проте, одночасно займаються своєю звичайною діяльністю. При цьому вся відповідальність лежить на менеджері проекту, який повністю може сконцентруватися на реалізації проекту і має більше свободи при призначенні співробітників для цього проекту.

На практиці і, перш за все, в середніх компаніях частіше переважають змішані форми.

Варіант 2. Проект реалізується поза рамками одного підприємства (організації), тобто команда формується переважно з представників різних організацій.

Як правило, проекти реалізуються далеко не завжди в рамках окремого підприємства. У таких випадках під конкретний проект створюються специфічні структурні утворення, як правило, які є адаптивними організаційними структурами.

Розрізняють чотири основні підходи до формування команди:

- цілеспрямовуючий (заснований на цілях);
- міжособистісний;
- рольовий;
- проблемно-орієнтований.

Цілеспрямовуючий підхід дозволяє членам команди краще орієнтуватися в процесах вибору і реалізації проекту.

Міжособистісний підхід сфокусований на поліпшенні міжособистісних відносин в команді і заснований на тому, що міжособистісна компетентність збільшує ефективність діяльності команди. Його мета – збільшення групової довіри, заохочення спільної підтримки, а також збільшення комунікацій в середині команди.

Рольовий підхід – проведення дискусії і переговорів серед членів команди щодо ролей. Передбачається, що ролі членів команди частково перекриваються. Командну поведінку може бути змінено в результаті зміни їх виконання, а також індивідуального сприйняття ролей.

Проблемно – орієнтований підхід (через рішення проблем) припускає організацію заздалегідь спланованих серій зустрічей з групою фахівців в рамках команди, що мають спільні організаційні відносини і мети. Підхід включає в себе послідовний розвиток процедур рішення командних проблем і потім досягнення головного командного завдання.

Головна мета формування команди – самостійне управління і подолання своїх проблем. Цей процес може не реалізовуватися відразу ж, а протягом

тривалого часу. Нерідко команді перешкоджає ефективно працювати саме керівництво або менеджер.

В ході спільної роботи визначаються найважливіші (актуальні) командні проблеми, і група може досягти нового рівноважного стану, яке встановлює більш високий рівень особистої участі і загальнокомандного клімату.

Роль являє собою сукупність очікувань щодо члена команди або людини на роботі. Групи нерідко стикаються з проблемами, викликаними труднощами, які пов'язані з тим, як члени команди визначають ролі і справляються з ними.

Рольова невизначеність має місце в тих випадках, коли людині неясна його роль. Для того, щоб добре виконувати будь-яку роботу, людина повинна знати, що від неї очікують. Навіть в зрілих групах і командах нездатність членів розділити очікування або прислухатися один до одного може іноді привести до нерозуміння.

Рольове перевантаження виникає в тих випадках, коли від людини очікують виконання занадто великої роботи, з якої та не може впоратися з якихось об'єктивних причин (недоліки в ресурсному забезпеченні, прогалини в професійних знаннях і уміннях, проблеми зі здоров'ям і т.д.).

Рольовий конфлікт виникає, коли людина виявляється не в змозі задовольнити очікування оточення, не здатна реагувати на рольові очікування, які суперечать один одному.

Рольові переговори - це один із способів управління рольової динамікою в будь-якій групі або сфері діяльності. Це процес, в ході якого індивіди займаються з'ясуванням рольових очікувань кожного щодо інших людей.

Можна також виділяти однорідні і неоднорідні (за ролями і функціями членів, їх професіями) команди. Таким чином, отримуємо систему класифікацій команд з кількох підстав (в залежності від розв'язуваної задачі число підстав класифікації може збільшуватися):

– однорідні (основне завдання – розподіл обсягів робіт) і неоднорідні (основне завдання – розподіл ролей і видів діяльності між агентами, а потім вже – розподіл обсягів робіт);

- постійні і тимчасові;
- формальні і неформальні;
- функціональні та творчі.

1.2 Життєві стадії команди проекту

Процес формування команди проекту (командоутворення) зазвичай розглядають як утворення єдиного, цілісного колективу управлінців, здатного ефективно досягати мети проекту [11].

Значення командної роботи по реалізації проекту укладається в можливості синергетичного ефекту від об'єднання групових зусиль, знань і вироблення групових управлінських рішень, тобто в досягненні "стану, при якому ціле більше ніж сума його складових частин".

Аналогічно життєвому циклу проекту команда проекту має свій життєвий цикл, в якому можна виділити п'ять основних стадій:

- формування;
- спрацювання;
- функціонування;
- реорганізація;
- розформування.

Розглянемо кожну стадію детальніше:

Формування. Особливості роботи в проекті полягають в тому, що фахівці команди не знають один одного, не є єдиним колективом з встановленими механізмами взаємодії, груповими установками. На цій стадії відбувається знайомство членів команди один з одним і з проектом загалом, формуються загальні цілі і цінності, визначаються норми і правила взаємодії, ставляться задачі команди і визначаються шляхи і принципи їх досягнення.

Спрацювання. Це період початку спільної роботи, розвитку згуртованості групи, що вирішує колективну задачу. Він характеризується підвищеним рівнем конфліктності, викликаним відмінностями в характерах фахівців, підходах, стилях

і методах розв’язання проблем. Всередині команди йде процес виявлення лідерів, формування неформальних груп, визначаються ролі окремих працівників і їх місце в команді, встановлюється психологічний клімат в колективі, його внутрішня культура тощо.

Робоча стадія. На основі сформованого командного почуття йде нормальний продуктивний процес роботи. Деталі взаємодії уточнюються по ходу виконання задач, спілкування в різних ділових ситуаціях. Задачею менеджера проекту на цій стадії є раціональний розподіл функцій між фахівцями і відділами, забезпечення відповідності особистих можливостей і здібностей, структурі і змісту робіт, що виконуються, з'єднання в робочих групах і функціональних підрозділах працівників з різними доповнюючими індивідуальними здібностями, підтримка в команді атмосфери довіри і взаємовиручки, єдності в розумінні цілей і задач проекту і способів їх досягнення; визначення і розв’язання конфліктних ситуацій; створення дійової системи мотивації, контроль за досягненням проміжних результатів проекту і координування діяльності всіх функціональних відділів.

Реорганізація. Стадія виникає при змінах в кількісному і якісному складах команди у випадках, викликаних: змінами в проекті (задачах, планах, результатах проекту), змінами структури управління проектом, завершенням окремих стадій проекту, зміною об'ємів і видів робіт, учасників проекту, заміною працівників через професійну невідповідність, додатковим залученням нових фахівців, запрошенням тимчасових експертів.

Розформування. При завершенні окремих стадій і всього проекту розформовуються окремі підрозділи і вся команда проекту. При цьому в залежності від прийнятої оргструктури виникають два варіанти подальших дій фахівців команди.

При матричній структурі управління працівники по закінченню проекту повертаються в свої функціональні підрозділи організації.

При проектній структурі управління менеджер проекту стикається з проблемою подальшого працевлаштування працівників, які не мають можливості повернутися на колишнє місце роботи. У цьому випадку, якщо очікується

замовлення на новий проект, при успіху діяльності команди менеджер має можливість запросити частину фахівців в команду нового проекту.

1.3 Постановка задачі

Загальна постановка – в рамках методології розроблення ПЗ треба визначити склад команд по виконавцям {X} для реалізації проекту (проектів) в цілому в рамках обмежень на вартість проекту (проектів) і його фаз, або загальної тривалості проекту (проектів) і його фаз.

Приватна постановка - визначити склад команд по виконавцям {X} для реалізації окремих фаз проекту в рамках певних обмежень на вартість і тривалість окремих фаз.

Вартість фази визначається згідно з рівнянням:

Вартість фази = Трудомісткість фази * Ціна одиниці часу фази.

Введемо обмеження на тривалість окремої фази проекту – D_i. Тривалість виконання і-ої фази повинна бути однаковою для будь-якої з обраних типів команд, тобто повинно виконуватися рівняння:

$$D_i=T_i/(k_j * A_{ij}) = const_i ,$$

де k_j – кількість виконавців j-ой команди; A_{ij} – кількість j-их команд для виконання і-ої фази; $k_j * A_{ij}$ - нормувальний коефіцієнт для окремої команди (загальна кількість виконавців по всіх командах даного типу).

2 ФОРМУВАННЯ МАТЕМАТИЧНОЇ МОДЕЛІ

Як об'єкт дослідження, будь-який проект, який представляє собою сукупність комплексів робіт різного ступеня складності, передбачає наявність у складі своєї команди виконавців різного рівня кваліфікації. При цьому питання формування раціонального складу виконавців різної кваліфікації є одним з основних факторів, що визначають ефективність реалізації проекту, яка характеризується відношенням загального обсягу виконуваних робіт до сумарних витрат фінансових коштів.

Найбільш відомими з існуючих підходів до формування складу команди проекту є методи, засновані на використанні інформаційної бази досвіду минулих розробок і теорії прецедентів. Однак висока ступінь унікальності проектів, а також велика нестабільність умов їх реалізації суттєво знижують ефективність прецедентного підходу до формування якісного складу команди проекту внаслідок впливу на заплановані процеси важко прогнозованих факторів. У цих умовах пропонується використовувати як інструмент формування ефективного складу команди проекту організаційні принципи централізованого, узгодженого і відкритого управління персоналом.

2.1 Математичні моделі формування команд

Виділяють [1] чотири математичні моделі формування та функціонування команд:

Задача про призначення – використовує, переважно, апарат оптимізації для вирішення задач формування складу команд, розподілу ролей та об'ємів робіт. На сьогоднішній день в математичній економіці та дослідженні операцій накопичено значний досвід постановки і вирішення різноманітних завдань розподілу обсягів робіт, який доцільно використовувати і при аналізі процесів ефективного формування та функціонування команд. Наприклад, транспортна задача і задача

про призначення є хрестоматійними завданнями дослідження операцій і мають безліч узагальнень, які доцільно використовувати при вирішенні завдань розподілу функцій між членами команди. Завдання пошуку оптимального складу команди може бути зведена до відомої задачі пошуку рішення кооперативної гри.

Теоретико-ігрові моделі – використовують апарат теорії ігор для опису та дослідження процесів формування і функціонування команд. Найбільше просування в цій області досягнуто в двох напрямках. Перше - вивчення моделей колективного стимулювання, в яких винагорода членів команди залежить від досягнутого ними результату спільної діяльності (що відображає автономність діяльності команди). При цьому члени команди самі узгоджено приймають рішення про оптимальний розподілі робіт між ними. Другий напрямок пов'язаний з моделями норм діяльності (правил, що пропонують членам команди ту чи іншу поведінку в залежності від ситуації) і репутації, тобто норм діяльності членів команди з точки зору один одного. Цей клас моделей адекватно описує автономність і стійкість функціонування команд.

Експериментальні дослідницькі – включають в себе імітаційні експерименти і ділові ігри.

Рефлексивні моделі – використовують апарат теорії рефлексивних ігор для опису взаємодії членів команди, маючих незбіжні взаємні уявлення о існуючих параметрах один одного.

2.1.1 Задача про призначення

Термін «задача про призначення» є умовним і охоплює широкий клас оптимізаційних задач, що включає завдання формування складу команд, завдання розподілу функцій (Ролей) в неоднорідних командах і завдання розподілу обсягів робіт.

Перераховані три типи завдань взаємопов'язані і вирішуються «Циклічно» - адже для того, щоб формувати склад команди, потрібно знати, які функції буде виконувати той чи інший агент, включений в команду, а для оптимального

розподілу функцій потрібно знати, який обсяг робіт доцільно виконувати даному агенту в рамках тієї чи іншої функції (рис. 2.1). Тому розглянемо послідовно завдання розподілу обсягів робіт, завдання розподілу функцій і завдання формування складу команди.

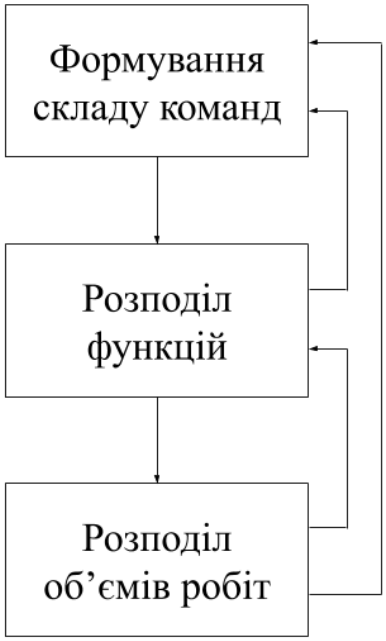

Рисунок 2.1 – Взаємозв'язок завдань формування складу команд, розподілу функцій і розподілу обсягів робіт

Задача розподілення об’ємів робіт. Нехай фіксований склад команди – безліч однорідних (виконуючих однотипні функції) агентів N = {1, 2, ..., n}, відомий сумарний обсяг робіт R ≥ 0, які потрібно виконати, і задані типи агентів $\{r_i\}_{i\in N}$ (характеристики, що відображають ефективність їх діяльності). Потрібно розподілити обсяги робіт між агентами.

Така постановка задачі є надто загальною і вимагає деталізації. Можуть бути різні варіанти. По-перше, слід розділити дискретні і безперервні задачі. У дискретної задачі обсяг робіт $d_i \geq 0$, який може виконувати i-ий агент, фіксований. Якщо інтерпретувати тип агента як собівартість виконання ним одиничного обсягу робіт, то отримаємо дискретну задачу розподілу обсягу робіт R між агентами з метою мінімізації сумарних витрат (змінна x_i приймає значення

0, якщо *i*-ий агент не працює, та значення 1, якщо він працює). Таку модель можна представити у вигляді формули (2.1):

$$\sum_{i \in N} d_i r_i x_i \rightarrow \min_{x_i \in \{0;1\}} . \quad (2.1)$$

де $d_i \geq 0$ – дискретне значення об'єму робіт, r_i – типи агентів команди, N – множина однорідних за функціями, типами агентів; при цьому x_i = 1, якщо даний агент бере участь у виконанні певного завдання, 0 – якщо він не виконує це завдання.

Загальним «недоліком» дискретних задач [1] є те, що лише мала їх частина має ефективні (поліноміальної складності) методи вирішення. Для NP-складних задач при малій їх розмірності можна використовувати метод повного перебору, а при збільшенні розмірності - різні евристичні чи інші методи вирішення.

Більш складними є завдання розподілу ресурсів команди між роботами, пов'язаними технологічно, наприклад – в рамках проекту. При заданому мережевому графіку, що відображає взаємозв'язок робіт, тривалість кожної роботи залежить від використовуваного для її виконання ресурсу. Отже, за рахунок розподілу обсягів робіт і ресурсів між агентами, можна впливати на довжину критичного шляху, що визначає тривалість проекту.

Для пошуку оптимального навантаження кожного з членів команди у таких моделях використовують принцип розподілення навантаження транспортних задач: при замкнутій задачі, де сума всіх a_i – кількість часу, який можна виділити на виконання задачі дорівнює сумі всіх b_j – час на фактичне виконання цих завдань. Також відомі витрати s_{ij} на виконання *i*-им агентом *j*-ой роботи протягом одиниці часу. Оскільки задача є замкнутою, тобто:

$$\sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j . \quad (2.2)$$

де сумарний часовий ресурс агентів дорівнює сумарній тривалості робіт (вводячи фіктивного агента або фіктивну роботу будь-яку незамкнуту завдання можна звести до замкнутої). Потрібно визначити розподіл агентів по роботах (функціям), що мінімізуючи сумарні витрати.

Формально задачу розподілення навантаження можна представити у вигляді формули (2.3):

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} s_{ij} \to \min_{\{x_{ij} \geq 0\}}, \tag{2.3}$$

$$\sum_{j=1}^{m} x_{ij} = a_i, i = \overline{1,n}, \tag{2.4}$$

$$\sum_{i=1}^{n} x_{ij} = b_j, j = \overline{1,m}. \tag{2.5}$$

за умов, що кожен агент завантажений повністю (2.4), а кожне завдання виконане (2.5).

Окремим випадком транспортної задачі є задача про призначення (у вузькому сенсі), яка полягає в наступному: є n агентів, які можуть виконувати різні роботи (реалізовувати різні функції, займати різні посади), число робіт дорівнює числу працівників (ввівши фіктивні посади та/або фіктивних агентів, завжди можна незамкнуту задачу привести до розглянутої замкнутої формі). Відомі витрати $\hat{S}_{ij}$ на призначення i-го агента на j-у посаду (наприклад, мінімальна зарплата, за яку він погодиться працювати на цій посади). Потрібно знайти призначення працівників на посади (Кожного працівника на одну і тільки одну посаду), що мінімізує сумарні витрати (якщо $\hat{S}_{ij}$ інтерпретується як ефективність від роботи i-го працівника на j-ій посаді, то оптимальне призначення має максимізувати сумарну ефективність). Формально задачу про призначення можна записати у вигляді:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}\,\hat{s}_{ij} \rightarrow \min_{\{x_{ij}\in\{0;1\}\}}, \quad (2.6)$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = \overline{1,n}, \quad (2.7)$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = \overline{1,n}. \quad (2.8)$$

Транспортна задача і задача про призначення є хрестоматійними завданнями дослідження операцій і мають безліч узагальнень [1] (облік обмежень на сумісність робіт, виконуваних одним агентом, обмежень на послідовність виконання робіт, багатокритеріальності і т.д., які доцільно використовувати при вирішенні завдань розподілу функцій між членами команди. Крім того, для таких задач може виявитися адекватним і апарат теорії масового обслуговування, в разі, якщо набір функцій, реалізованих командою, змінюється в часі по відомим (статистично описуваним) законам.

Вміючи знаходити оптимальний розподіл функцій і обсягів робіт для фіксованого складу членів команди, можна ставити і вирішувати завдання формування оптимального складу команди.

Задача формування складу команди. Розглянемо коротко відомі на сьогоднішній день підходи теорії управління та економічної науки до завдань формування складу організаційних систем.

В рамках економіки праці основний результат, що визначає оптимальну кількість працівників, відображає рівність виробленого ними граничного продукту (граничної продуктивності) і граничних витрат на їхнє залучення й утримання. Об’єм додаткової продукції (доходу), який отримує фірма, наймаючи одного додаткового (понад вже працюючих) працівника (одиницю праці), називається граничним продуктом праці. Граничні витрати є витрати на прийом на роботу додаткового працівника. Умова максимізації прибутку (різниці між доходом і витратами) вимагає, щоб прибуток був максимальний. Для цього слід змінювати число зайнятих (збільшувати, якщо граничний дохід перевищує граничні витрати, і зменшувати в іншому випадку) до тих пір, поки граничний

дохід не буде дорівнює граничним витратам. Такий підхід цілком можна застосовувати для визначення оптимального розміру однорідної команди.

В економіці організацій прийнятий наступний загальний підхід до визначення оптимального розміру організації. З одного боку, існує ринок – як система обміну прав власності. З іншого боку, економічні агенти об'єднуються в організації, які взаємодіють на ринку. Поясненням існування економічних організацій служить необхідність компромісу між транзакційними і організаційними витратами, які визначаються «витратами на координацію» всередині організації, що ростуть зі збільшенням її розмірів. Для команд характерна наявність тісних інформаційних та інших зв'язків між усіма її членами. Тому з ростом числа членів команди, організаційні витрати (залежать від числа зв'язків) ростуть дуже швидко. Напевно, цим пояснюється те, що практично в жодній сфері діяльності не зустрічаються команди, що складаються з декількох сотень або навіть десятків людей.

Транзакційні витрати перешкоджають ринку замістити собою організацію, а організаційні витрати перешкоджають організації замістити собою ринок. Основна ідея (якісна), яка використовується в економіці організацій при обговоренні задач формування складу, полягає в тому, що, так як і транзакційні, і організаційні витрати залежать від розміру організації та її структури, то теоретично повинні існувати оптимальні параметри організації, при яких досягається урівноваження згаданих тенденцій заміщення.

Перші систематичні постановки завдань формування складу організаційних систем (ОС) з'явилися нещодавно [6]. Можна виділити три загальних підходи до вирішення завдань формування складу ОС.

Перший підхід полягає в «лобовому» розгляді всіх можливих комбінацій потенційних учасників ОС. Його перевага - знаходження оптимального рішення, недолік - висока обчислювальна складність.

Другий підхід ґрунтується на методах локальної оптимізації (послідовному переборі складів ОС з деякою околиці певного складу). Використовувані при цьому евристичні методи, як правило, мають прозорі змістовні інтерпретації, але

в загальному випадку не дають оптимального рішення і тому вимагають оцінювання їхньої гарантованої ефективності.

Та, нарешті, третій підхід полягає у виключенні завідомо неефективних комбінацій агентів на підставі аналізу специфіки завдання. Наприклад, якщо можна апріорі впорядкувати претендентів на включення в команду по спадаючій ефективності їх діяльності або граничного вкладу, привносимого в команду, то задача про оптимальний склад (число можливих команд з n претендентів має порядок 2^n) Зведеться до задачі про оптимальний розмір команди (що має набагато меншу обчислювальну складність - адже з n упорядкованих претендентів можна скласти n непустих команд різного розміру). При цьому обчислювальна складність різко скорочується, і іноді вдається отримати точне (оптимальне) рішення, але, на жаль, даний підхід можна застосовувати далеко не завжди, і в кожному конкретному випадку можливість його використання вимагає відповідного обґрунтування.

Формально задача формування команди полягає в знаходженні її складу N^*, який забезпечує максимальну ефективність, яка має такий вигляд:

$$N^* = \arg\left(\max_{N \subseteq N_0} \Phi(N)\right). \qquad (2.9)$$

де N_0 – усі претенденти на місце у команді, N – дійсний склад команди, $\Phi(N)$ – функціонал ефективності, який дає кожному можливому складу команди $N \subseteq N_0$ дійсне число.

Задача (2.9) є задачею дискретної оптимізації. На допустимі склади команди можуть додатково накладатися як вимоги обов'язкового включення в неї тих чи інших груп агентів (що забезпечують реалізацію певних функцій), так і заборони на включення тих чи інших груп агентів (наприклад, таких, про які відомо, що вони мають низьку ефективність спільної діяльності).

2.1.2 Теоретико-ігрові моделі

Правила прийняття рішень в умовах невизначеності, конфліктності та зумовленого ними ризику базуються на різних концепціях. Найбільш відомою, достатньо дослідженою й широко використовуваною в теорії та на практиці є концепція теорії гри та статистичних рішень.

Теорія гри – це розділ сучасної математики, в якому вивчаються математичні моделі прийняття рішень в умовах невизначеності, конфліктності, тобто в ситуаціях, коли інтереси сторін (гравців) або протилежні (у випадку антагоністичних ігор), або не збігаються, хоча й не протилежні (у випадку ігор з непротилежними інтересами). Засновниками теорії гри є американські вчені Джон (Януш) фон Нейман та Оскар Моргенштерн [12]. У другій половині 40-х років XX ст. вони спробували за допомогою математики описати характерні для ринкової економіки явища конкуренції як деяку “гру”.

Теоретико-ігрові моделі побудовані на основі принципів теорії ігор у процесі формування та функціонування команд при розподілі обов’язків. До цього напрямку формування команд також включають модель Маршака-Рандера, де перемога команди залежить від вектору прийнятих рішень членами команди та поведінки природи, модель колективного стимулювання, де до уваги приймається лише фінальний результат без розподілення досягнень серед членів команди, не важливо хто і як досяг поставленої задачі – важливий лише результат, його швидкість, якість та задоволення первісних потреб, та модель репутації, що базується на тому, які дії і в яких ситуаціях буде виконувати той чи інший агент, і норм діяльності, коли обирається допустима норма – можливий стан природи у множині допустимих дій агентів – яка буде мати максимальну ефективність.

В своїх роботах Д. Маршак [14] запропонував наступний підхід до опису прийняття рішень в команді (при цьому під командою їм розуміється група людей з непротилежними інтересами), який отримав подальший розвиток в його власних роботах, в роботах Р. Раднер [16, 17] та ін..

Розглянемо команду (в термінології Маршака-Раднера) – безліч агентів $N = \{1,2, \dots, n\}$, в якій i-ий агент приймає рішення (вибирає дію) $x_i \in X_i, i \in N$. Виграш команди $u(x, \theta)$ залежить від вектора рішень членів команди $x = (x_1, x_2, \dots, x_n) \in X' = \prod_{i \in N} X_i$ і від стану природи $\theta \in \Omega$.

Відзначимо, що в даній моделі (як і в більшості теоретико-ігрових моделей, наслідуючих традиції Маршака-Раднера) цільові функції всіх агентів - членів команди – однакові (Більш того, в деяких роботах команда визначається саме як безліч агентів, що мають співпадаючі цільові функції). Дане припущення відображає таку властивість команди, як єдність мети діяльності її членів. Але, агенти в загальному випадку характеризуються різними множинами допустимих дій і мають різну апріорну інформацію про стан природи (сукупність цих уявлень складає інформаційну структуру команди).

Агент і приймає рішення відповідно до своєї функцією прийняття рішень – відображенням $d_i: \Omega \to X_i$, що належить безлічі допустимих відображень D_i, $i \in N$. Вектор $d = (d_1(\cdot), d_2(\cdot), \dots, d_n(\cdot))$ називається функцією прийняття рішень команди.

Фіксуємо імовірнісний розподіл $p(\cdot)$ на безлічі Ω. Очікуваний виграш команди дорівнює:

$$U\big(d(\cdot), p(\cdot)\big) = \int_{\Omega} u(d(\theta), \theta) p(\theta) d\theta. \qquad (2.10)$$

Якщо апріорі відомо імовірнісний розподіл на безлічі розподілів $p(\cdot)$, то можна обчислити Байєсівський виграш команди як математичне очікування виразу (2.10). Функція прийняття рішень $d(\cdot)$, що максимізує байєсівський виграш команди, називається байєсовою функцією прийняття рішень.

Таким чином, перше завдання – знаходження при заданій інформаційній структурі функції прийняття рішень, яка максимізувала б очікуваний виграш команди. Друге завдання - вибір інформаційної структури, яка (при використанні

відповідної оптимальної функції прийняття рішень) також максимізувала б очікуваний виграш команди. Модель Маршака-Раднер розвивалася в багатьох роботах, класичними стали роботи: Т. Гровса з розподілу ресурсу і стимулювання в командах (коли цільові функції агентів починають різнитися за рахунок введення стимулювання, яке спонукає агентів приймати узгоджені рішення), Ерроу та Раднер з вивчення впливу інформованості членів команди (інформаційної структури) на ефективність використання ресурсу.

Таким чином, модель Маршака-Раднер враховує несуперечливість цілей членів команди, які здійснюють спільну діяльність, але майже не акцентує увагу на інших характеристиках команд

2.1.3 Експериментальні дослідницькі

Експериментальні дослідницькі команди включають імітаційні експерименти та ділові ігри. Під «експериментальними» дослідженнями команд приймають ділові ігри або імітаційні експерименти, що проводяться задля перевірки тих чи інших гіпотез про процеси або ж умови формування, функціонування та загального стану команд. Ці команди можуть базуватися на одному із двох принципів: зрівняльному або ієрархічному. За результатами ділових ігор робиться висновок, що при обох механізмах має місце опортуністична поведінка членів команди (опортуністичною поведінкою називається «переслідування власного інтересу, що доходить до віроломства»).

В зрівняльному принципі всі члени команди отримують однакове винагородження за проведення роботи за результатом, який їм вдалося отримати, не звертаючи увагу на можливість одних членів команди зробити більший внесок у фінальний результат.

Цей принцип не позбавлений недоліків. Наприклад, при використанні зрівняльного механізму існує можливість, що за рахунок неформальних лідерів команди деякі її агенти розраховують на отримання додаткових привілеїв, винагород, не доклавши значних зусиль до свого прагнення. При прийнятті

центром в повторюваній грі рішень про розміри винагород агентів виявляється, що наявність центру (Ієрархічної структури команди) призводить до підвищення ефективності функціонування команди в цілому.

Ієрархічний принцип базується на виборі лідеру з команди, який буде мати більші привілеї, але ж додається в нього і відповідальність та обов'язки за інших агентів команди.

Значна кількість експериментальних досліджень присвячено вивченню уподобань людей приймати рішення індивідуально або в рамках команди, а також аналізу ефективності діяльності реальних команд в порівнянні з групами індивідуумів. Так, по результатам, цих [30] досліджень виявлено, що більше 60% людей вважають за краще колективне прийняття рішень в рамках команди.

Але ж на практиці можна відслідкувати ефект «справедливості», адже кожен агент команди зацікавлений, щоб його здібності були оцінені у повному обсязі. Тому, якщо ми візьмемо на розгляд команду, що складається з двох членів, то прагнення справедливості виражається в тому, що агента цікавить не тільки його винагорода за вклад в досягнення мети, а й винагорода його колеги. Цей механізм можна детальніше розглянути на наступному прикладі.

Нехай до команди входить n агентів, що мають цільові функції $f_i(\cdot)$, $i \in N$. Побудуємо цільові функції цієї команди (2.11):

$$f_i(\bullet) = f_i(\bullet) - \sum_{j \neq i} \alpha_{ij} \max\{0;\, f_i - f_j\} - \sum_{j \neq i} \beta_{ij} \max\{0;\, f_i - f_j\}, \qquad (2.11)$$

де невід'ємні константи α_{ij} та β_{ij} відображають вразливість i-го агента до того, що винагорода іншого члена групи буде більша за його винагороду.

У еволюційних іграх передбачається, що поведінка в типовій, багаторазово повторюємій взаємодії являє собою підсумок еволюційного процесу формування поведінки. Тому, на відміну від класичної теорії ігор, носить нормативний характер, еволюційна теорія ігор намагається передбачити, яку стратегію

поведінки оберуть гравці, при цьому функція виграшу характеризує успіх стратегії, а не окремого гравця. У багатьох випадках використовується еволюційний підхід, в якому процес формування поведінки (Модифікації стратегій) описується динамічною системою, визначальною розподіл по стратегіях всередині популяції в залежності від результатів взаємодії в попередні моменти часу. Цей процес сходиться до того чи іншого рівноваги.

Дана класифікація тестує, чому члени команди діють так чи інакше і як вони реагують на дії їх колег по команді.

2.1.4 Рефлексивні моделі

Рефлексивні моделі використовують методи рефлексивних ігор для побудови моделі взаємодії всередині команди. Також за однією з класифікацій [1] команди поділяють на однорідні та неоднорідні.

Основна мета однорідних команд – розподілення об'єму робіт серед членів команди. Розглянемо цей тип команд на прикладі, нехай у команді буде N агентів. Стратегія дій x_i, що потребує від нього витрат $c_i(x_i, r_i)$, де $r_i > 0$ – тип агента, що відображає ефективність його діяльності. Нехай метою всіх дій агентів буде досягнення результату, що відображає їх сумарні дії, за мінімальних затрат. Таким чином, задачу умовної оптимізації можна подати у такий вигляд (2.12):

$$\sum_{i \in N} \frac{x_i^2}{2r_i} \to \min_{\{x_i > 0 \sum_{i \in N} x_i = 1\}} . \qquad (2.12)$$

Ідея полягає в тому, що кожен агент моделює поведінку своїх опонентів на підставі власного сприйняття їх дій, умов, в яких вони їх виконують.

В неоднорідних командах члени групи виконують різні ролі, або ж навіть поєднують у собі декілька ролей. Нехай команда складається з n ($N = \{1, 2, 3, \ldots, n\}$ агентів, а успішна діяльність команди потребує виконання та досягнення певних результатів, що знаходяться у множині $M = \{1, 2, 3, \ldots, m\}$, а

ефективність виконання *i*-м членом команди *j*-го завдання позначимо через $r_i^j \geq 0$, де $i \in N,\ j \in M$.

У даного класу команд використано показник, що визначає здатність команди до виконання завдань певного рівня.

Наприклад, професіоналізм певного агенту можна визначити за наступною формулою (2.13):

$$r_i = \frac{1}{m}\sum_{j=1}^{m} r_i^j,\ i \in N. \tag{2.13}$$

Професіоналізм команди можна визначити як середню здатність виконання завдань різноманітної складності (2.14):

$$r = \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} r_i^j\ . \tag{2.14}$$

Середня кваліфікація команди по кожній з функцій визначається за формулою (2.15):

$$H^j = \frac{1}{n}\sum_{i=1}^{n} r_i^j,\ j \in M\ . \tag{2.15}$$

Неоднорідність кваліфікації певного агенту – це відхилення його ефективності виконання завдань:

$$d_i = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(r_i^j - r_i)^2},\ i \in N. \tag{2.16}$$

Неоднорідність команди визначається як сума різниць ефективностей агентів (2.17):

$$d = \frac{1}{2mn(n-1)} \sum_{i,k=1}^{n} \sum_{j=1}^{m} | r_i^j - r_k^j | \cdot \qquad (2.17)$$

Спеціалізованість команди, що характеризує наявність в ній для кожної функції агентів, що спеціалізуються саме на реалізації даної функції. Даний показник визначимо як відношення числа членів команди, що виконують при оптимальному розподілі функцій будь-які функції, до загальної кількості членів команди n.

Отже, підсумовуючи результати аналізу, можна сформулювати фактори, які потрібно виконати під час формування команди, задля досягнення мети за найменших ресурсних затрат:

- всі члени команди повинні знати, які вимоги вони повинні виконати;
- кожен член команди має чітко позначені рамки самостійної діяльності;
- команда повинна мати стимул задля досягнення поставленої цілі;
- повинен бути вільний комунікативний зв'язок серед членів команди, керівництва.

2.2 Математична модель організаційного управління складом виконавців комплексу робіт з невизначеним ступенем складності

Команда проекту розглядається як розподілена система однорідних груп виконавців комплексів робіт різного ступеня складності, фінансуються єдиним центром і відрізняються рівнем кваліфікації. Функціонування окремої групи виконавців *i*-ої кваліфікації формалізується моделлю чинника з порядку спадання віддачею

$$Q_i = a_i (R_i + x_i)^{\alpha_i} y_i^{\beta_i}, \qquad (2.18)$$

де $\alpha_i + \beta_i < 1; 0 < \alpha_i, \beta_i < 1;\ \forall_i \in \overline{\{1, n\}}$; Q_i – обсяг робіт, що виконується i-ою групою виконавців проекту; R_i – технічне забезпечення i-ї групи виконавців проекту; x_i – капіталовкладення в технічне забезпечення i-ї групи виконавців проекту; y_i – оборотні кошти i-ї групи виконавців проекту; a_i, α_i, β_i – параметри моделі i-ї групи виконавців проекту.

Планований обсяг робіт i-го ступеня складності Q_i визначає собою необхідну кількість N_i виконавців i-ої кваліфікації

$$N_i = Q_i/(\Phi_{\text{Д}_\text{і}} \text{К}_{\text{вн}_\text{і}}),, \tag{2.19}$$

де $\Phi_{\text{Д}_\text{і}}, \text{К}_{\text{вн}_\text{і}}$ – дійсний фонд часу і коефіцієнт виконання норм окремого виконавця i-ої кваліфікації відповідно.

Реалізація проекту розглядається як багатокроковий процес виконання запланованих робіт різного ступеня складності шляхом поетапного фінансування. На кожному кроці фінансування проекту ефективність команди виконавців оцінюється відношенням загального обсягу виконаних робіт до сумарних витрат фінансових коштів

$$E(\bar{x}, \bar{y}) = \sum_{i=1}^{n} a_i (R_i, x_i)^{\alpha_i} {y_i}^{\beta_i} / \sum_{i=1}^{n} (x_i + y_i), \tag{2.20}$$

де $\bar{x} = x_1, \dots, x_n; \bar{y} = y_1, \dots, y_n$.

Розглянуті моделі формалізації організаційного управління персоналом проекту дозволяють підвищити ефективність формування складу виконавців в умовах неповної інформованості про ступінь складності запланованих робіт. При заданому терміні реалізації проекту визначити кількісний склад виконавців необхідної кваліфікації виконують комплекси робіт різного ступеня складності. При обмежених фінансових ресурсах розрахувати необхідні обсяги капітальних

вкладень і оборотних коштів при поетапному фінансуванні окремих груп виконавців різної кваліфікації, відповідних централізованого, узгодженим і відкритого принципам організаційного управління персоналом проекту.

2.3 Модель та метод оптимізації складу виконавців проектів з розроблення програмного забезпечення згідно з методологією MSF

Модель процесів MSF (MSF process model) представляє загальну методологію розробки та впровадження IT-рішень. Особливість цієї моделі полягає в тому, що завдяки своїй гнучкості і відсутності жорстко нав'язуваних процедур вона може бути застосована при розробці широкого кола проектів. Ця модель поєднує в собі властивості двох стандартних моделей: каскадної (waterfall) та спіральної (spiral). Модель включає наступні фази: "Вироблення концепції", "Планування", "Розробка", "Стабілізація", "Впровадження".

Модель проектної групи MSF (MSF Team Model) описує підхід Microsoft до організації, працюючого над проектом персоналу для максимізації успішності виконання проекту. Дана модель визначає рольові кластери виконавців, їх області компетенції та зони відповідальності, а також рекомендації членам проектної групи, що дозволяють їм успішно виконати проект. Відповідно до моделі MSF проектні групи будуються як невеликі багатопрофільні команди, члени яких розподіляють між собою відповідальність і доповнюють області компетенцій один одного. Це дає можливість чітко сфокусувати увагу на потребах проекту. Проектну групу об'єднує єдине бачення проекту, прагнення до втілення його в життя, високі вимоги до якості роботи і бажання самовдосконалюватися.

Модель проектної групи включає наступні рольові кластери:

– Архітектура (architect) – відповідає за архітектуру програмного забезпечення;
– Управління продуктом (product management) – бізнес-пріоритети, маркетинг, представництво інтересів замовника;

- Управління програмою (program management) – розробку архітектури рішення, адміністративні служби;
- Розробка (development) – розробку додатків та інфраструктури, технологічні консультації;
- Тестування (test) – планування, розробку тестів і звітність по тестах;
- Задоволення споживача (user experience) – навчання, ергономіку, графічний дизайн, технічну підтримку;
- Управління випуском (release management) – інфраструктуру, супровід, бізнес-процеси, випуск готового продукту.

Вони відповідальні за різні області компетенції (functional areas) та пов'язані з ними цілі і завдання проекту. Одна роль (або один кластер) може бути представлена одним або декількома співробітниками в залежності від розміру проекту, його складності та професійних навичок.

Методологія MSF може застосовуватися в будь-яких проектах: починаючи з малих, в яких задіяні 2-3 виконавці, і закінчуючи великими (сотні виконавців). Основна ідея проектної моделі полягає в тому, що в команді повинні бути обов'язково представлені всі сім ролей (рольових кластерів). Зазвичай виділення як мінімум однієї людини на кожен рольової кластер забезпечує захист інтересів кожної з ролей, але це економічно виправдано не для всіх проектів. Як правило, члени проектної групи можуть об'єднувати ролі. У малих проектних групах об'єднання ролей є необхідним. При цьому повинні дотримуватися два принципи: роль команди розробників не може бути об'єднана ні з якою іншою роллю, другий принцип - це уникнення поєднання таких ролей, що мають конфлікти власних інтересів. Наприклад, менеджмент продукту має на меті задовольнити замовника, в той час як менеджмент програми забезпечує готовність продукту у відведений час і в рамках чинного бюджету. У разі поєднання цих ролей виникає ризик, що затребувана замовником зміна або не буде розглянута, або буде прийнята без належного аналізу впливу результатів на проект. Подання цих ролей різними людьми в проектній команді забезпечує рівновагу двох суперечливих точок зору. Те ж саме відноситься до можливості об'єднання ролей розробки і тестування.

Використання рольових кластерів не має на увазі і не нав'язує ніякої спеціальної структури організації або обов'язкових посад. Адміністративний склад ролей може широко варіюватися в різних організаціях і проектних групах. Найчастіше ролі розподіляються серед різних підрозділів однієї організації, але іноді частина їх відводиться спільноті споживачів або зовнішнім по відношенню до організації консультантам і партнерам. Ключовим моментом є чітке визначення працівників, відповідальних за кожен рольової кластер, їх функцій, відповідальності і очікуваного вкладу в кінцевий результат.

Модель проектної групи MSF не забезпечує успіх сама по собі. Є багато інших факторів, що визначають успіх чи невдачу проекту, але структура проектної групи, безумовно, вносить істотний внесок.

Відповідна структура команди є фундаментом успіху, і реалізація моделі MSF з її використанням лежать в основі принципів, допоможе зробити проектні групи більш ефективними і, як наслідок, більш успішними.

Розглянемо ситуацію, коли в команді існує різна кількість виконавців, які спільно об'єднують всі сім рольових кластерів. Нехай існують виконавці X = {X1, X2, X3, X4, X5, X6, X7, ... Xn}, які володіють різними поєднаннями навичок рольових кластерів. При цьому виконавці мають навички не менше ніж двох доповнюючих один одного рольових кластерів. Можливі наступні об'єднання ролей для виконавців X (табл. 2.1).

Таблиця 2.1 – Варіанти несуперечливого об'єднання ролей моделі проектної групи за методологією MSF

Виконавець	Роль
X1	P1, P3, P4
X2	P2, P5, P6
X3	P1, P3, P7
X4	P1, P4
X5	P2, P5, P6, P7

Продовження таблиці 2.1

Виконавець	Роль
X6	P3, P5, P7
X7	P1, P3
X8	P2, P5
X9	P2, P6
X10	P3, P7
X11	P5, P7

Таким чином, можливі наступні об'єднання виконавців в рамках команд згідно з моделлю проектної групи MSF (табл. 2.2).

Таблиця 2.2 – Склад виконавців команд моделі MSF

№ команди	Склад виконавців одної команди
K1	X1, X5
K2	X2, X1
K3	X4, X6
K4	X6, X1
K5	X7, X6, X4
K6	X8, X7, X4, X11
K7	X9, X7, X4, X11
K8	X9, X3, X4
K9	X10, X3, X4
K10	X10, X2
K11	X11, X7, X8, X4
K12	X11, X1, X8
K13	X11, X4, X9, X10

Оскільки проект виконується за фазами (модель процесів), то обов'язковою умовою виконання фази є використання хоча б однієї команди. Ідеальним рішенням є використання однієї команди для виконання всього проекту.

2.4 Метод розв'язку задачі

Передбачає розробку і дослідження підходів і методів з урахуванням того, що кількість проектів в організації може збільшуватися, що потребує перерахунку як кількості команд, задіяних при виконанні різних проектів, так і складу їх виконавців для своєчасного та якісного виконання робіт усіх проектів.

У зв'язку з наведеними особливостями, що накладаються на рольові кластери, згідно з рекомендаціями стандарту MSF 4.0 розроблені наступні варіанти можливих об'єднань ролей (кластерів), зведені в табл. 2.3.

Таблиця 2.3 – Варіанти об’єднання ролей в моделі проектної групи MSF

	Архітектура (P_1)	Управління продуктом (P_2)	Управління програмою (P_3)	Розробка (P_4)	Тестування (P_5)	Задоволення споживача (P_6)	Управління випуском (P_7)
Архітектура (P_1)		Ні	Так	Так	Не бажано	Не бажано	Не бажано
Управління продуктом (P_2)	Ні		Ні	Ні	Так	Так	Не бажано
Управління програмою (P_3)	Так	Ні		Ні	Не бажано	Не бажано	Так

Продовження таблиці 2.3

	Архітектура (P_1)	Управління продуктом (P_2)	Управління програмою (P_3)	Розробка (P_4)	Тестування (P_5)	Задоволення споживача (P_6)	Управління випуском (P_7)
Розробка (P_4)	Так	Ні	Ні		Ні	Ні	Ні
Тестування (P_5)	Не бажано	Так	Не бажано	Ні		Так	Так
Задоволення споживача (P_6)	Не бажано	Так	Не бажано	Ні	Так		Не бажано
Управління випуском (P_7)	Не бажано	Не бажано	Так	Ні	Так	Не бажано	

Покладемо, що окрему фазу проекту можна реалізувати проектними командами, що мають різний склад виконавців при обмеженні на бюджет C_i (обмеженні на вартість) *i*-ї фази проекту. Враховуючи різний склад і кількість проектних команд, які можуть реалізувати окремий проект (фазу проекту) при наявному обмеженні C_i на вартість (бюджет), сформуємо наступне нелінійне рівняння:

$$\sum_{ij} b_{ij} X_{ij} X_{ik} X_{il} = C_i , \qquad (2.21)$$

де b_{ij} – коефіцієнт, що враховує: кількість команд *j*-го типу, кількість виконавців *j*-ї команди, вартість одиниці часу виконання *i*-ї фази і комунікацій

між членами команд для виконавців *j*-ї команди, що реалізують розглянуті поєднання ролей для різних виконавців проекту; C_i – вартість (бюджет) проекту.

Нехай є виконавці $X = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7, \ldots X_n\}$, які мають (або наділені) різними поєднаннями навичок рольових кластерів, необхідних для виконання проекту відповідно до вимог. Для прикладу розглянемо типову ситуацію, коли виконавці мають навички не менше двох доповнюють один одного рольових кластерів (табл. 2.1) та по ним складені проектні команди, згідно до наведених вище рекомендацій (табл. 2.2).

Тоді постановка задачі оптимізації складу розробників, в яких реалізуються найбільш ефективне поєднання ролей (рольових кластерів) зводиться до наступної: визначити оптимальний склад по виконавцям $\{X_t\}$ для реалізації проекту в рамках обмежень на вартість проекту і його окремих фаз, а також на час виконання (тривалість) проекту і його окремих фаз. Будемо вважати, що окрему фазу проекту можна реалізувати при бюджеті C_i (обмеження на вартість) *i*-ої фази проекту.

З огляду на різну кількість проектних команд і різний склад команд, за допомогою яких можна реалізувати окрему фазу проекту за наявного обмеження C_i на її вартість, сформуємо наступне нелінійне рівняння:

$$b_{i1}\cdot X_1\cdot X_5 + b_{i2}\cdot X_2\cdot X_1 + b_{i3}\cdot X_4\cdot X_6 + b_{i4}\cdot X_6\cdot X_1 + b_{i5}\cdot X_7\cdot X_6\cdot X_4 +$$
$$b_{i6}\cdot X_8\cdot X_7\cdot X_4\cdot X_{11} + b_{i7}\cdot X_9\cdot X_7\cdot X_4\cdot X_{11} + b_{i8}\cdot X_9\cdot X_3\cdot X_4 + + b_{i9}\cdot X_{10}\cdot X_3\cdot X_4 + b_{i10}\cdot X_{10}\cdot X_2 + \quad (2.22)$$
$$b_{i11}\cdot X_{11}\cdot X_7\cdot X_8\cdot X_4 + b_{i12}\cdot X_{11}\cdot X_1\cdot X_8 + b_{i13}\cdot X_{11}\cdot X_4\cdot X_9\cdot X_{10} = C_i.$$

Для формалізації умови виконання всіх робіт *i*-ой фази введемо ще один параметр β_j, що враховує участь (або неучасть) *j*-ой проектної команди з певним складом виконавців, що поєднує в собі необхідний набір рольових компетенцій, у виконанні *i*-ой фази:

$$\beta_i = \begin{cases} 1, \textit{якщо проектна команда і} \\ \textit{приймає участь в j-й фазі;} \\ 0, \textit{в іншому випадку.} \end{cases}, \qquad (2.23)$$

Тоді рівняння (2.22) прийме наступний вигляд:

$$b_{i1}\cdot \beta_1 \cdot X_1 \cdot X_5 + b_{i2}\cdot \beta_2 \cdot X_2 \cdot X_1 + b_{i3}\cdot \beta_3 \cdot X_4 \cdot X_6 + b_{i4}\cdot \beta_4 \cdot X_6 \cdot X_1 + b_{i5}\cdot \beta_5 \cdot X_7 \cdot X_6 \cdot X_4 + b_{i6}\cdot \beta_6 \cdot X_8 \cdot X_7 \cdot X_4 \cdot X_{11} + b_{i7}\cdot \beta_7 \cdot X_9 \cdot X_7 \cdot X_4 \cdot X_{11} + b_{i8}\cdot \beta_8 \cdot X_9 \cdot X_3 \cdot X_4 + + b_{i9}\cdot \beta_9 \cdot X_{10} \cdot X_3 \cdot X_4 + b_{i10}\cdot \beta_{10} \cdot X_{10} \cdot X_2 + b_{i11}\cdot \beta_{11} \cdot X_{11} \cdot X_7 \cdot X_8 \cdot X_4 + b_{i12}\cdot \beta_{12} \cdot X_{11} \cdot X_1 \cdot X_8 + b_{i13}\cdot \beta_{13} \cdot X_{11} \cdot X_4 \cdot X_9 \cdot X_{10} = C_i. \qquad (2.24)$$

У разі оптимізації складу розробників для виконання робіт за усьома фазами проекту для вирішення поставленого завдання необхідно додатково сформувати кількість рівнянь, яка визначається кількістю фаз проекту згідно з використовуваною методологією виконання проекту (в моделі MSF 4.0 використовується 5 фаз проекту). В результаті отримаємо систему нелінійних булевих рівнянь.

Поставлена задача зводиться до обчислення коренів рівняння (2.24), яке є нелінійним булевим рівнянням (або системою нелінійних булевих рівнянь типу (2.22)), рішення якої носить комбінаторний характер. Для отримання рішень далі пропонується формалізація і алгоритми розвязку поставленого завдання на основі рангового підходу до вирішення комбінаторних задач [19].

У загальному випадку нелінійне рівняння можна представити у вигляді

$$\sum_{j=1}^{p_1} C_{1j} S_1(C_n^1) + \sum_{j=1}^{p_2} C_{2j} S_2(C_n^2) + \ldots + \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k) + \ldots + \sum_{j=1}^{p_n} C_{nj} S_n(C_n^n) = \mathrm{b}, \qquad (2.25)$$

де $Sr(C_n^r) = S_1 + S_2 + \ldots + S_{p_r}$ - сума всіх можливих поєднань добутків змінних, що містять в кожному добутку $Sr = X_p X_к \ldots X_m$ r різних змінних; $p_r = \frac{n!}{r!(n-r)!}$; C_{rj} - коефіцієнти що стоять в добутках S_r містять r змінних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Для визначеної задачі необхідно написати програмну реалізацію. Програмний засіб повинен вирішувати поставлену в першому розділі задачу за допомогою математичної моделі описаної в другому розділі. Програмний засіб повинен отримувати від користувача набір значень відповідно до проектної команди та видавати результат, на основі якого можна зробити висновок про найліпший склад команди для проекту або певних його фаз.

Весь алгоритм роботи програми можна описати наступними кроками:

- Отримання вхідних даних від користувача;
- Генерація системи рівнянь;
- Розрахунок всіх можливих комбінацій X;
- Знаходження всіх можливих розв'язків рівнянь;
- Знаходження спільних розв'язків рівнянь;
- Вивід результатів роботи.

Крім того, необхідно розробити зручний та зрозумілий інтерфейс користувача та введення і виведення даних із програми.

3.1 Вибір середовища та мови розробки

Для реалізації алгоритму була вибрана мова програмування C# та фреймворк .Net. Для створення графічного інтерфейсу використовується технологія Windows Forms, для передачі даних між користувачем та програмою використовується графічний інтерфейс або текстовий файл.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі

моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів, що було реалізоване в C++.

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#. Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

Microsoft .NET – програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна збірка в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET – крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD, від Microsoft і варіант технології для ОС Linux в проекті Mono, в рамках угоди між Microsoft з Novell, DotGNU.

.NET поділяється на дві основні частини – середовище виконання, по суті віртуальна машина та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор.

Як і технологія Java, середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL

(Microsoft Intermediate Language), або просто IL. Застосування байт-кода дозволяє отримати крос-платформовість на рівні скомпільованого проекту, а не на рівні сирцевого тексту, як, наприклад, в C. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище JIT-компілятором в машинні коди цільового процесора.

Windows Forms - інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача, і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код - класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C #, C ++, так і на VB.Net, J # і ін.

3.2 Розробка програмного засобу

В цілому алгоритм роботи програмного засобу можна описати наступною діаграмою варіантів використання (рис. 3.1).

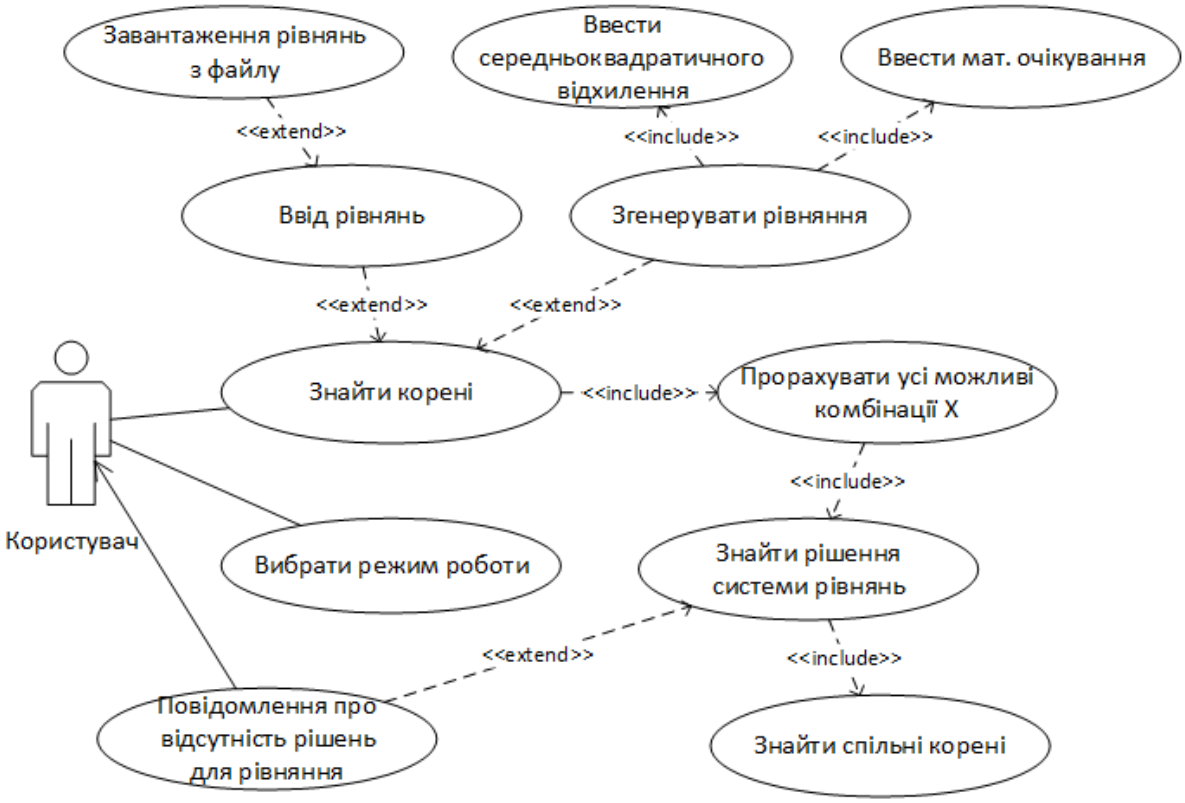

Рисунок 3.1 – Діаграма варіантів використання програмного засобу

З діаграми видно, що користувач може обрати режим роботи програми, та почати пошук коренів. Залежно від режиму, є два варіанти подальшого сценарію, перший – користувач вводить дані про рівняння в систему або завантажує їх з файлу, другий – система сама генерує рівняння, користувач повинен ввести лише значення математичного очікування та середньоквадратичне відхилення. Далі система прораховує усі можливі комбінації X та знаходить рішення системи рівнянь і спільні корені. Якщо рішення для рівняння не знайдено, користувач отримає про це повідомлення.

Розглянемо схеми деяких алгоритмів програми. При виборі автоматичного режиму, користувач може обрати між двома сценаріями, які генерують рівняння системи. На рис. 3.2 наведена схема алгоритму генерації параметрів системи для першого сценарію:

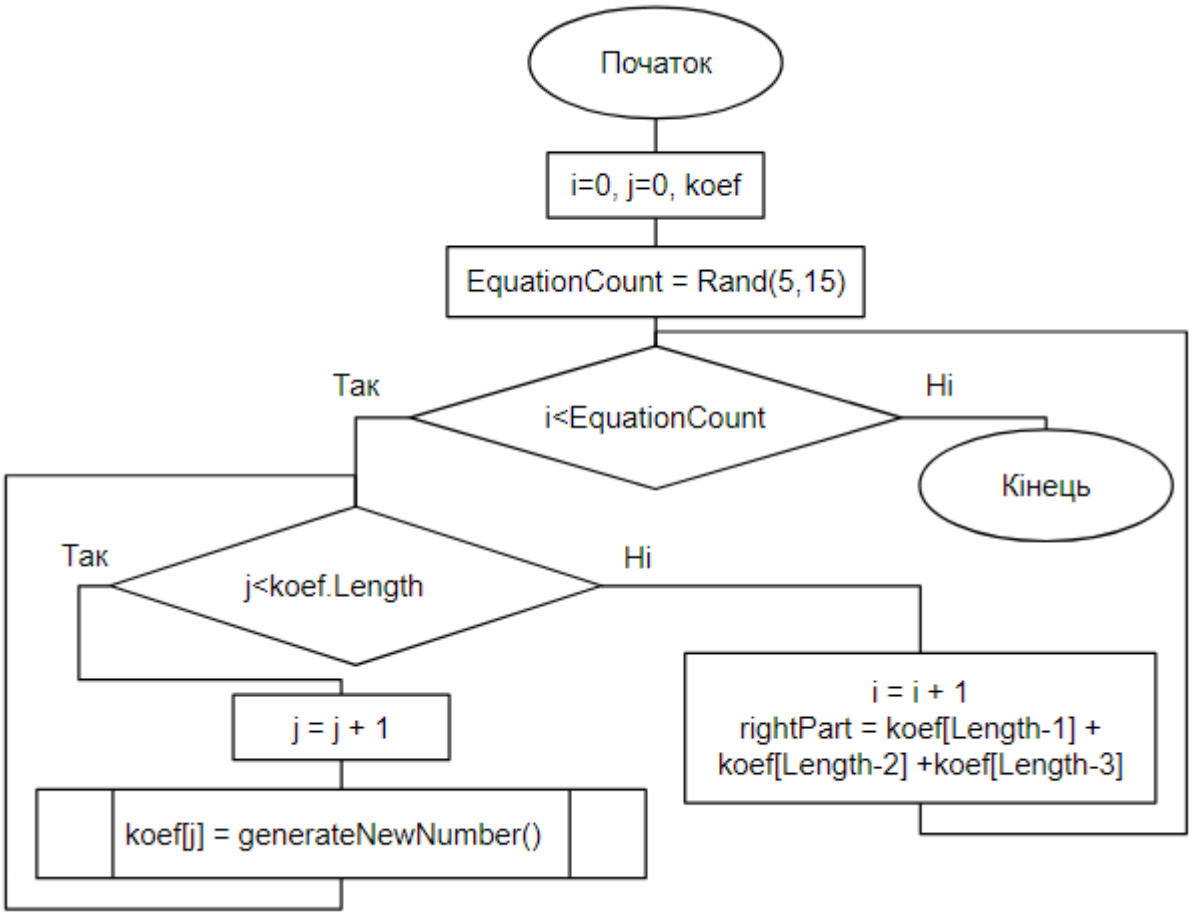

Рисунок 3.2 – Схема алгоритму роботи функції GenerateSystem1

Позначки: koef – масив коефіцієнтів команд; EquationCount – кількість рівнянь; rightPart – права частина рівняння; generateNewNumber – функція, яка генерує випадкові значення з урахуванням математичного очікування та середньоквадратичного відхилення. Функція GenerateSystem1 генерує систему з x рівнянь, x є [5;15]. Послідовно при кожній ітерації система генерує коефіцієнт за нормальним розподільним законом зі середнім значенням та відхиленням, що задаються користувачем. Права частина рівняння генерується як сума останніх трьох коефіцієнтів її лівої частини. Для другого сценарію, алгоритм роботи програми майже той самий, за винятком того, що права частина генерується як випадкове число. Після чого починається рішення кожного з рівнянь. На рис. 3.3 наведена схема алгоритму пошуку розв’язків рівняння.

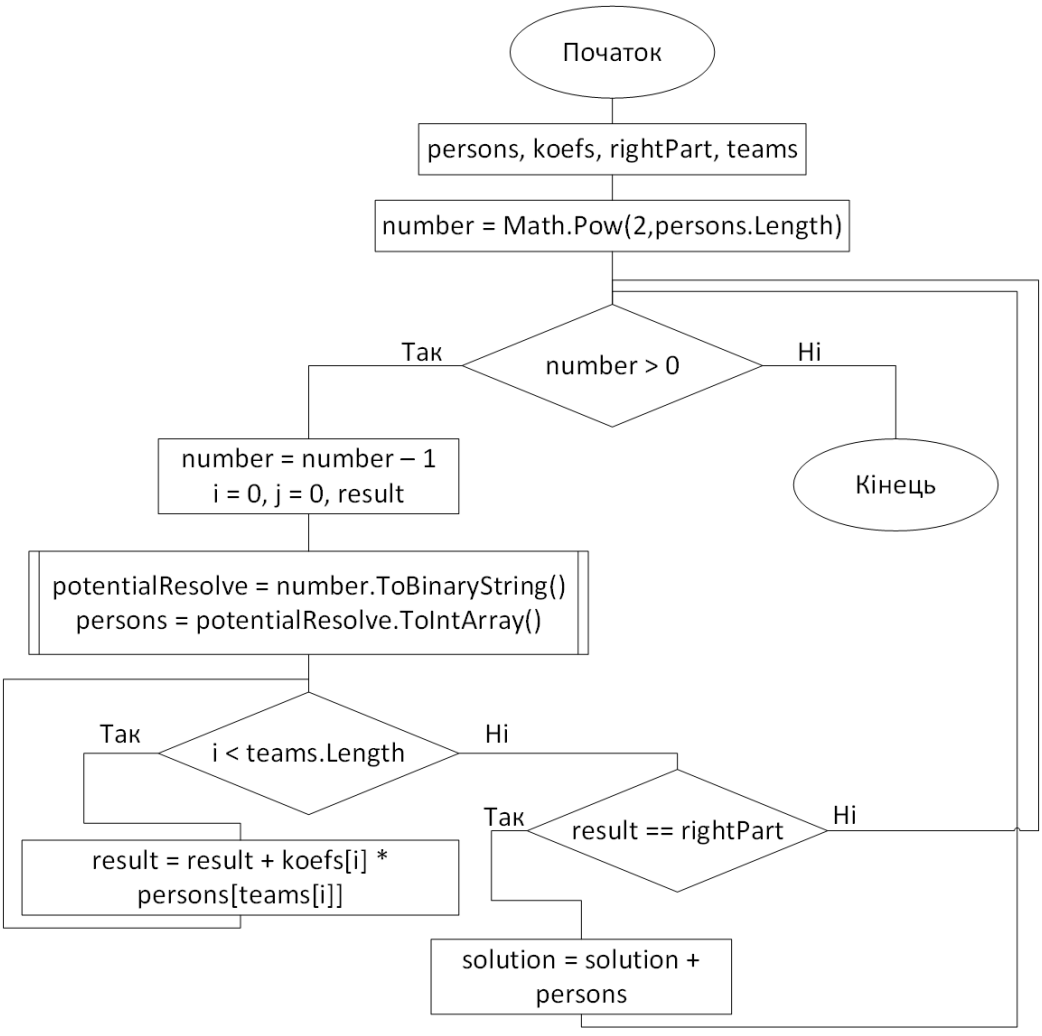

Рисунок 3.3 – Схема алгоритму роботи функції SolveEquation

Позначки: persons – масив X, в якому для кожного значення є два варіанти 1:0; koefs – масив коефіцієнтів команд; rightPart – значення правої частини рівняння; teams – масив номерів X у конкретній команді; number – кількість усіх можливих комбінацій; solution – масив усіх можливих рішень рівняння.

Алгоритм працює за таким принципом: спочатку в залежності від кількості X у системі, розраховується число, яке при переводі у двійкову систему буде складати послідовність одиниць. Зменшуючи це число, буде отримана нова послідовність одиниць. Далі ця послідовність записується до масиву persons. Послідовно рухаючись масивом шукаємо невідому з номером, що співпадає з поточним індексом значення в рядку і записуємо це значення як значення невідомого. Підставивши таким чином всі значення, виконуємо всі дії в рівнянні і перевіряємо отримане значення. Якщо відповідь вірна, записуємо її до масиву рішень для кожного рівняння. Наведений вище алгоритм повторюється для кожного рівняння системи.

На рис. 3.4 зображена схема алгоритму фунції GenerateNewNumber, яка генерує випадкові числа, враховуючи математичне очікування та середньоквадратичне відхилення.

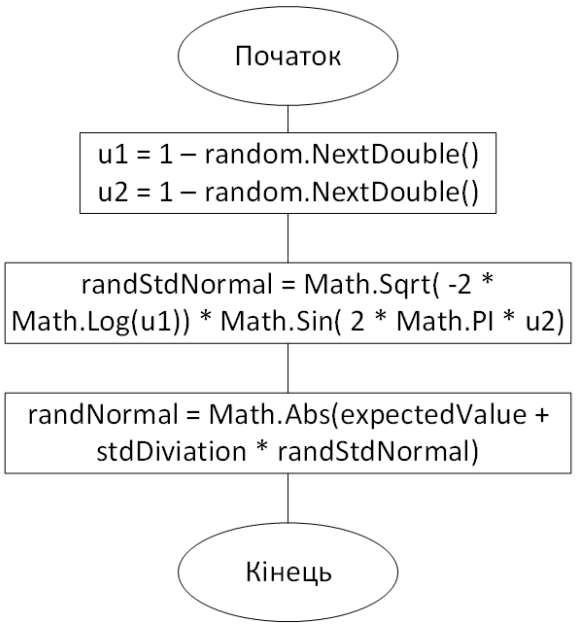

Рисунок 3.4 – Схема алгоритму роботи функції GenerateNewNumber

3.3 Опис принципів роботи програми

В ході вирішення задачі оптимізації складу команд з розроблення програмного забезпечення створено програмний продукт, що може бути використаний для знаходження найліпшого співвідношення кількості членів команди та їх ролей за методологією MSF, за яким пропонується вирішення задач з обмеженням на час та бюджет виконання проекту. Програмний продукт призначений як для виконання стандартних сценаріїв дій, так і для дій за сценарієм користувача, враховуючи його особисті побажання щодо технічних та якісних показників - кількості рівнянь системи, величини бюджету та коефіцієнту b при кожній можливій комбінації змінних. При цьому b_{ij} – коефіцієнт, що враховує кількість команд j-го типу, кількість виконавців у j-й команді, вартість одиниці часу виконання i-ої фази і комунікації між членами команд для виконавців у j-ій команді.

При запуску програми на виконання перед користувачем з’являється початкове вікно програми (рис. 3.5), в якому пропонується обрати один з трьох сценаріїв роботи програми.

У початковому вікні є два режими вибору роботи програми, «Автоматичний» який надає вибір користувачу між двома сценаріями, та «Ручний» в якому користувач може ввести свої значення. Також на початковому екрані є поля для відображення рівнянь системи, їх коренів та коренів, що є спільними для сумісної системи, поля для введення значень математичного очікування та середньоквадратичного відхилення, та час (у мілісекундах), що був витрачений на генерування системи рівнянь, знаходження рішення для кожного з рівнянь та знаходження спільних коренів для рівнянь системи.

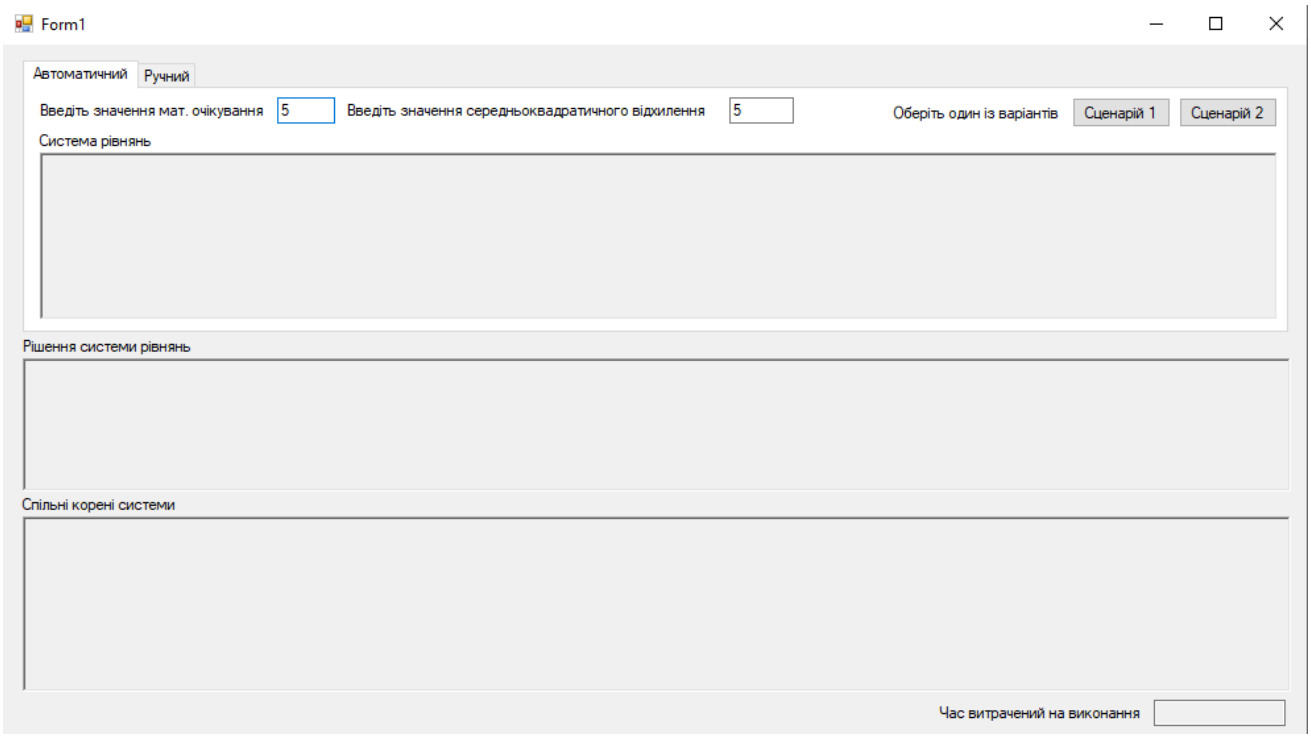

Рисунок 3.5 – Початкове вікно програми

Обравши математичне очікування та середньоквадратичне відхилення випадкових коефіцієнтів при складових рівнянь та натиснувши кнопку «Сценарій 1», користувач запускає виконання програми за сценарієм 1 та отримує дані щодо згенерованої системи.

Робота програми за сценарієм 1. При запуску програми за сценарієм 1 програма генерує систему з x рівнянь, x є [5;15]. Послідовно при кожному додатку система генерує коефіцієнт за нормальним розподільним законом зі середнім значенням та відхиленням, що задаються користувачем. Права частина рівняння генерується як сума останніх трьох коефіцієнтів її лівої частини. Після чого починається рішення кожного з рівнянь. Нехай усі невідомі дорівнюють одиниці, тоді, записавши їх послідовно і привівши отриману послідовність одиниць в десяткову систему, отримаємо 2047. Це означає, що, поступово зменшуючи отримане число на одиницю і перевівши його у двійкову систему, отримуємо нову послідовність значень 0 та 1. Записавши цю послідовність в рядок, можна сформулювати рішення наступним чином: послідовно рухаючись рядком шукаємо невідому з номером, що співпадає з поточним індексом значення в рядку

і записуємо це значення як значення невідомого. Підставивши таким чином всі значення, виконуємо всі дії в рівнянні і перевіряємо отримане значення. Якщо відповідь вірна, записуємо її до масиву рішень для кожного рівняння. Наведений вище алгоритм повторюється для кожного рівняння системи. Після розв'язку всіх рівнянь система шукає рівняння, що містять однакові корені і записує номера рівнянь та рішення, що є для них спільними (тобто є сумісними).

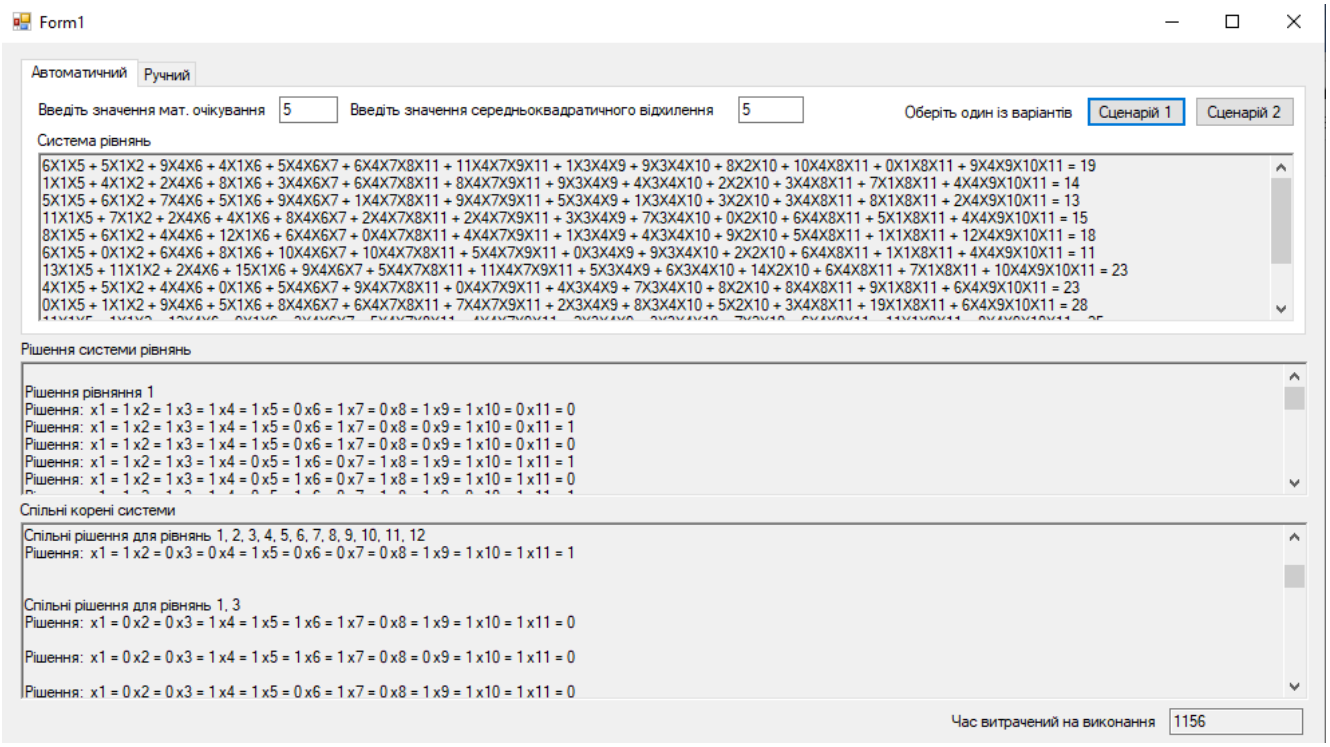

Рисунок 3.6 – Результати роботи програми за сценарієм 1

З результатів роботи програми видно що для наведеної системи було знайдено спільне рішення. Команда, у складі якої будуть X1, X4, X8, X9, X10, X11, є сумісною для усіх рівнянь (фаз) проекту та може бути застосована як оптимальне рішення для виконання усього проекту.

Робота програми за сценарієм 2. Вказавши математичне очікування та середньоквадратичне відхилення та натиснувши кнопку «Сценарій 2», користувач запускає виконання програми за сценарієм 2 та отримує данні про згенеровану систему (рис. 3.7).

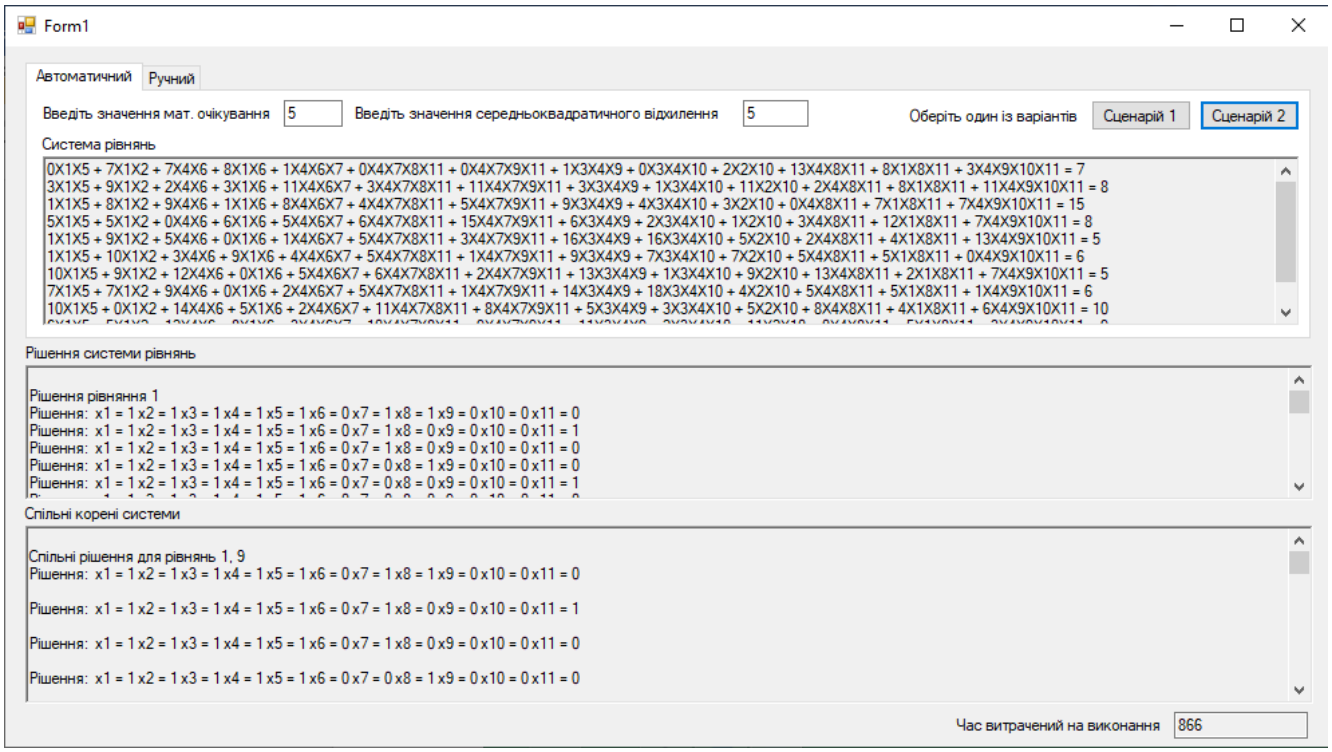

Рисунок 3.7 – Результати роботи програми за сценарієм 2

При запуску програми за сценарієм 2 програма генерує систему з кількості x рівнянь, $x \in [5;15]$. Послідовно при кожному додатку система генерує число за нормальним законом розподілу зі середнім значенням та відхиленням, що задаються користувачем. Права частина рівняння генерується за принципом генерування коефіцієнтів. Наступні дії сценарію 2 співпадають із сценарієм 1.

З рисунку 3.7 видно, що за результатами розрахунків система не знайшла спільних рішень для усього проекту. Були знайдені сумісні рішення лише для деяких окремих фаз. Це означає, що система не є сумісною та потребує аналізу та корегування.

У вікні ручного режиму, як і на початковому, представлені області для відображення системи рівнянь, рішень рівнянь системи та спільних рішень системи, а також часу, що витрачений на проведення розрахунків. В цьому вікні користувач може ввести свої рівняння у систему та знайти для них рішення та спільні корені (рис. 3.8).

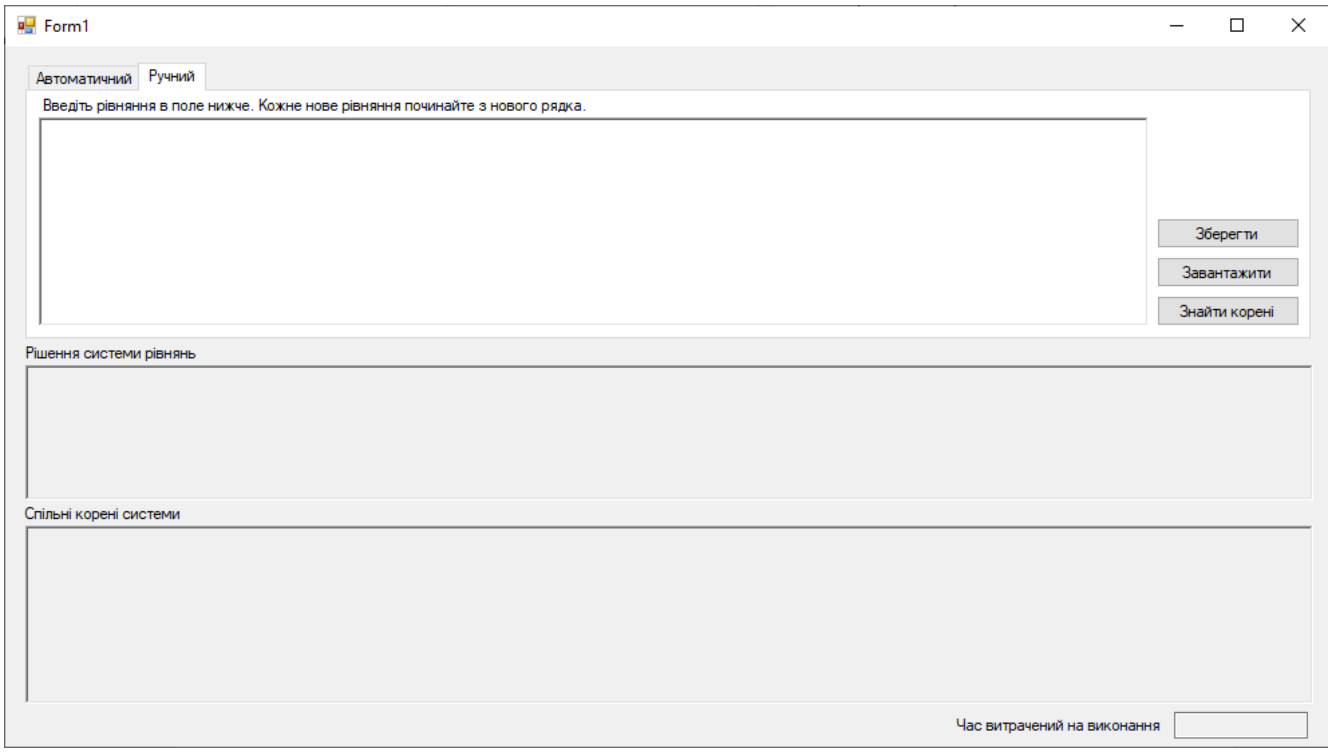

Рисунок 3.8 – Вікно програми для роботи у ручному режимі

Робота програми за сценарієм 3. Після натиснення на кнопку режиму «Ручний» з’являється вікно (рис. 3.8) для виконання роботи програми за сценарієм 3.

При запуску програми за сценарієм 3 користувачеві системи пропонується ввести рівняння, значення коефіцієнтів при складових цих рівнянь та значення правої частини рівнянь системи (рис. 3.9). Далі робота програми за сценарієм 3 співпадає із роботою програми за сценарієм 1.

Три сценарії призначені для знаходження оптимального балансу між складами програмних команд проектів, обмеженнями на час їх виконання та бюджет. Користувачеві надається можливість самостійно задати усі обмеження та кількість рівнянь у системі, щоб кожен конкретний випадок можна було налаштувати для оптимізаційної задачі.

Розглянемо приклад, в команді існує певна кількість виконавців, які спільно об'єднують всі сім рольових кластерів. Всього є 10 виконавців. Можливі об'єднання ролей для виконавців X наведені у табл. 3.1.

Таблиця 3.1 – Варіанти несуперечливого об'єднання ролей моделі проектної групи MSF

Виконавець	Роль
X1	P1, P4
X2	P1, P3, P6
X3	P1, P2
X4	P5, P6
X5	P2, P4, P5, P7
X6	P3, P5, P7
X7	P1, P4
X8	P2, P3
X9	P2, P6
X10	P3, P7

Таким чином, можливі наступні об'єднання виконавців в рамках команд згідно з моделлю команд проектної групи MSF (табл. 3.2).

Таблиця 3.2 – Склад виконавців команд моделі MSF

№ команди	Склад виконавців одної команди
K1	X1, X6
K2	X2, X5
K3	X3, X6
K4	X4, X7, X8
K5	X7, X6, X4
K6	X8, X7, X4, X10
K7	X9, X7, X4, X10
K8	X8, X3, X4
K9	X1, X9, X10

Продовження таблиці 3.2

№ команди	Склад виконавців одної команди
К10	Х10, Х5
К11	Х10, Х7, Х8, Х4

Оскільки проект виконується за фазами (модель процесів), то обов'язковою умовою виконання певної фази є використання хоча б однієї команди. Ідеальним рішенням є використання однієї команди для виконання всього проекту.

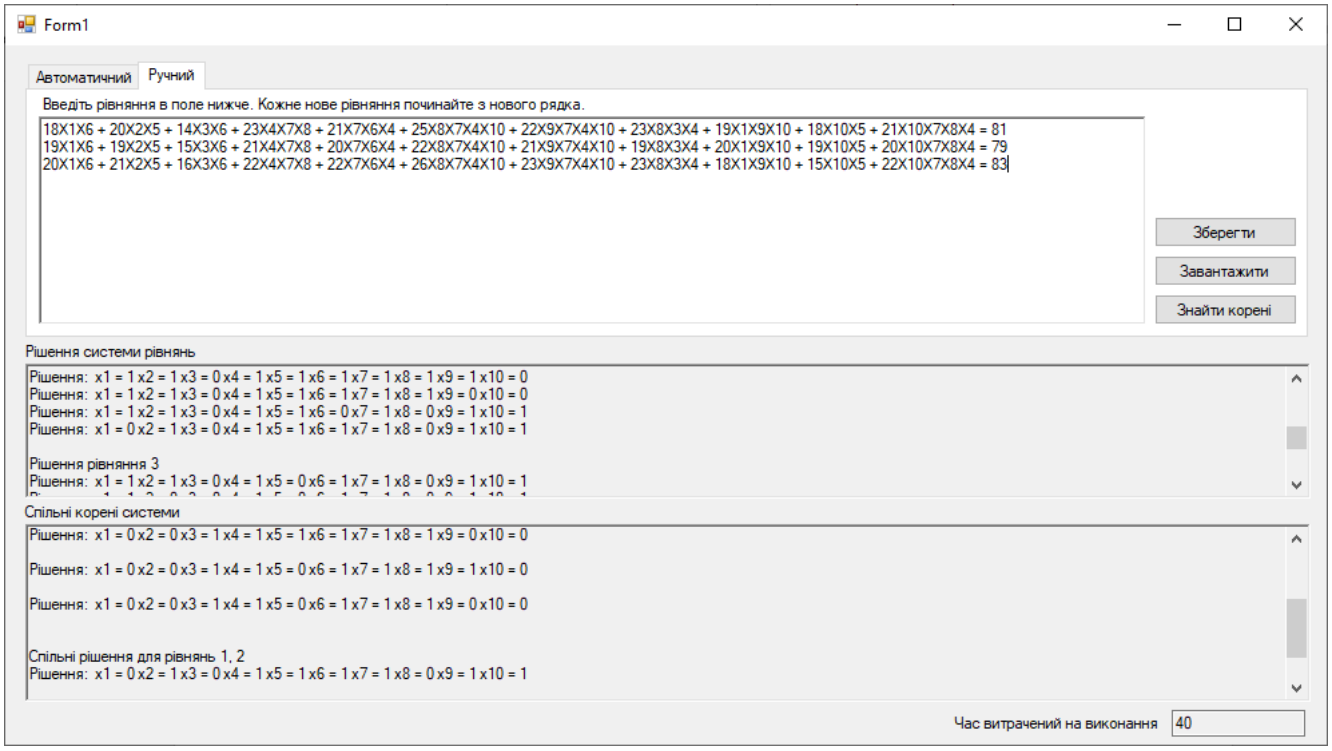

Рисунок 3.9 – Результати роботи програми за сценарієм 3

В результаті роботи програми за сценарієм користувача були знайдені спільні рішення для рівнянь 1,2 а також для рівнянь 1,3. Спільного рішення для всіх фаз одночасно знайдено не було. Це означає що спільного рішення для системи не знайдено. В такому разі потрібно проаналізувати рівняння та внести зміни до системи доки буде не знайдено спільного рішення.

За результатами розглянутих прикладів побудуємо таблицю, до якої занесемо рішення якщо вони є, за кожному з сценаріїв (табл.. 3.3).

Таблиця 3.3 – Сумісні рішення рівнянь за сценаріями

Назва сценарію	Рішення
Сценарій 1	X1, X4, X8, X9, X10, X11
Сценарій 2	Спільного рішення не знайдено
Сценарій 3	Спільного рішення не знайдено

З табл. 3.3 робимо висновок що для першого сценарію, в якому коефіцієнти створювались випадковим чином за законом нормального розподілу, а права частина рівнянь визначалися як сума останніх трьох коефіцієнтів рішення було знайдено. Для другого сценарію в якому коефіцієнти створювались таким же чином як і в першому, але права частина була створена за законом нормального розподілу, були знайдені рішення лише для деяких фаз, для усієї системи спільного рішення знайдено не було.

За сценарієм користувача, як і за другим сценарієм, були знайдені спільні рішення для деяких фаз, але для усього проекту спільного рішення знайдено не було. В такому разі користувачу потрібно проаналізувати отримані дані та зробити висновки про зміни, які необхідно внести до параметрів системи для знаходження спільного рішення.

ВИСНОВКИ

Атестаційна робота присвячена розробці оптимізаційної моделі формування проектних команд для створення програмного забезпечення. Актуальність такого підходу зумовлена тим, що при формування проектних команд існує проблема створення ефективного складу команди за умов, що команда не є однорідною, та існують обмеження на конкретні групи фахівців та весь проект.

В першому розділі було розглянуто, сучасний стан проблеми формування команд, що собою представляє проектна команда та процес її формування, які бувають види команд та їх класифікації, життєвий цикл проектних команд, а також була сформована постановка задачі.

У другому розділі, для розв’язку поставленої задачі, була побудована математична модель, що описує такий об’єкт, як команда проекту, враховуючі необхідні обмеження. Був проведений аналіз існуючих моделей формування та функціонування команд, серед яких було виділені «Задача про призначення», «Теоретико-ігрові моделі», «Експериментальні дослідницькі» та «Рефлексивні». Після аналізу моделей були сформовані фактори, які потрібно виконати під час формування команди, задля досягнення мети при обмеженні ресурсних затрат.

Розглянуто модель процесів MSF, що представляє загальну методологію розробки та впровадження ПЗ. Розглянуто підхід Microsoft до організації працюючого над проектом персоналу у вигляді рольових кластерів задля максимізації успішності виконання проекту (проектів).

Розроблена математична модель розв’язку постановленої задачі на основі нелінійного булевого рівняння та системи нелінійних булевих рівнянь, які відбивають участь певних рольових кластерів розробників ПЗ за MSF та обмеження на вартість фази проекту (або проекту загалом). Запропоновано метод розв’язку задачі, що передбачає розробку і дослідження підходів і методів з урахуванням того, що кількість проектів в організації може збільшуватися, що потребує перерахунку як кількості команд, задіяних при виконанні різних

проектів, так і складу їх виконавців для своєчасного та якісного виконання робіт усіх проектів.

Розроблено алгоритм роботи програми та програмний засіб, що знаходить найліпший склад команди для проекту. Визначено основі вимоги до програмної реалізації алгоритму рішення досліджуваної задачі. Для реалізації обрано мову програмування C# та фреймворк .NET. Користувацький інтерфейс було побудовано на технології Windows Forms, для операційної системи Windows.

Для реалізації алгоритму створена діаграма варіантів використання та схеми основних функцій. Описано принципи роботи програмного засобу та проведені дослідження з різними вхідними даними та проаналізовані результати проведених досліджень.

Розроблений програмний засіб вирішує поставлену задачу, при цьому має простий для користування інтерфейс та легкий і зрозумілий спосіб введення початкових даних та виведення результату роботи програми.

Розроблений програмний засіб може використовуватися менеджерами проектів в ІТ-сфері для знаходження оптимального складу проектних команд для розроблення ПЗ для різних методологій розроблення програмних систем.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Новиков Д. А. Математические модели формирования и функционирования команд [Текст]/ Новиков Д. А. – Изд-во: ФИЗМАТЛИТ, 2008. – 186 с.

2. Вільна енциклопедія Википедия [Електронний ресурс] / Мережева енциклопедія Wikipedia. 2000. - Forefront TMG 2010: Режим доступу: http://ru.wikipedia.org/ . - Загл. с екрана.

3. Баркер А. Как еще лучше … управлять людьми. – М.: ФАИР-Пресс, 2002.

4. Вартанян А. А. Управление командой и организацией в бизнес-среде. – М.: Доброе слово, 2006.

5. Армстронг М. Практика управления человеческими ресурсами. – СПб.: Питер, 2005.

6. Прецедентный метод формирования команды исполнителей проекта / Д.Э. Лысенко, И.В. Чумаченко, Ю.С. Выходец, В.П. Пономаренко // Системи обробки інформації. – Х.: ХУПС, 2008. – Вип. 3 (70). – С. 168-170

7. Моделирование организационного управления в многоуровневых структурах / В.Г. Кучмиев, А.И. Лысенко, В.М. Момот, И.В. Чумаченко. – Х.: НАКУ «ХАИ», 2004. – 231 с.

8. Бронштейн М. Управление командами для «чайников». – М.: Вильямс, 2004.

9. Балашов В.Г., Заложнев А.Ю., Иващенко А.А., Новиков Д.А. Механизмы управления организационными проектами. – М.: ИПУ РАН, 2003.

10. Грекул, В. И. Проектирование информационных систем[Текст] / Н. Л. Коровкина – Изд-во: ИНТУИТ, 2005. – 240с.

11. Галкина Т.П. Социология управления: от группы к команде. – М.: Финансы и статистика, 2004.

12. Нейман Д.Ф., Морнгерштерн О. Теория игр и экономическое поведение / Д.Ф. Нейман Пер. с англ. – М.: Наука, 1970.

13. Губко М.В. Математические модели оптимизации иерархических структур. – М.: ЛЕНАНД, 2006.

14. Marshak J. Elements for the theory of teams // Management Science. 1955. № 1. P. 127 – 137.

15. Мартин, Роберт К. Чистый код: создание, анализ и рефакторинг. Библиотека программиста [Текст]/ Роберт К. Мартин. – Изд-во: Вильямс, 2015. – 464 с.

16. Groves T., Radner R. The allocation of resources in a team // Journal of Economic Theory. 1972. Vol. 4. № 2. P. 415 – 441.

17. Radner R. Team decision problems // The Annals of Mathematical Statistics. 1962. Vol. 33. № 3. P. 857 – 881.

18. Новиков Д.А. Теория управления организационными системами. 2-е изд. – М.: Физматлит, 2007.

19. Листровой С.В. Подход к формированию оптимальных проектных структур на основе рангового метода решения нелинейных булевых уравнений / С. В. Листровой, С. В. Минухин // Пробл. упр. и информатики. - 2011. - № 5. - С. 110-122.

20. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание. [Текст] / Мэтью Мак-Дональд — М.: «Вильямс», 2013. — 1024 с.

21. Великий тлумачний словник сучасної української мови (з дод. і допов.) [Текст] / Уклад. і гол. ред. В. Т. Бусел. — 5-те вид. — К. ; Ірпінь : ВТФ «Перун», 2005. — ISBN 966-569-013-2.

22. Metanit.com: Сайт о программировании, про создание сайтов и IT-технологии [Електронний ресурс] / metanit.com, 2012-2014. Режим доступу: http://metanit.com/index.php. - Загл. с екрана.

23. Основы Microsoft Solution Framework / Майкл С. В. Пер. с англ. – П.: Русская Редакция, 2008.

24. Мироненко М.С. 6 Міжнародна науково-практична конференція "SCIENCE, SOCIETY, EDUCATION: TOPICAL ISSUES AND DEVELOPMENT PROSPECTS". / М.С. Мироненко Зб. Матеріалів конференції. – Харків. 2020. 283 – 288 с.

25. Teh [Электронный ресурс]. – Режим доступа: www/ URL: http://www.interface.ru/rational/teh.htm– Загл.с экрана.

26. Крачтен Ф. Введение в Rational Unified Process. /: Ф. Крачтен. Пер. с англ. – М.: Вильямс, 2002.

27. Зараменских Е.П. Управление жизненным циклом информационных систем: монография / Е.П. Зараменских. – Новосибирск: Издательство ЦРНС, 2014. – 270 с.

28. Фоппель К. Создание команды. Психологические игры и упражнения. – М.: Генезис, 2002.

29. Новиков Д.А., Цветков А.В. Механизмы стимулирования в многоэлементных организационных системах. – М.: Апостроф, 2000.

30. Мишин С.П. Оптимальные иерархии управления в экономических системах. – М.: ПМСОФТ, 2004.

31. Kocher M., Straub S., Sutter M. Individual or team decisionmaking – causes and consequences of self-selection. – Innsbruck: University of Innsbruck. Discussion Paper, 2004. – 25 p.

32. Михеев В.Н. Живой менеджмент проектов. – М.: Эксмо, 2007.

33. Мескон М., Альберт М., Хедоури Ф. Основы менеджмента. – М.: Дело, 1998.

34. Менар К. Экономика организаций. – М.: ИНФРА-М, 1996