Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Автоматизації та комп'ютерно-інтегрованих технологій*
(повна назва)

Кафедра *Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*

Розроблення компонентів системи структурно-параметричного реінжинірингу комп'ютерно-інтегрованих технологічних процесів
(тема)

Виконав:
студент *II* курсу, групи *КІТПВм-21-1*
Ханджян В. В.
(прізвище, ініціали)

Спеціальність *151 Автоматизація та комп'ютерно-інтегровані технології*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Комп'ютерно-інтегровані технологічні процеси і виробництва*
(повна назва освітньої програми)

Керівник *проф. Безкоровайний В. В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАМ _____ *Невлюдов І. Ш.*
(підпис) (прізвище, ініціали)

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет *Автоматизації та комп'ютерно-інтегрованих технологій*

Кафедра *КІТАМ*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *151 Автоматизація та комп'ютерно-інтегровані технології*
(код і повна назва)

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

Освітня програма *Комп'ютерно-інтегровані технологічні процеси і виробництва*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Ханджяну Володимиру Вадимовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення компонентів системи структурно-параметричного реінжинірингу комп'ютерно-інтегрованих технологічних процесів»*
затверджена наказом університету від *«07» листопада 2022 р. № 1464 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«07» грудня 2022 р.*
3. Вихідні дані до *роботи Об'єкт дослідження – комп'ютерно-інтегрований технологічний процес (ТП). Предмет дослідження – методи оптимізації комп'ютерно-інтегрованого ТП. Предмет розробки – засіб структурно-параметричного реінжинірингу комп'ютерно-інтегрованих ТП. Функція – підвищення ефективності комп'ютерно-інтегрованих ТП. Технічне забезпечення: ІВМ-сумісний персональний комп'ютер. Програмні засоби: ОС Microsoft Windows 10; транслятори мов програмування високого рівня; середовища розробки програм.*

4. Перелік питань, що потрібно опрацювати в роботі Вступ. Огляд і аналіз проблеми реінжинірингу ТП. Комп'ютерно-інтегровані ТП. Методи оптимізації ТП. Постановка мети та задач дослідження. Математичне забезпечення системи структурно-параметричного реінжинірингу ТП. Постановка задачі структурно-параметричного реінжинірингу. Вибір методу оцінювання ТП. Розроблення алгоритмів розв'язання задачі. Розроблення програмного забезпечення. Вибір мови програмування та засобів для розробки програмного забезпечення. Результати експериментів та їх аналіз. Охорона праці. Висновки. Перелік джерел посилання.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри): *демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) на аркушах формату А4 (12-16 сторінок).*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання, аналіз завдання, уточнення плану*		
	роботи	*01.09.22*	*Виконано*
2	*Аналіз сучасного стану проблеми реінжинірингу*		
	технологічних процесів	*08.09.22*	*Виконано*
3	*Огляд математичних моделей і методів*		
	оптимізації технологічних процесів	*15.09.22*	*Виконано*
4	*Розробка математичного забезпечення задачі*	*29.09.22*	*Виконано*
5	*Розробка програмного забезпечення задачі*	*13.10.22*	*Виконано*
6	*Проведення експериментальних досліджень*	*27.10.22*	*Виконано*
7	*Підготовка публікацій за результатами*		
	дослідження	*10.11.22*	*Виконано*
8	*Оформлення пояснювальної записки*	*24.11.22*	*Виконано*
9	*Подання закінченої роботи науковому керівникові*	*26.11.22*	*Виконано*
10	*Усунення зауважень наукового керівника*	*01.12.22*	*Виконано*
11	*Подання роботи на рецензування*	*02.12.22*	*Виконано*
12	*Підготовка презентації*	*03.12.22*	*Виконано*
13	*Попередній захист*	*05.12.22*	*Виконано*
14	*Подання роботи до екзаменаційної комісії*	*07.12.22*	*Виконано*

Дата видачі завдання *«01» вересня 2022 р.*

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка: 84 с., 4 табл., 16 рис., 1 дод., 37 джерел.

ВИРОБНИЧА СИСТЕМА, ІМІТАЦІЙНА МОДЕЛЬ, КОМП'ЮТЕРНО-ІНТЕГРОВАНИЙ ТЕХНОЛОГІЧНИЙ ПРОЦЕС, ОПТИМІЗАЦІЯ, РЕІНЖИНІРИНГ.

Об'єкт дослідження – комп'ютерно-інтегрований технологічний процес (ТП).

Предмет дослідження – методи оптимізації комп'ютерно-інтегрованого ТП.

Мета кваліфікаційної роботи – підвищення ефективності комп'ютерно-інтегрованих ТП за рахунок розробки засобу їх структурно-параметричного реінжинірингу.

Методи дослідження – теорія систем, теорія прийняття рішень, логічний та системний аналізу, імітаційне моделювання.

Результати роботи – розроблено компоненти математичного та програмного забезпечення системи структурно-параметричного реінжинірингу комп'ютерно-інтегрованих ТП. Створено веб-додаток, що двома методами виконує генерацію та моделювання різних варіантів схем побудови ТП і надає користувачеві найефективніший серед них.

Створений додаток може використовуватися на підприємствах і в організаціях, де вирішуються задачі проєктування чи реінжинірингу ТП. Практичне використання розробки дозволить без втрати точності рішень скоротити час розв'язання задачі структурно-параметричної оптимізації комп'ютерно-інтегрованих технологічних процесів.

Результати кваліфікаційної роботи апробовані на 3-х наукових конференціях.

ABSTRACT

Explanatory note: 84 pages, 4 tables, 16 figures, 1 appendices, 37 sources.

PRODUCTION SYSTEM, SIMULATION MODEL, COMPUTER-INTEGRATED TECHNOLOGICAL PROCESS, OPTIMIZATION, RE-ENGINEERING.

The object of research is a computer-integrated technological process (TP).

The subject of the research is optimization methods of computer-integrated TP.

The purpose of the qualification work is to increase the effectiveness of computer-integrated TPs due to the development of a tool for their structural and parametric reengineering.

Research methods – systems theory, decision-making theory, logical and system analysis, simulation modeling.

Results of the work – mathematical and software components of the system of structural-parametric reengineering of computer-integrated TPs were developed. A web application has been created that, using two methods, generates and simulates various variants of TP construction schemes and provides the user with the most effective one among them.

The created application can be used at enterprises and organizations where tasks of design or reengineering of TP are solved. Practical use of the development will allow to reduce the time of solving the problem of structural-parametric optimization of computer-integrated technological processes without losing the accuracy of the solutions.

The results of the qualification work were tested at 3 scientific conferences.

ЗМІСТ

ВСТУП

Ефективність виробничих технологічних процесів (ТП) багато в чому визначається рішеннями, які приймаються на етапах їхнього проектування та реінжинірингу. Синтез варіантів побудови виробничих ТП передбачає розв’язання множини задач їхньої структурної, топологічної та параметричної оптимізації за множиною функціональних і вартісних показників (собівартість продукції, її якість, завантаження устаткування, продуктивність тощо). Це обумовлює актуальність науково-прикладних задач удосконалення технологій, математичних моделей і методів їх системної оптимізації. Однією з актуальних при цьому вважається задача структурно-параметричної оптимізації під час реінжинірингу ТП.

Одним з підходів до визначення типу і необхідної кількості технологічного обладнання є багатократне імітаційне моделювання. Зазвичай при розробці технологічного процесу інженер-технолог має інформацію про верстати чи інше обладнання, які будуть використовуватися для обробки деталей та приблизну інформацію про те який об’єм продукції планується випускати. Маючи ці дані інженер-технолог може за допомогою мови GPSS створити імітаційну модель виробництва та проаналізувати його роботу. Але постає питання скільки імітаційних моделей створить інженер-технолог перш ніж отримає оптимальну? Чи буде розроблена модель найоптимальнішою? Отже, має сенс автоматизувати процес розробки імітаційних моделей та пошуку серед них найкращої. Для створення та аналізу імітаційних моделей можна скористатися високорівневими мовами програмування. Чудовою ідеєю було б розробити додаток, який зміг би в залежності від вхідних даних генерувати імітаційні моделі, результати імітації кожної моделі можна зберігати у базу даних, або ж у окремий файл для подальшого аналізу [1].

Актуальність роботи обумовлена необхідністю прискорення процесів

проектування комп’ютерно-інтегрованих ТП та вибору найкращих варіантів їхньої побудови з оцінкою за множиною показників.

Метою кваліфікаційної роботи є підвищення ефективності комп’ютерно-інтегрованих ТП за рахунок розробки засобу їх структурно-параметричного реінжинірингу.

Об’єктом дослідження є комп’ютерно-інтегрований технологічний процес.

Предметом дослідження – методи оптимізації комп’ютерно-інтегрованого ТП.

Методи дослідження – теорія систем, теорія прийняття рішень, логічний та системний аналізу, імітаційне моделювання.

Для досягнення мети кваліфікаційної роботи передбачається розв’язання множини завдань:

– виконати огляд і аналіз проблеми реінжинірингу комп’ютерно-інтегрованих ТП;

– сформулювати постановку мети та задач дослідження;

– сформулювати постановку задачі структурно-параметричного реінжинірингу ТП;

– обрати базовий метод розв’язання багатокритеріальної задачі, запропонувати його удосконалення та побудувати схеми відповідних алгоритмів;

– обрати мову програмування та середовище розробки програми;

– створити програму для розв’язання задачі;

– спланувати, провести комп’ютерні експерименти, оцінити використаних методів;

– розглянути питання безпеки життєдіяльності;

– оформити пояснювальну записку згідно з рекомендаціями [2], та вимогами ДСТУ 3008:2015 [3].

Результати дослідження доповідались на трьох наукових конференціях [4-6].

1 ОГЛЯД І АНАЛІЗ ПРОБЛЕМИ РЕІНЖИНІРИНГУ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

1.1 Комп’ютерно-інтегровані технологічні процеси

Технологічний процес – це послідовність операцій, які необхідно виконати для того щоб з вхідної сировини отримати готовий продукт. Є частиною виробничого процесу, який може мати кілька різних технологічних процесів [7].

Фаза – комплекс робіт, виконання яких характеризує завершення певної частини технологічного процесу та пов'язане з переходом предмета праці з одного якісного стану до іншого.

У машинобудуванні та приладобудуванні технологічні процеси в основному поділяються на три фази [8, 9]:

– заготівельна – процеси отримання заготовок методами лиття, штамповки, пресування, висадки, різки металу тощо;

– обробна – процеси перетворення заготовок у готові деталі шляхом: механічної обробки, штамповки, термообробки, хімічної обробки, хіміко-термічної обробки тощо;

– складальна – процеси зборки виробів із готових деталей.

Технологічний процес складається з послідовно виконуваних над даним предметом праці технологічних дій – операцій.

Операція – частина технологічного процесу, що виконується в певній фазі процесу і складається з низки дій над кожним предметом праці або групою предметів, що спільно обробляються.

Операції, що не ведуть до зміни геометричних форм, розмірів, фізико-хімічних властивостей предметів праці, не належать до технологічних операцій (транспортні, вантажно-розвантажувальні, контрольні, випробувальні,

комплектувальні та ін.).

Операції різняться також залежно від засобів праці [10]:

– ручні, що виконуються без застосування машин, механізмів та механізованого інструменту;

– машинно-ручні – виконуються за допомогою машин або ручного інструменту за безперервної участі робітника;

– машинні – виконувані на верстатах, установках, агрегатах за обмеженої участі робітника (наприклад, установка, закріплення, пуск та зупинка верстата, розкріплення та зняття деталі). Решта виконує верстат;

– автоматизовані – виконуються на автоматизованому обладнанні або автоматичних лініях.

Робота зі створення технологічних процесів у загальному випадку включає [9]:

– аналіз вихідних даних розробки технологічного процесу;

– підбір діючого типового, групового технологічного процесу чи пошук аналога одиничного процесу;

– вибір вихідної заготовки та методів її виготовлення;

– вибір технологічних баз;

– складання технологічного маршруту обробки;

– розроблення технологічних операцій;

– розробку чи уточнення послідовності переходів в операції;

– вибір засобів технологічного оснащення (ЗТО);

– визначення потреби ЗТО, замовлення нових ЗТО, у тому числі засобів контролю та випробувань;

– вибір засобів механізації та автоматизації елементів процесу та внутрішньоцехових засобів транспортування;

– призначення та розрахунок режимів обробки;

– нормування технологічного процесу;

– визначення вимог техніки безпеки;

– розрахунок економічної ефективності технологічного процесу;

– оформлення технологічних процесів.

1.2 Реінжиніринг технологічних процесів

Швидкі зміни попиту на продукцію, поява нових технологій і обладнання викликає необхідність періодичних змін виробничих технологічних процесів [1]. Реінжиніринг являє собою сукупність засобів, заходів і методів, у тому числі відповідних інформаційних технологій, призначених для кардинального поліпшення основних показників діяльності підприємства [11].

На рисунку 1.1 наведені показники, які вказують на те, що необхідно провести реінжиніринг технологічного процесу.

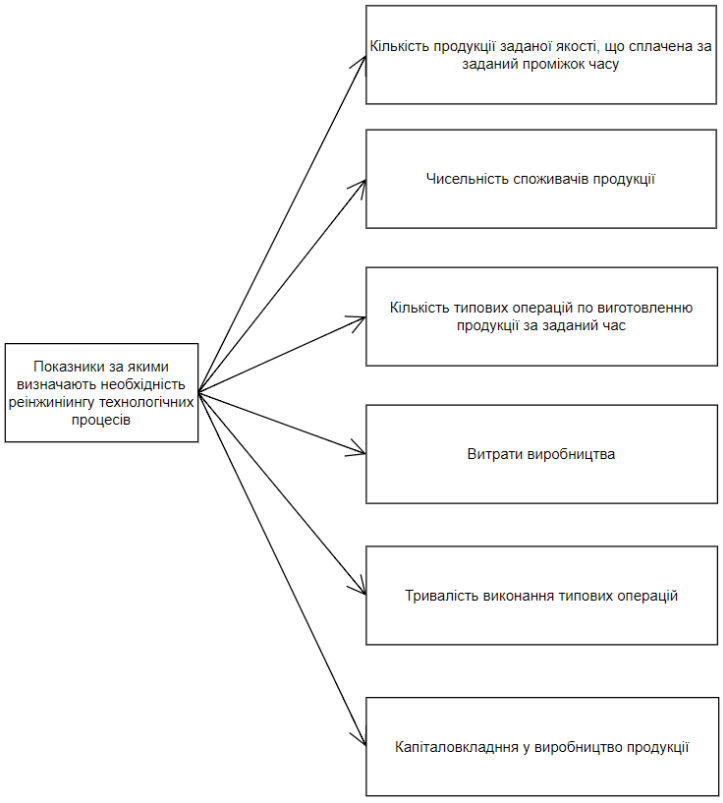

Рисунок 1.1 – Показники необхідності реінжинірингу ТП

Реінжинірингом, як і створенням плану технологічного процесу займається інженер-технолог. Як і у випадку зі створенням плану технологічного процесу він власноруч створює моделі нового технологічного процесу і проводить пошуки найоптимальнішої – це займає багато часу, до того ж обраний новий план технологічного процесу може бути не найкращим серед усіх можливих варіантів. Отже, процес реінжинірингу потребує автоматизації [12].

1.3 Огляд показників якості та ефективності технологічних процесів

Під якістю технологічного процесу слід розуміти сукупність властивостей, що визначають ступінь його придатності для досягнення заданих вимог по якості продукції, продуктивності праці, собівартості тощо. Висока якість технологічного процесу є матеріальною основою для випуску продукції високої якості.

При оцінці якості технологічних процесів необхідно виходити з надзвичайної складності їх природи, що об’єднує у собі технологічні, організаційні та економічні аспекти проблеми. Тому виникає необхідність класифікації властивостей технологічних процесів, визначення степеню їх впливу на якість технологічних процесів. Можливе використання комплексного показника для оцінки якості технологічного процесу, як узагальнюючого одиничні показники з урахуванням їх важливості [12, 13].

Видається можливим і необхідним розрізняти класифікацію властивостей технологічного процесу в цілому і класифікацію властивостей окремих технологічних операцій. Це надає можливість оцінити якість технологічного процесу і операцій з яких він складається.

Стосовно машинобудування та приладобудування використовується наступна класифікація властивостей технологічного процесу: технічні, економічні, ергономічні та естетичні і безпеки.

До технічних властивостей відносять [14]:

– точність;

– стабільність;

– надійність;

– рівень автоматизації;

– швидкодія;

– контрольованість;

– рівень виходу придатної продукції;

– патентна чистота.

До економічних властивостей відносять:

– матеріаломісткість;

– металомісткість;

– енергоємність;

– продуктивність;

– технологічна трудомісткість;

– технологічна собівартість;

– економічність.

До ергономічних та естетичних властивостей відносять:

– зручність обслуговування та управління;

– гігієнічність.

До властивостей безпеки відносять:

– рівень токсичності;

– рівень шуму;

– вибухобезпеку;

– ступінь забруднення навколишнього середовища.

Ефективність технологічного процесу оцінюють за рядом показників, що поділяють на наступні групи: технологічні, економічні та екологічні [14].

До технологічних показників відносять: потужність (продуктивність), вихід цільового продукту, селективність, степінь конверсії, витратні коефіцієнти по

сировині та енергії.

Потужність – це максимальна продуктивність апарату. Зазвичай під час проектування процесу вказується потужність апарату чи технологічної лінії. Потужність характеризує ступінь досконалості застосовуваної технології.

Продуктивність – це кількість отриманого цільового продукту чи витрати сировини за одиницю часу. Вона розраховується за формулою (1.1):

$$P = \frac{G}{t}, \tag{1.1}$$

де P – продуктивність;

G – кількість цільового продукту (сировини);

t – час.

Кількість цільового продукту визначають в одиницях маси (кг, т) чи об'єму ($м^3$). Як одиниця часу використовують годину чи добу, рідше місяць чи рік. Продуктивність відносять до цільового продукту чи основного компонента сировини.

При аналізі практичної роботи процесу застосовують показник «продуктивність» (як правило, продуктивність менша за потужність).

Вихід продукту – один із основних критеріїв ефективності технологічного процесу. Розрізняють абсолютний та відносний вихід.

Для процесів, перебіг яких може бути виражений рівнянням реакції, вихід продукту розраховують на одиницю маси.

Конверсія вихідних речовин характеризує повноту перетворення вихідних речовин у кінцевий цільовий продукт. Конверсія, як і вихід, виражається у відсотках чи частках одиниці.

Загальна конверсія – це відношення кількості однієї з вихідних речовин, що прореагувала за всіма можливими напрямами, до масової кількості тієї ж речовини,

що пройшла через реакційний апарат.

Степінь конверсії розраховують за формулою (1.2):

$$K = \frac{G_{\text{реаг. сир.}}}{G}, \tag{1.2}$$

де $G_{\text{реаг. сир.}}$ – кількість сировини, що прореагувала;

G – кількість сировини.

Для визначення ступеня корисного використання сировини розраховують корисну конверсію.

Корисна конверсія – це відношення масової кількості речовини $G^*_{\text{сир.}}$, що перетворилася на цільовий продукт, до кількості вихідної речовини G, пропущеної через реактор (1.3):

$$K^* = \frac{G^*_{\text{сир.}}}{G}. \tag{1.3}$$

У тих випадках, коли в процесі беруть участь кілька вихідних речовин, вихід і конверсію розраховують за найбільш цінним компонентом сировини.

Для складних процесів, у яких одночасно з основним процесом протікають інші процеси, що призводять до отримання побічних продуктів, вводять поняття селективності. Селективність є одним з найважливіших показників ефективності технологічних процесів. Селективність визначають за формулою (1.4):

$$S = \frac{G^*_{\text{сир.}}}{G_{\text{реаг. сир.}}}. \tag{1.4}$$

Витратні коефіцієнти – показують кількість витраченої сировини чи енергії на

одиницю виробленої продукції.

Між коефіцієнтом витрати на сировину, виходом продукту та конверсією, розрахованих на 100 мас. ч. сировини, існує така залежність (1.5):

$$b = M_{\text{с}_{bip}}\left(M_{\text{пр}} K \eta_p\right)\text{п/т}, \tag{1.5}$$

де b – витратний коефіцієнт;

$M_{\text{с}_{bip}}$ – молекулярна маса сировинного продукту;

$M_{\text{пр}}$ – молекулярна маса цільового продукту;

K – загальна степінь конверсії;

η_p – рівноважний вихід;

п/т – стехіометричні коефіцієнти реакції.

Економічні показники визначають економічну ефективність виробництва. Найважливішими серед них є собівартість продукції та продуктивність праці.

Екологічні критерії виробництва широко використовуються в промисловості для оцінки стану довкілля та досконалості технологічних процесів.

1.4 Огляд систем автоматизації створення та реінжинірингу технологічних процесів

1.4.1 Система «ВЕРТИКАЛЬ»

Система «ВЕРТИКАЛЬ» надає наступні можливості:

– проектування технологічного процесу;

– створення керуючих програм для обладнання з числовим програмним управлінням;

– технологічні розрахунки;

– формування технологічної документації у відповідності до ДСТУ та стандартів, що використовуються підприємством;

– підтримка єдиного інформаційного простору для управління життєвим циклом виробу.

Отже, система «ВЕРТИКАЛЬ» дозволяє в автоматизованому режимі проектувати технологічні процеси, в основі яких лежить ієрархічна структура з операцій, переходів, обладнання, професій, оснастки та інших технологічних об'єктів, а також надає можливість паралельного проектування складних та наскрізних техпроцесів групою технологів у реальному режимі часу.

1.4.2 Система «T-FLEX Технологія»

Система «T-FLEX Технологія» надає наступні можливості:

– використання електронних технологічних процесів як нормативних даних для планування ресурсів підприємства, оперативного планування та обліку виробництва;

– забезпечення користувачів інструментами колективної взаємодії у рамках єдиного інформаційного простору підприємства або групи підприємств з забезпеченням регламентованого доступу користувачів до різних даних;

– підтримка єдності, несуперечності і актуальності даних, що зберігаються;

– вбудовуваність в систему електронного документообігу підприємства з використанням бізнес-процесів і електронних підписів;

– підтримка тісного зв’язку з системою планування, так як технологічну підготовку підприємства потрібно планувати і керувати нею.

Отже, система «T-FLEX Технологія» дозволяє в автоматизованому режимі проектувати технологічні процеси, а також вести документацію і планувати та керувати технологічною підготовкою підприємства.

1.4.3 Система «Costimator»

«Costimator» – це американська серія програмного забезпечення для оцінки витрат, розроблена Томасом Чаркевичем у 1982 році та призначена для моделювання виробничих витрат. Програмне забезпечення спроектовано, розроблено та продається компанією MTI з Вест-Спрінгфілда, Массачусетс.

Система «Costimator» надає наступні можливості:

– оцінка з допомогою або без допомоги 3D-моделі CAD;

– наявність перевірених моделей витрат для швидкої оцінки робочого часу та витрат;

– одночасна оцінка кількох обсягів з автоматичною амортизацією витрат на налаштування та інші витрат, що залежать від кількості;

– централізована база даних, що забезпечує оновлення в режимі реального часу, а також швидкий пошук і перегляд оціночних даних по всій організації;

– інструменти імпорту, що дозволяють масово завантажувати виробничі дані, щоб швидко оновлювати котирування останньою інформацією;

– інструменти експорту, що можна легко використовувати для створення звітів користувача та посилань на інше програмне забезпечення, таке як системи ERP/MRP;

– автоматична генерація офіційної пропозиції та серії корисних управлінських звітів.

1.4.4 Система «ADEM»

Система «ADEM» призначена для автоматизації вирішення проектних, конструкторських та технологічних завдань у ґалузі машинобудування. Система використовується в наступних областях: авіаційній, атомній, аерокосмічній, машинобудівній, електро- та приладобудівній. Система ADEM також орієнтована

на проектування та виробництво складного оснащення, інструменту, штампів та прес-форм. Система може бути корисна фахівцям з технічного дизайну, деревообробки, будівництва та архітектури.

До складу «ADEM» входить модуль «ADEM CAPP». «ADEM CAPP» – це модуль проектування технологічних процесів, що призначений для автоматизації проектування одиничних, групових та типових технологічних процесів, та відомостей деталей до них за всіма технологічними операціями машинобудування та приладобудування відповідно до єдиної системи технологічної документації. Містить довідники обладнання, інструментів, матеріалів, оснащення. Дозволяє вести розрахунок трудових норм і норм витрати матеріалів.

Основні задачі, що вирішуються системою «ADEM»:

– проектування виробів;

– об’ємне та плоске моделювання;

– оформлення креслень та іншої конструкторської документації;

– проектування технологічних процесів;

– оформлення технологічної та супровідної документації;

– програмування верстатів із числовим програмним управлінням;

– управління архівами і проектами;

– оновлення накопичених знань;

– збільшене трудове нормування;

– управління довідковими даними.

1.5 Методи оптимізації технологічних процесів

Оптимізація параметрів для технологічного процесу вирішує завдання вибору методу, при якому найменші витрати на обчислення дадуть більше інформації про необхідний процес.

Процеси знаходяться у прямій залежності від того, які саме методи будуть

застосовані в роботі під час пошуку найрезультативнішого рішення для конкретної ситуації. Усього можна виділити п’ять методів, що включають:

– аналітичні, у ході застосування яких здійснюється пошук кращого варіанта серед наявних;

– програмування, ця група включає лінійні, динамічні, геометричні методи, що враховують оптимізацію через вибір найбільш результативного процесу;

– градієнтні з обмеженням або обмеження;

– автоматичні, з самоналаштуванням, які будуть оптимальними для дуже складних систем;

– статичні чи активні, які використовують різні підходи (активний пошук чи пасивне спостереження).

Оптимізація для технічних процесів застосовується для вибору оптимального варіанта з наявних, тобто фактично виконується пошук екстремуму для $F(X)$ за допомогою варіювання наявних проектних (заданих попередньо) значень для X в межах наступної області припущення (1.6):

$$extr\ F(X), X \in D_x, \tag{1.6}$$

де $F(X)$ – функція, що використовується;

X – вектор змінних;

D_x – допустима робоча область X.

Вибір буде індивідуальним, він відповідає заданим процесам та умовам. Найчастіше – це найменша собівартість, тобто найменші фінансові витрати, максимально можлива продуктивність за заданих умов з найменшим часом, що необхідний для виготовлення однієї одиниці продукції.

Методи оптимізації технологічних процесів можуть використовувати один або декілька критеріїв, тобто в роботі застосовуватимуться різні параметри, багатокритеріальна оптимізація. При цьому буде створено один компромісний

критерій, що враховує одразу кілька вибраних параметрів, так званих Е-локальних критеріїв ($E_1, E_2, E_3, \ldots E_r$). Для кожного такого критерію буде вирішуватись завдання оптимізації розробки технологічних процесів, після чого буде виконано обчислення екстремального значення для E_i (при і рівному 1, 2, 3, …, r).

Рівняння відхилення критерію буде записано таким чином (1.7):

$$Q_i = E_i - E_{i^*}. \quad (1.7)$$

Окремо для кожного з них слід обчислити ваговий коефіцієнт λ_i ($0 < \lambda_i < 1$ і $\sum \lambda_i = 1$), що необхідно для визначення важливості параметра в рамках технологічного процесу. Для запису компромісного критерію застосовується адитивна або максимінна функція. Тільки після цього вирішується оптимізація параметрів процесу. Для вирішення можуть застосовуватись різні методи, включаючи імітаційні, аналітичні, комбіновані [11].

Аналітичні методи оптимізації технологічного процесу виробництва передбачають застосування засобів математичного програмування. Усього чотирнадцять таких методів, включаючи покоординатний підйом, градієнтний спуск, виключення областей, дихотомії, поділу інтервалу, Фіббоначі, Розенбока та інші.

Імітаційна оптимізація управління технологічними процесами передбачає створення імітаційної моделі, основа якої дає можливість вибрати задовільний варіант технологічного процесу. При розрахунках застосовуються способи виключення / вибору відповідної моделі, що дозволяє досягти заданого критерію. При моделюванні застосовуються такі мови, як GPSS, Симула, Сімскрипт.

Комбінований метод передбачає використання окремих прийомів вище зазначених методів, об’єднання аналітичного та імітаційного методів в один, що дозволяє досягти оптимального результату. Такий спосіб застосовується за певних умов та необхідності отримання найбільш точного результату.

Вибір методу залежить від ситуації, умов розрахунків та інших даних, включаючи вимоги до результативності. Часто оптимальним є комбінований метод, що є більш гнучким і дозволяє працювати практично за будь-яких умов.

1.6 Постановка мети та задач дослідження

Після проведення огляду та аналізу існуючих систем структурно-параметричного реінжинірингу технологічних процесів виникли передумови для розробки більш ефективної методики генерації та оцінювання ефективності технологічних процесів з подальшим впровадження у систему реінжинірингу технологічних процесів.

На даний час жодна з існуючих систем структурно-параметричного реінжинірингу технологічних процесів не виконує оцінювання ефективності технологічного процесу за іншими показниками окрім вартісних. Отже, пропонується розробити засіб структурно-параметричного реінжинірингу технологічних процесів, що надасть користувачам можливість використовувати додаткові критерії ефективності, окрім цінових та можливість задавати коефіцієнт важливості кожного з критеріїв ефективності.

Метою кваліфікаційної роботи є підвищення ефективності комп’ютерно-інтегрованих ТП за рахунок розробки засобу їх структурно-параметричного реінжинірингу.

Для досягнення поставленої мети необхідно:

– сформулювати постановку задачі структурно-параметричного реінжинірингу ТП;

– обрати базовий метод розв’язання багатокритеріальної задачі, запропонувати його удосконалення та побудувати схеми відповідних алгоритмів;

– обрати мову програмування та середовище розробки програми;

– створити програму для розв’язання задачі;

– спланувати, провести комп’ютерні експерименти, оцінити використаних методів;

– розглянути питання безпеки життєдіяльності.

1.7 Висновки

У першому розділі кваліфікаційної роботи було проаналізовано необхідність створення системи реінжинірингу технологічних процесів. Були розглянуті існуючі аналоги розроблюваної системи структурно-параметричного реінжинірингу технологічного процесу, можна стверджувати, що жодна з розглянутих систем, не допомагає повністю автоматизувати процес реінжинірингу. Усі існуючі системи використовуються для розрахунків вартості роботи технологічного процесу та побудови плану технологічно процесу.

Розроблювана система у свою чергу зможе змоделювати усі можливі плани технологічного процесу та надати користувачу найефективніші з них за усіма можливими критеріями ефективності. Користувач зможе обрати лише ті критерії ефективності, які важливі для тих чи інших умов, а також задати важливість кожного обраного критерію. Крім того, система надаватиме множину ефективних варіантів ТП за множиною показників для остаточного вибору особи, що приймає рішення.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ СТРУКТУРНО-ПАРАМЕТРИЧНОГО РЕІНЖИНІРИНГУ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ

2.1 Постановка задачі структурно-параметричного реінжинірингу

Структурна оптимізація – це метод побудови способу з’єднання елементів у систему, яка буде максимально ефективною з точки зору наперед поставленої практичної мети і при заданих і завжди обмежених ресурсах [15].

Параметрична оптимізація – це процес знаходження значень параметрів при яких досягається максимальна ефективність виконання процедури чи забезпечуються найкращі характеристики пристрою чи системи з погляду встановленого критерію [16].

Структурно-параметрична оптимізація є комбінацією структурної та параметричної оптимізації.

У рамках кваліфікаційної роботи магістранта необхідно розробити систему реінжинірингу технологічних процесів, у результаті роботи котрої користувач отримає список найефективніших планів технологічного процесу.

Розроблювана система використовуватиметься для реінжинірингу технологічних процесів приладобудівних підприємств.

Результати виконання програми мають надаватися у діапазоні часу від 1 до 5 хвилин.

Розроблюваною системою можуть користуватися будь-які користувачі, але вона розроблюється для технологів, що відповідають за реінжиніринг технологічного процесу.

Система, що розробляється у рамках кваліфікаційної роботи повинна мати наступні можливості:

– за вхідними даними розробити схеми технологічного процесу та обрати

серед них найоптимальнішу за багатокритеріальним оцінюванням;

– зручний інтерфейс для створення власної схеми технологічного процесу та проведення імітаційного моделювання створеної схеми;

– розрахунки вартості розробленого технологічного процесу;

– формування технологічної документації у відповідності до ДСТУ;

– забезпечити користувачів інструментами колективної взаємодії у рамках єдиного інформаційного простору підприємства або групи підприємств з забезпеченням регламентованого доступу користувачів до різних даних.

У якості вхідних даних виступатимуть:

– план технологічного процесу, який потребує реінжинірингу;

– кількісна міра готової продукції, яку потрібно виготовити за один робочий день (також користувач повинен мати можливість обрати тиждень, місяць, три місяці, шість місяців та один рік);

– важливість кожного з критеріїв оцінювання технологічного процесу.

У якості вихідних даних користувач отримає звіт у форматі .pdf з інформацією про найефективніші плани технологічного процесу. Плани ранжуються за допомогою багатокритеріальної оптимізації. У якості критеріїв ранжування виступатимуть:

– кількість готових виробів за заданий період;

– середня завантаженість обладнання (зокрема, верстатів) на кожному етапі виробництва;

– середній час очікування деталей у черзі до обладнання;

– якість виробів;

– кількість бракованих виробів.

Користувач матиме змогу обирати важливість кожного з критеріїв оптимізації у процентному відношенні, загальна сума важливості не має бути більшою за 1 чи 100%.

Об’єктом дослідження є лінійний технологічний процес, що складається з n

операцій, на кожній з яких використовується m_i, $i = \overline{1,n}$ одиниць обладнання j-го типу, $j = \overline{1,j_i}$.

Обладнання кожної з операцій характеризується продуктивністю p_{ij}, $i = \overline{1,n}$, $j = \overline{1,j_i}$, якістю виконання операції q_{ij}, $i = \overline{1,n}$, $j = \overline{1,j_i}$ та наведеними витратами на його придбання, встановлення й експлуатацію c_{ij}, $i = \overline{1,n}$, $j = \overline{1,j_i}$.

Необхідно знайти найкращий варіант побудови технологічного процесу з множини допустимих $x \in X$, що визначається:

– кількістю обладнання на кожній з операцій $M = [m_i]$, $\overline{1,n}$;

– його типом $x = [x_{ij}]$, $\overline{1,n}$, $j = \overline{1,J}$ (де $J = \max\limits_{1 \le i \le n}\{j_i\}$), який з обмеженими наведеними витратами забезпечує необхідні обсяги випуску продукції й її якість [17, 18]:

$$k_1(x,M) = \sum_{i=1}^{n}\sum_{j=1}^{j_i} c_{ij} m_i x_{ij} \to \min_{x,M},\ k_1(x,M) \le k_1^*, \tag{2.1}$$

$$k_2(x,M) = \min_j\{m_i p_{ij} x_{ij}\} \to \max_{x,M}, k_2(x,M) \ge k_2^*, \tag{2.2}$$

$$k_3(x,M) = \min_j\{q_{ij} x_{ij}\} \to \max_{x,M}, k_3(x,M) \ge k_3^*, \tag{2.3}$$

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

де $k_1(x,M), k_2(x,M), k_3(x,M)$ – функції часткових критеріїв витрат, продуктивності та якості;

$x = [x_{ij}]$ – булева матриця.

Частковими випадками розглянутої задачі (2.1)-(2.4) є задачі оптимізації технологічного процесу за показниками [17, 18]:

– витрат $k_1(x,M)$ в умовах обмежень на показники продуктивності й якості;

– витрат і продуктивності $k_1(x,M), k_2(x,M)$ в умовах обмежень на показник

якості;

– витрат і якості $k_1(x, M), k_3(x, M)$ в умовах обмежень на показник продуктивності;

– продуктивності й якості $k_2(x, M), k_3(x, M)$ в умовах обмежень на витрати $k_1(x, M) \leq k_1^*$.

Отже, виходячи з вище сказаного, метою роботи є створення компонентів системи, яка за заданими вхідними параметрами знайде найефективніший план технологічного процесу, шляхом розрахунку кількості верстатів для кожної операції виробництва.

2.2 Вибір методу багатокритеріального оцінювання технологічного процесу

Для того щоб комплексно оцінити варіант ТП необхідно скористатися одним з методів для розв’язання задач багатокритеріальної оптимізації з використанням узагальненого (інтегрального) критерію оптимальності [4].

Суть методів для розв’язання задач багатокритеріальної оптимізації з використанням узагальненого (інтегрального) критерію оптимальності полягає у тому, що власні критерії $f_i(\vec{x}), i = \overline{1, n}$ будь-яким чином поєднуються в один інтегральний критерій (2.5), а потім знаходиться максимум чи мінімум цієї цільової функції [4]:

$$F(x) = \Phi\big(f_1(x), f_2(x), \dots, f_n(x)\big). \qquad (2.5)$$

Якщо об'єднання власних критеріїв будується, виходячи з об'єктного взаємозв'язку власних критеріїв і критерію узагальненого, то тоді оптимальне рішення оптимізаційної задачі буде коректне. Але таке об'єднання здійснити вкрай складно чи неможливо, тому, як правило, узагальнений критерій є результатом чисто формального об'єднання власних (часткових) критеріїв

оптимальності.

В залежності від того, яким чином власні критерії поєднуються в узагальнений критерій оптимальності розрізняють наступні види узагальнених критеріїв [19]:

– адитивний критерій;

– класичний мультиплікативний критерій (не є робочим);

– максимінний (мінімаксний) критерій.

Маючи декілька рішень, кожне з яких отримано на основі врахування багатьох критеріїв оптимізації, необхідно їх порівняти між собою, або надати перевагу одному з них. Для розв'язання цієї задачі будується функція корисності, яка дає змогу визначити показник ефективності рішення і процес надання переваги зводиться до порівняння чисел-значень.

При цьому особа, що приймає рішення (ОПР), враховує, що один набір значень локальних критеріїв володіє перевагою над іншими, якщо йому відповідає більше значення функції переваги.

2.2.1 Метод адитивного критерію

Цільову функцію будують шляхом додавання нормованих значень власних критеріїв. В загальному випадку узагальнена цільова функція має наступний вид (2.6) [19]:

$$F(\vec{x}) = \sum_{i=1}^{n} \alpha_i \, \varepsilon_i, \tag{2.6}$$

де n – кількість об'єднаних часткових критеріїв;

α_i – ваговий коефіцієнт i-го часткового критерію;

ε_i – числове значення i-го часткового критерію;

Власні критерії мають різну фізичну природу і тому різну розмірність. А значить просто підсумовувати їх некоректно. У зв'язку з цим у попередній формулі числові значення власних критеріїв поділяються на деякі дільники, що нормують і призначається в такий спосіб:

– як нормуючі дільники приймаються директивні значення чи параметри критеріїв, які задаються замовником (що можна отримати з технічного завдання). Вважається, що значення проектних параметрів, закладені в технічному завданні, є оптимальними чи найкращими;

– як нормуючі дільники приймаються максимальні (мінімальні) значення критеріїв, що досягаються в області припустимих рішень.

Розмірності власних критеріїв оптимальності та відповідних дільників, що нормують, однакові, тому в підсумку узагальнений адитивний критерій виходить безрозмірною величиною.

Алгоритм розв’язання задачі багатокритеріальної оптимізації зображено на рисунку 2.1.

Перевага даного методу полягає в тому, що, як правило, завжди вдається визначити єдиний оптимальний варіант розв’язання оптимізаційної задачі.

До недоліків цього методу можна віднести таке [19]:

– труднощі (суб'єктивізм) у визначенні вагових коефіцієнтів;

– адитивний критерій не випливає з об'єктної ролі власних критеріїв і тому виступає як формальний математичний прийом;

– в адитивному критерії відбувається взаємна компенсація власних критеріїв, тобто зменшення одного з них може бути компенсовано збільшенням іншого критерію.

Рисунок 2.1 – Схема алгоритму розв’язання задачі багатокритеріальної оптимізації з використанням узагальненого адитивного критерію

2.2.2 Метод максимінного критерію

Максімінний критерій працює за принципом компромісу, який ґрунтується на ідеї рівномірності. Сутність принципу максиміна полягає в наступному – при проектуванні складних технічних систем і наявності великого числа власних критеріїв встановити між ними аналітичний взаємозв'язок дуже складно, а в більшості випадків і неможливо, тому намагаються знайти такі значення проектних (параметрів) $\bar{\bar{x}} = \{x_1, x_2, \dots, x_m\}$, при яких нормовані значення всіх власних критеріїв рівні між собою (2.7) [19]:

$$\alpha_i f_i(X) = K, \tag{2.7}$$

де $f_i(X)$ – нормоване значення i-го критерію;

K – константа.

При великій кількості власних критеріїв через складні взаємозв'язки домогтися виконання зазначеного вище співвідношення дуже складно. Тому, на практиці, так варіюють значеннями проектних змінних проектування $x_1, x_2, \dots, x_m$, при яких послідовно «підтягуються» ті нормовані критерії, чисельні значення яких у вихідному рішенні виявилися найменшими. Оскільки ця операція виробляється в області компромісу, підтягування відстаючого критерію неминуче призводить до зниження значень частини інших критеріїв. Але при проведенні ряду кроків можна домогтися визначеного ступеня зрівноважування суперечливих власних критеріїв, що і є метою принципу максиміна.

Формально принцип максиміна формулюється в такий спосіб: вибрати такий набір змінних $X^{(0)} \in X$, при якому реалізується максимум з мінімальних нормованих значень власних критеріїв, тобто $F\left(X^{(0)}\right) = maxmin\ f_i(X)$.

Такий принцип вибору $X^{(0)}$ називають гарантований результат. Він запозичений з теорії ігор, де є основним принципом. Схема алгоритму

розв’язання ЗБО з використанням максимінного підходу наведено на рисунку 2.2.

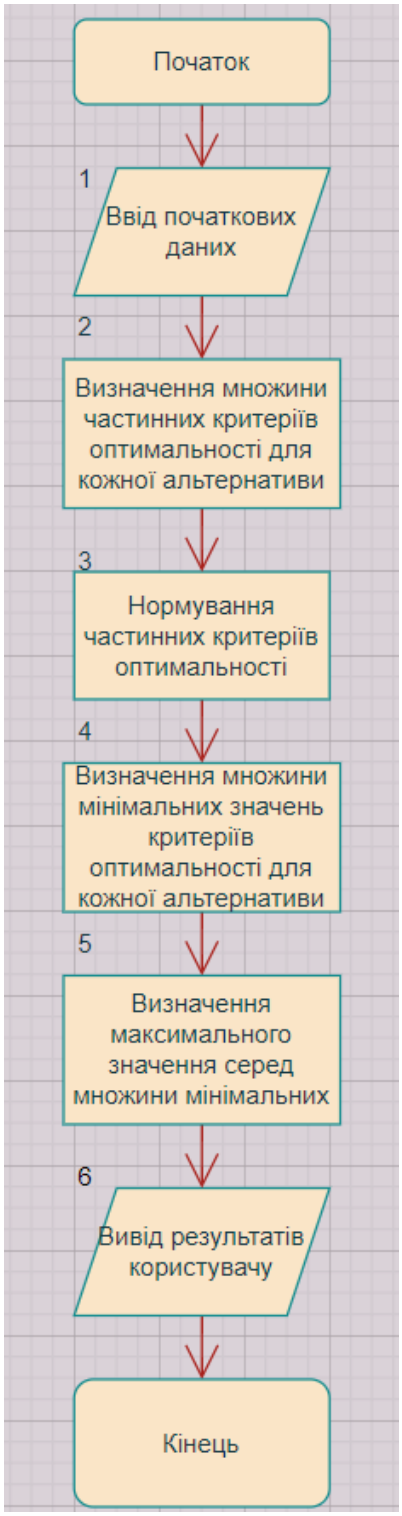

Рисунок 2.2 – Схема алгоритму розв’язання задачі багатокритеріальної оптимізації з використанням максимінного підходу

2.3 Розроблення алгоритмів

2.3.1 Алгоритм визначення завантаженості верстатів

Завантаженість верстатів – це один з найважливіших показників ефективності технологічних процесів. Якщо верстати мають невелику завантаженість – це свідчить про те, що підприємство несе збитки, через простоювання.

Для обчислення завантаженості верстатів необхідно зберегти робочу кількість часу верстату та потім розділити цей час на час робочої зміни та помножити на 100% (2.8):

$$Z = \frac{t_{\text{роб.верст.}}}{t_{\text{роб.зміна}}} \cdot 100\%, \tag{2.8}$$

де Z – завантаженість верстату;

$t_{\text{роб.верст.}}$ – час роботи верстату у поточну зміну;

$t_{\text{роб.зміна}}$ – час робочої зміни.

На рисунку 2.3 наведено схему алгоритму визначення завантаженості верстатів з позначеними етапами роботи.

Розглянемо їх більш детально:

– «1» – ввід початкових даних. Початковими даними є час робочої зміни, час, що витрачає верстат на виготовлення одного виробу (у форматі $n \pm m$) та час надходження деталей до верстату;

– «2» – цикл для моделювання роботи верстату протягом робочого дня;

– «3» – генерація часу, що витрачає верстат на виготовлення одного виробу;

– «4» – кінець циклу;

– «5» – обчислення завантаженості верстата за формулою (2.8)

– «6» – вивід результатів користувачу.

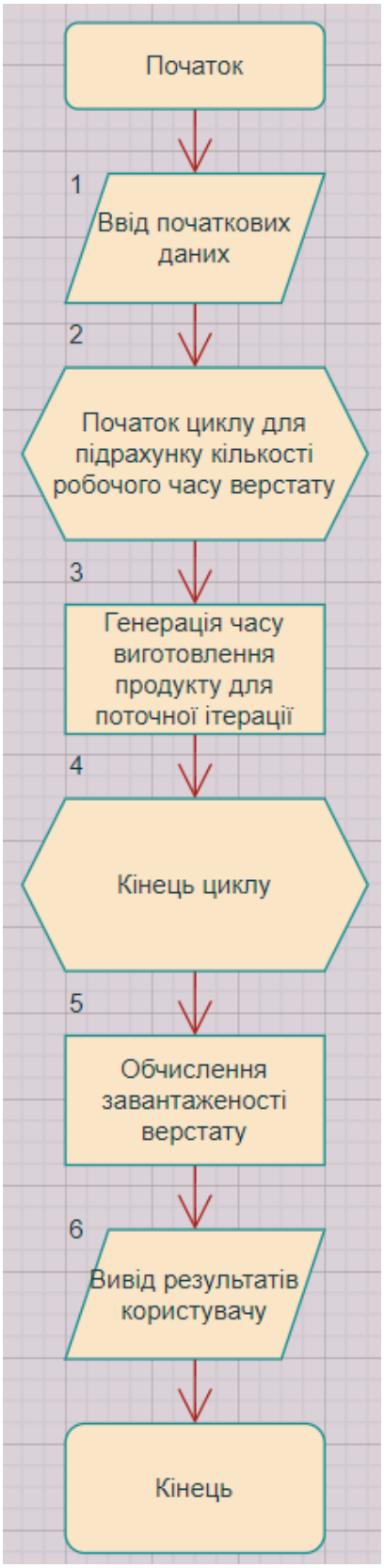

Рисунок 2.3 – Схема алгоритму визначення завантаженості верстатів

2.3.2 Алгоритм визначення обсягу виробництва

Обсяг виробництва показує кількість вироблених деталей без урахування бракованої продукції. У рамках кваліфікаційної роботи будуть проводитися розрахунки кількості оброблених виробів для кожного верстата та кількість бракованих продуктів.

На рисунку 2.4 наведено схему алгоритму визначення обсягу виробництва. Розглянемо алгоритм більш детально:

– «1» у якості початкових даних вводяться час робочої зміни, час, що витрачає верстат на виготовлення одного виробу (у форматі $n \pm m$), час надходження деталей до верстату та процент браку;

– «2» – цикл для моделювання роботи верстату протягом робочого дня;

– «3» – підрахунок кількості виготовлених деталей виконується за допомогою лічильника;

– «4» – вірогідність того, що виріб є бракованим перевіряється шляхом імітації браку за заданими процентами;

– «5» – кінець циклу;

– «6» – виведення кількості виготовлених та бракованих деталей.

Алгоритм буде використовуватися для кожного верстату, для того щоб користувача міг переглянути статистику. Як обсяг виробництва враховуватиметься сумарна кількість виготовлених виробів на останньому етапі без врахування бракованої продукції (2.9):

$$N = \sum_{i=1}^{n} (n_i - b_i), \tag{2.9}$$

де n_i – кількість деталей виготовлених поточним верстатом;

b – кількість бракованих деталей, що виготовлені поточним верстатом.

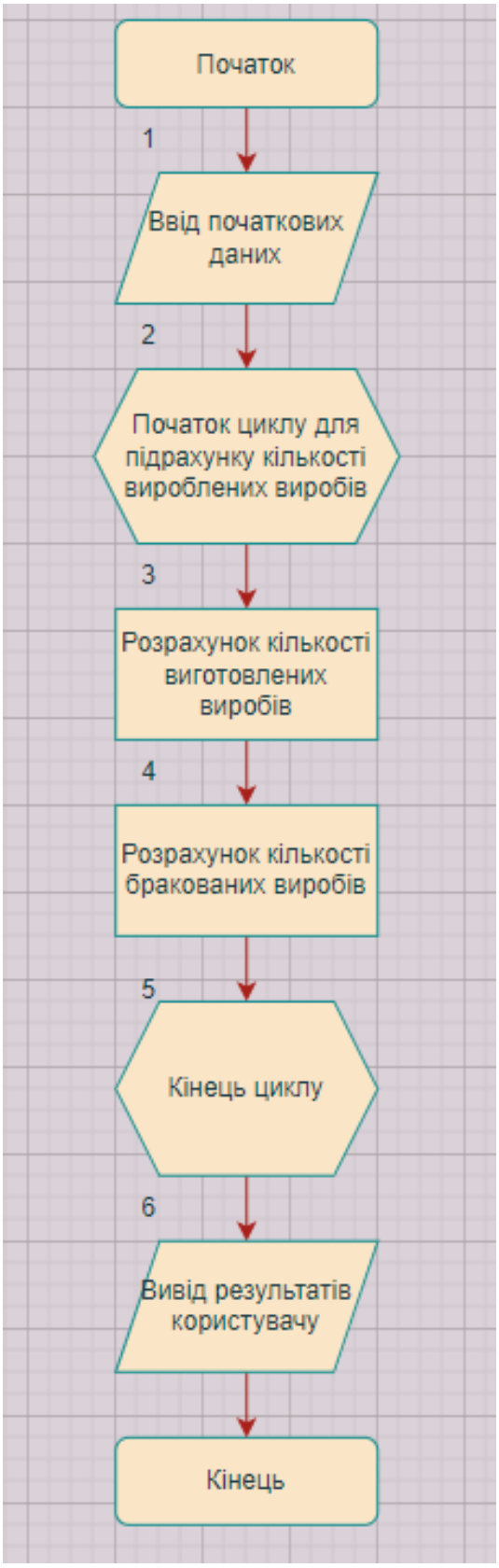

Рисунок 2.4 – Схема алгоритму визначення обсягу виробництва

2.3.3 Алгоритм генерації схем технологічного процесу

Для генерації схем технологічного процесу необхідно розробити алгоритм, з допомогою якого будуть генеруватися найефективніші схеми. Якщо генерувати схеми технологічного процесу без якоїсь стратегії – це займе занадто багато часу.

Для того, що визначити діапазон у якому можна змінювати кількість верстатів на кожному з етапів потрібна інформація про три параметри:

– кількість деталей, яку потрібно виготовити (задається користувачем) для першої фази виробництва, або ж кількість доступних матеріалів (для другого і подальших етапів);

– діапазон часу, що використовується для виготовлення однієї деталі;

– процент браку для кожного з верстатів.

Формули для розрахунку кількості виготовлених деталей одним верстатом (2.10):

$$\begin{cases} N_{min} = \dfrac{t_{\text{час зміни}}}{t_{\text{макс.час виг.}}}, \\ N_{max} = \dfrac{t_{\text{час зміни}}}{t_{\text{мін.час виг.}}}, \end{cases} \tag{2.10}$$

де N_{min} – мінімальна кількість виготовлених деталей;

N_{max} – максимальна кількість виготовлених деталей;

$t_{\text{час зміни}}$ – час робочої зміни у хвилинах;

$t_{\text{макс.час виг.}}$ – максимальний час, що витрачає верстат на виготовлення однієї деталі;

$t_{\text{мін.час виг.}}$ – мінімальний час, що витрачає верстат на виготовлення однієї деталі.

Після того, як діапазон кількостей, що виготовлені одним верстатом відомий можна розрахувати діапазон кількості необхідних верстатів, для того, щоб виготовити задану кількість деталей протягом однієї робочої зміни (2.11):

$$\begin{cases} V_{min} = \dfrac{N - N_{max\ old}}{N_{max}}, \\ V_{max} = \dfrac{N - N_{min\ old}}{N_{min}}, \end{cases} \tag{2.11}$$

де V_{min} – мінімальна кількість необхідних верстатів, при якій виконується задана норма виготовлення деталей за одну робочу зміну;

V_{max} – максимальна кількість необхідних верстатів, при якій виконується задана норма виготовлення деталей за одну робочу зміну;

N – кількість деталей, яку необхідно виготовити за робочу зміну;

$N_{max\ old}$ – максимальна кількість деталей, що виготовлена усіма старими верстатами за робочу зміну;

$N_{min\ old}$ – мінімальна кількість деталей, що виготовлена усіма старими верстатами за робочу зміну.

Такі обчислення проводитимуть для кожної фази виробництва, а самі ж схеми технологічного процесу будуть генеруватися на базі розрахованих значень, кількість верстатів для кожної фази буде вибиратися у діапазоні $[V_{min}, V_{max}]$.

Генерація схем буде відбуватися наступним чином: для першого етапу виробництва, для кожного можливого значення кількості верстатів у діапазоні $[V_{min}, V_{max}]$ будуть перебрані усі можливі значення кількості верстаті на інших етапах виробництва.

Після створення та моделювання усіх допустимих схем технологічного процесу з допомогою методів багатокритеріальної оптимізації буде проведене ранжування отриманих результатів.

2.4 Висновки до другого розділу

У другому розділі були розглянуті існуючі методи розв’язання задач багатокритеріальної оптимізації. Серед трьох розглянутих методів для подальшого

використання був обраний метод адитивного критерію, так як він має найбільші переваги є найлегшу реалізацію.

Метод мультиплікативного критерію був одразу ж відкинутий, як найменш точний, через те, що вагові критерії перемножуються, практично неможливо точно оцінити ефективність технологічних процесів. Метод максиміну був відкинути через складність програмної реалізації – не усі кроки цього методу можна коректно виконувати програмними засобами.

Також було розроблено два алгоритми, що допомагають визначити завантаженість верстатів та обсяги виробництва. Ці значення у подальшому використовуються при розв’язанні задачі багатокритеріальної оптимізації як критерії ефективності технологічних процесів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів для розробки програмного забезпечення

Історія розвитку імітаційного моделювання визначила декілька поколінь програмних засобів, еволюція яких може бути представлена у вигляді зміни шести поколінь [21, 22]:

а) перше покоління (1950-ті роки) – програмування моделей на мовах високого рівня без будь-якої спеціальної підтримки. Програми для задач програмування розроблювалися на основі універсальних мов, таких як FORTRAN та ALGOL;

б) друге покоління (1960-ті роки) – спеціальна підтримка моделювання у вигляді відповідних виразів мови, генераторів випадкових чисел, засобів представлення результатів:

1) 1960-1965 рр. – з’явилися перші мови моделювання: GPSS (мова транзактів), SIMSCRIPT (мова подій), CSL (мова робіт), SOL, GASP, SLAM;

2) 1965-1970 рр. – створено друге покоління мов моделювання: GPSS V, SIMSCRIPT II.5, SIMULA 67, GASP-IV;

в) третє покоління (1970-ті роки) – мови третього покоління мають можливість комбінованого неперервно-дискретного моделювання. Розвиток вже розроблених мов та засобів моделювання, орієнтований на підвищення ефективності процесів моделювання і перетворення моделювання у біль простий та швидкий метод дослідження складних систем. До третього покоління відносять: CADSIM, DEMOS, ACSL, MODEL-6, GEAR;

г) четверте покоління (1980-ті роки) – орієнтація на конкретні області додатка, можливість анімації. Розробка імітаційних систем, що містять інтерфейс непрограмуючого користувача, вхідні і вихідні аналізатори, можливість анімації

процесу імітаційного моделювання. Перенос програмного забезпечення для імітаційного моделювання на персональні ЕОМ з використанням засобів графічного інтерфейсу (для візуалізації і анімації процесів моделювання). До четвертого покоління відносять: SLAM II PC System Animation, PC Model SIMFACTORY, GPSS PC, XCELL;

г) п’яте покоління (1990-ті роки) – графічний інтерфейс, інтегроване середовище для створення і редагування моделей, планування експериментів, управління моделюванням і аналіз результатів. Розробка засобів технологічної підтримки процесів розподіленого моделювання на мультипроцесорних ЕОМ і мережах. До п’ятого покоління відносять: SIMPLEX II, SIMPLE++;

д) шосте покоління (кінець 1990-х років) – інтегровані системи імітаційного моделювання, у яких розвиваються найважливіші особливості засобів п’ятого покоління. До шостого покоління відносять: Arena, AutoMod, Anylogic.

3.1.1 Аналіз мов імітаційного моделювання

3.1.1.1 GPSS

Мова імітаційного моделювання GPSS є набором команд і операторів для опису об'єкта моделювання (модельованої системи). Об'єктом моделювання є будь-який об'єкт, який потрібно дослідити за допомогою моделі. Таким об'єктом може бути виробниче підприємство, транспортна система, організація торгівлі тощо [23].

На основі мови імітаційного моделювання GPSS розроблено ряд систем імітаційного моделювання. Система імітаційного моделювання – це програмний засіб, що забезпечує всі етапи роботи з моделлю: підготовку та редагування тексту моделі, налагодження моделі, моделювання, отримання результатів моделювання.

Основне призначення мови моделювання GPSS та розробленої на його основі системи GPSS World – імітаційне моделювання дискретних динамічних систем.

Під динамічною системою розуміється будь-яка система, стан якої змінюється у часі. Дискретна динамічна система – це будь-яка система, стан якої змінюється дискретно (стрибкоподібно) в дискретні (віддалені один від одного) моменти часу, коли відбуваються будь-які події. Типовий приклад дискретної динамічної системи – будь-який об'єкт, який може бути представлений у вигляді системи масового обслуговування (СМО). Стан СМО (наприклад, кількість заявок, що знаходяться на обслуговуванні або в черзі) змінюється у моменти часу, коли чергова заявка надходить на обслуговування або завершує його; ці моменти розділені інтервалами часу (випадковими чи постійними), тобто, є дискретними. Зміни станів СМО також є дискретними: наприклад, кількість заявок СМО може змінюватися тільки на цілі числа (1, 2 і т. д.) [24].

Мова імітаційного моделювання GPSS найбільше підходить для моделювання систем, які можуть бути представлені у вигляді СМО. У мові GPSS є спеціальні засоби для моделювання потоків заявок, одноканальних та багатоканальних вузлів СМО, черг тощо. Мова GPSS дозволяє моделювати практично будь-які СМО: одноканальні та багатоканальні, з необмеженими чергами, з відмовами, з обмеженнями на чергу, розімкнені та замкнуті тощо. Багато основних характеристик СМО (коефіцієнти завантаження вузлів, довжини черг і т. д.) автоматично визначаються у процесі моделювання та виводяться у складі вихідних даних моделі. У той же час за допомогою мови GPSS можуть вирішуватись завдання моделювання систем, для яких зазвичай не використовується опис у вигляді СМО [25].

Робота мови GPSS заснована на використанні методу Монте-Карло. Найчастіше операції методу Монте-Карло (звернення до генераторів випадкових чисел, перевірки умов тощо), виконуються у мові GPSS автоматично, тобто, вони приховані від користувача. Однак за потреби користувач сам має можливість реалізувати необхідні операції методу Монте-Карло [26].

3.1.1.2 SIMSCRIPT

Мова SIMSCRIPT уперше була запропонована у 1962 році співробітником компанії RAND Corporation Г. Маркевичем і зареєстрована компанією CACI Products Company. Перша версія мови була подійно-орієнтованою, в ній існували динамічні об'єкти з атрибутами та статичні об'єкти – активності. Під час взаємодії між цими елементами створювались черги об'єктів, над якими можна виконувати певні дії. Дії визначаються підпрограмами, що викликають підпрограми подій. Події, які бувають зовнішніми та внутрішніми, заносяться до списку подій. Програмний код виглядає як звичайна англійська мова.

З типової подійно-орієнтованої мови, в перших версіях, SIMSCRIPT розвинулась у інтегрований засіб розробки імітаційних моделей, у версії SIMSCRIPT II.5, керування яким виконується за допомогою системи SIMLAB, що включає універсальну мову моделювання SIMSCRIPT II.5 з упровадженими конструкціями для провесно-орієнтованого моделювання і утиліти. Версія SIMSCRIPT II.5 дає змогу створювати дискретно-подійні, неперервні та комбіновані моделі. Система SIMSCRIPT II.5 складається з таких компонентів:

– SIMLAB – середовище розробки та програмування, яке може застосовуватись для платформ Windows та UNIX;

– SIMDRAW – графічний редактор;

– COMPILER – транслятор програми Simscript II.5 у код C (це гарантує можливість використання коду для різних систем);

– LIBSIM – модуль динамічної підтримки бібліотеки під час виконання моделювання;

– LIBSIMG – графічна бібліотека, що містить графіку та анімацію;

– SIMDEBUG – діалоговий символічний налагоджувач;

– DATA GRAPH – автономний пакет, розроблений для пристосування різних функцій розподілів імовірностей для даних модельованої системи або імітаційної

моделі;

– SIMVIDEO – утиліта для фіксації і програвання отриманих результатів моделювання за допомогою анімації.

Система SIMSCRIPT широко застосовується в багатьох країнах як засіб моделювання систем передача даних, комп'ютерних мереж, транспортних і виробничих систем, військових операцій і планування логістичних процесів. Вона включає засоби опису предметної області, до якої належать пристрої, атрибути, набори (тобто черги), за допомогою яких користувач може співвідносити реальні об'єкти з моделлю. Це подійно-орієнтована мова, яка може бути перетворена в мову, схожу за формою на мову моделювання процесів. Для забезпечення графічного інтерфейсу з імітаційною моделлю SIMSCRIPT на персональному комп'ютері додатково встановлюється пакет SIMGRAPHICS.

Ці мови програмування в даний час досить широко використовуються. Проте при їх вживанні виникає проблема ідентифікації досліджуваних процесів і неминучого узгодження з вже закладеною в цих мовах структурою. Це є одній з причин, що перешкоджають активнішому використанню методів імітації при проектуванні ГВС. Інша причина такої ситуації пояснюється тим, що універсальні системи і мови моделювання володіють надмірністю засобів і методів, що викликають необхідність застосовувати потужну обчислювальну техніку і що істотно збільшують ма-шинний час обчислень. Це у свою чергу робить неможливим вживання чисельних методів математичного програмування при структурно-параметричній оптимізації ВС, оскільки один прогін моделі може тривати декілька годин. Тим не менше на разі досить широко входять у застосування програмні засоби високого рівня, які здатні повністю супроводжувати виріб упродовж його життєвого циклу. Це так звані PLM-системи (Product Lifecycle Management) – технологія керування життєвим циклом виробів.

Організаційно-технічна система, що забезпечує управління всією інформацією про виріб і пов'язаних з ним процесів протягом усього його життєвого

циклу, починаючи з проектування і виробництва до зняття з експлуатації. При цьому в якості виробів можуть розглядатися (кораблі та автомобілі, літаки і ракети, комп'ютерні мережі та ін.) Інформація про об'єкт, що міститься в PLM-системі є цифровим макетом цього об'єкта Серед таких систем найбільш широко відомі Tecnomatix від Siemens PLM Software, та Delmia від Dassault Systems [22].

3.1.1.3 GASP IV

GASP IV складається з набору підпрограм, що написані мовою FORTRAN. Підпрограми організовані так, щоб допомагати аналітикам у підготовці дискретних, неперервних або комбінованих імітаційних моделей. GASP IV формалізував підхід до підготовки таких моделей, надаючи відповідний світогляд, підкріплений підготовленими підпрограмами, які обробляють незалежну від проблеми структуру моделі. Надане бачення світу описує стан суб’єктної системи в термінах набору змінних стану та набору об’єктів із пов’язаними з ними атрибутами. Філософія моделювання GASP IV полягає в тому, що динамічне моделювання системи можна отримати шляхом моделювання подій у системі та переходу часу від однієї події до іншої [26].

Кожна імітаційна модель, що створена мовою GASP IV складається з:

– набір підпрограм, які описують правила роботи системи (підпрограми, що визначають події, умови, що викликають події, і траєкторії змінних стану);

– списки та матриці, які зберігають інформацію;

– виконавча рутина.

Набір підпрограм, що описують правила роботи, представляє технологічну логіку досліджуваної системи. Список і матриці представляють конкретні сутності, їх атрибути та відповідну керуючу інформацію. Змінні, які припадають на велику кількість програм моделювання, визначаються та надаються як змінні GASP, від користувача вимагається визначення лише залежних від проблеми змінних, які не є

GASP-змінними.

Визначення «події» GASP IV є основоположним для світогляду, який підтримує моделювання безперервних, дискретних або комбінованих систем у межах однієї концептуальної структури.

Подія – це будь-який момент часу, після якого стан системи не може бути прогнозованим з впевненістю.

Слід зазначити, що це визначення не пов’язує подію з будь-якою зміною, дискретною чи постійною, у стані системи. Такий зв’язок часто існує, але можлива подія без пов’язаної зміни стану системи. І навпаки, можлива зміна стану системи без пов’язаної події.

У GASP IV корисно описувати події з точки зору механізму, за допомогою якого вони плануються. Ті події, які відбуваються у визначений час, називаються подіями часу. Це тип події, який зазвичай розглядається у поєднанні з симуляцією «наступної події».

Події, які відбуваються, коли виконуються задані умови, визначені в умовах стану системи, називаються подіями стану. На відміну від подій часу, вони не заплановані на майбутнє. Однак вони можуть ініціювати події часу. Так само події часу можуть ініціювати події стану [26].

Виконання типової програми GASP IV починається з наданої користувачем основної програми, яка ініціює моделювання. Потім керування передається до GASP, виконавчої процедури, яка контролює моделювання до завершення.

GASP спочатку викликає підпрограму DATIN, яка ініціалізує всі змінні GASP безпосередньо або із читання картки даних. Окрім ініціалізації, DATIN також забезпечує ехо-перевірку вхідних даних.

Одразу після ініціалізації GASP готується до випередження змодельованого часу. CASP використовує комбінований метод «наступної події» та «крок оцінки кроку» для випередження часу. Цей комбінований метод необхідний через потенційне існування станів-подій, розташування яких на часовій осі невідоме.

GASP спочатку перевіряє, чи є подія часу для обробки. Якщо є, ця подія обробляється шляхом виклику підпрограми EVNTS (IX) із відповідним кодом події. Якщо ні, GASP перевіряє, чи є активний стан або рівняння похідних. Якщо їх немає, час переміщується від події часу до події часу в міру обробки кожної. Тобто відбувається як у GASP II. Якщо є активний стан або рівняння похідної, використовується інший механізм появи часу. Час просувається на максимально допустимий крок (зазначений користувачем) або до наступної події часу, залежно від того, що менше (якщо є активні похідні рівняння, це передбачає проміжні кроки та перевірку точності). У цей момент перевіряється стан системи, щоб побачити, чи сталася подія стану. Якщо у подію стану було передано більше, ніж указаний допуск, час і стан скидаються на початок кроку, і виконується спроба змінити розмір кроку. Якщо жодні події стану не були передані на межі, що перевищують вказаний допуск, обробляються всі події стану, які відбулися в межах зазначеного допуску. Якщо подія часу є запланованою, вона обробляється. Якщо подія не є запланованою, починається інший крок [26].

Після виконання заданих користувачем умов виконання програми завершується. Потім GASP викликає підпрограму SUMRY, щоб надати зведений звіт, викликає підпрограму OTPUT, щоб надати визначений користувачем вихід, перевіряє кількість запусків, що залишилися, а потім або починає новий запуск, або повертається до основної програми.

3.1.2 Вибір мови програмування для розробки інтерфейсу користувача

Для функціонування компонентів системи структурно-параметричного реінжинірингу комп’ютерно-інтегрованих технологічних процесів потрібно створити інтерфейс доступ до системи – мається на увазі створити веб/десктопний-додаток, який надасть користувачам можливість вводити вхідні дані технологічного процесу та отримувати у результаті найоптимальніший план технологічного

процесу з усіма необхідними розрахунками. Жодна з мов імітаційного моделювання, що приведені у підрозділі 2.1 не надає можливості створити інтерфейс користувача, код доводиться писати руками.

Розглянемо високорівневі мови програмування та проаналізуємо, яка з них найкраще підійде для створення інтерфейсу користувача.

3.1.2.1 Мова програмування Java

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License [27].

Мова Java проста у вивченні та містить відносно мало абстракцій у порівнянні з іншими мовами програмування. Віртуальна машина JVM служить міцною, переносимою, високопродуктивною основою для виконання програм на Java (та інших мовах). Спільно ці взаємопов’язані технології слугують міцною основою, на яку спираються комерційні організації, коли їм доводиться обирати мову на якій створюватимуться додатки.

Але на цьому переваги Java не закінчуються. З моменту створення Java було створено доволі велику екосистему сторонніх бібліотек і компонентів. Це означає, що розробники можуть отримати велику вигоду з існування поєднувачів та драйверів практично для будь-якої існуючої технології: як закритої та запатентованої так і відкритої з загальнодоступним вихідним кодом.

Насправді в сучасній технологічній екосистемі можна рідко знайти компонент технології, що не пропонує з’єднувач із Java. Будь-який тип

корпоративної системи поточного контролю, реляційні бази даних, нереляційні розподілені бази даних типу NoSQL, системи обміну повідомленнями, Інтернет речей (IoT) – все це інтегрується з Java [28].

Саме ця обставина стала головною причиною для прийняття технологій Java на підприємствах та у великих компаніях. Розробники змогли розкрити свій творчий потенціал, користуючись готовими бібліотеками та компонентами. Це спонукало розробників робити усвідомлений вибір і створювати відкриті, найбільш оптимальні архітектури на основі технологій Java.

Платформа Java стає привабливою для виконання прикладного коду завдяки поєднанню багатої екосистеми та першорозрядної віртуальної машини з відкритим стандартом для двійкового програмного коду. Насправді, крім Java, є чимало інших мов програмування, націлених на застосування JVM і взаємодію з Java, що дозволяє вигідно користуватися у них успіхами даної платформи. До них належать Kotlin, Scala, Groovy та багато інших мов. І хоча всі вони невеликі в порівнянні з Java, проте займають окремі ніші в середовищі Java, слугуючи джерелом для нововведень і складаючи здорову конкуренцію Java [29].

3.1.2.2 Мова програмування C#

Мова C# була створена корпорацією Microsoft для підтримки середовища .NET Framework і спирається на багату спадщину в ґалузі програмування. Її головним розробником був Андерс Хейльсберг (Anders Hejlsberg) – найвідоміший фахівець із програмування. C# походить безпосередньо від двох найвдаліших у ґалузі програмування мов: C та C++. Від мови C він успадкував синтаксис, багато ключових слів і операторів, а від C++ – вдосконалену об'єктну модель. Крім того, C# тісно пов'язаний з Java [30].

Маючи спільне походження, але багато в чому відрізняючись, C# і Java дуже схожі. В обох мовах підтримується розподілене програмування та застосовується

проміжний код для забезпечення безпеки та переносимості, але відмінності криються в деталях реалізації. Крім того, в обох мовах надається чимало можливостей для перевірки помилок при виконанні, забезпечення безпеки та керованого виконання, хоча й у цьому випадку відмінності у деталях реалізації. Але на відміну від Java, мова C# надає доступ до покажчиків - засобів програмування, які підтримуються в C++. Отже, C# поєднує у собі ефективність, властиву C++, і типову безпеку, характерну Java. Більше того, компроміси між ефективністю та безпекою у цій мові програмування ретельно врівноважені та абсолютно прозорі.

Незважаючи на те що в Java успішно вирішуються багато питань переносимості програм у середовищі Інтернет, її можливості все ж таки обмежені. Їй, зокрема, бракує міжмовної можливості взаємодії, яка називається також багатомовним програмуванням. Це можливість коду, написаного однією мовою, легко взаємодіяти з кодом, написаним іншою мовою. Міжмовна можливість взаємодії потрібна для побудови великих, розподілених програмних систем. Вона бажана також для створення окремих компонентів програм, оскільки найбільш цінним компонентом вважається той, який може бути використаний у різних мовах програмування і в найбільшій кількості операційних середовищ [31].

Іншою можливістю, яка відсутня в Java, є повна інтеграція з платформою Windows. Незважаючи на те, що програми на Java можуть виконуватися в середовищі Windows, за умови, що встановлена віртуальна машина Java, середовища Java і Windows не сильно пов'язані. А оскільки Windows є найпоширенішою операційною системою в усьому світі, відсутність прямої підтримки Windows є істотним недоліком Java [30].

C# входить до першої п'ятірки найпопулярніших мов програмування на 2022 рік. Він використовується в багатьох великих компаніях, а також у невеликих стартапах. Зараз компанія Microsoft наголошує на розвитку універсальності і кросспалтформенності для цієї мови. Вже зараз з його допомогою можна розробляти практично будь-який тип програм.

Компанія Microsoft залишається однією з найбільших IT компаній світу, а C# її флагманська мова програмування, яка постійно розвивається і вбирає у себе нові можливості. Тому в найближчому майбутньому проблем у цієї мови виникнути не повинно.

3.1.2.3 Мова програмування C++

C++ – це мова, що поєднує в собі функціональні можливості мови C та принципи об'єктно-орієнтованого програмування, у дев'яностих роках минулого століття зайняла лідуючу позицію серед існуючих мов програмування, і продовжує утримувати її і донині. Своїм походженням C++ завдячує мові програмування C, тому їй властиві такі характеристики, як ефективність, компактність, швидкодія та переносимість. Завдяки принципам об'єктної орієнтації, мова програмування C++ пропонує оригінальну методологію програмування, яка дозволяє вирішувати сучасні завдання програмування, ступінь складності яких невпинно зростає. Завдяки можливості використання шаблонів мова C++ пропонує ще одну нову методологію програмування: узагальнене програмування. У всьому цьому є свої позитивні та негативні сторони. Характеристики мови C++ свідчать як про те, що це дуже потужна мова програмування, але й те, що на її вивчення треба витратити досить багато часу [32].

У мові програмування C++ об'єднані три самостійні традиції програмування: традиція процедурної мови, прикладом якої є C; традиція об'єктно-орієнтованої мови, представленої класами, які у мову C++ додає мова C; традиція узагальненого програмування, що підтримується шаблонами C++.

При проектування мови C++ використовувалися наступні правила [32]:

– розроблено як універсальну мову зі статичними типами даних, ефективністю та переносимістю мови C;

– розроблено так, щоб безпосередньо та всебічно підтримувати безліч стилів

програмування (процедурне програмування, абстракцію даних, об'єктно-орієнтоване програмування та узагальнене програмування);

– розроблено так, щоб давати програмісту свободу вибору, навіть якщо це дає можливість вибирати неправильно;

– розроблено так, щоб максимально зберегти сумісність із C, тим самим уможливлюючи легкий перехід від програмування на C;

– уникає таких особливостей, які залежать від платформи чи не є універсальними;

– не накладає жодного надлишкового навантаження на програму, яка не використовує жодних можливостей;

– розроблено так, щоб не вимагати надто ускладненого середовища програмування.

Поговоримо про недоліки мови C++ [32]:

– наявність безлічі можливостей, що порушують принципи типобезпеки призводить до того, що у програмі може легко трапитися помилка, яку важко вловити. Замість контролю з боку компілятора розробники змушені дотримуватися дуже нетривіальних правил кодування. Насправді ці правила обмежують C++ рамками якоїсь більш безпечної підмови. Більшість проблем типобезпеки C++ успадковано від, але важливу роль у цьому питанні відіграє і відмова автора мови від ідеї використовувати автоматичне управління пам'яттю (наприклад, складання сміття). Так візитівкою C++ стали вразливості типу «переповнення буфера»;

– погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку файлу заголовка (#include) серйозно уповільнює компіляцію при підключенні великої кількості модулів. Для усунення цього недоліку багато компіляторів реалізують механізм прекомпіляції заголовкових файлів Precompiled Headers;

– нестача інформації про типи даних під час компіляції (CTTI);

– мова C++ є складною для вивчення та компіляції;

– деякі перетворення типів неінтуїтивні. Зокрема, операція над беззнаковим та знаковим числами видає беззнаковий результат;

– препроцесор C++ (успадкований від C) дуже примітивний. Це призводить з одного боку до того, що з його допомогою не можна (або важко) здійснювати деякі завдання метапрограмування, а з іншого, через свою примітивність, він часто призводить до помилок і вимагає багато дій щодо обходу потенційних проблем. Деякі мови програмування (наприклад, Scheme і Nemerle) мають набагато потужніші та безпечніші системи метапрограмування (також звані макросами, але мало нагадують макроси C/C++);

– з кінця XX століття у співтоваристві C++ набуло поширення так зване метапрограмування з урахуванням шаблонів. Власне, воно використовує особливості шаблонів C++ з метою реалізації з їхньої основі інтерпретатора примітивного функціонального мови програмування виконується під час компіляції. Сама по собі ця можливість дуже приваблива, але, внаслідок вищесказаного, такий код дуже складно приймати і налагоджувати. Мови Lisp/Scheme, Nemerle та деякі інші мають більш потужні та одночасно простіші для сприйняття підсистеми метапрограмування. Крім того, в мові D реалізована порівнянна потужність, але значно простіша в застосуванні підсистема шаблонного метапрограмування;

– хоча декларується, що C++ мультипарадигменна мова, реально в мові відсутня підтримка функціонального програмування. Частково цей пробіл усувається різними бібліотеками (Loki, Boost) використовуючи засоби метапрограмування для розширення мови функціональними конструкціями (наприклад, підтримкою лямбд/анонімних методів), але якість подібних рішень значно поступається якості вбудованих у функціональні мови рішень. Такі можливості функціональних мов як зіставлення із зразком взагалі дуже складно емулювати засобами метапрограмування [32].

3.1.2.4 Мова програмування Python

Python – це потужна мова програмування, що підтримує безліч парадигм програмування. Python оптимізований для забезпечення високої продуктивності програмістів, читабельності коду та якості програмного забезпечення [35].

Python є скриптовою мовою і відрізняється високим ступенем універсальності. Завдяки цьому оптимально підходить для безлічі платформ і завдань, від серверних ОС до мобільних додатків під iOS/Android [36].

Переваги мови Python [36]:

– швидкість розробки. Для створення програм на Python потрібно написати набагато менший обсяг коду, ніж у випадку з іншими популярними мовами (Java, C). Це помітно прискорює розробку, дозволяючи створювати складне програмне забезпечення швидше, ніж за допомогою інших мов програмування;

– логічний синтаксис. Логічний синтаксис цієї мови спрощує читання та розуміння її коду, завдяки чому її досить легко освоїти. Python по праву вважається однією з найбільш простих мов програмування для початківців;

– різноманітність бібліотек. Крім стандартної бібліотеки, для Python є великий вибір додаткових бібліотек. Серед найпопулярніших з них варто відзначити SQLAlchemy (для роботи з базами даних), Pygame (для розробки мультимедійних додатків та ігор), Flask та Django (для розробки серверної частини ПЗ), NumPy (для розробки у сфері машинного навчання та штучного інтелекту), Pandas (для обробки big data) тощо;

– масштабованість. Написані на Python програми та додатки легко розширюються та масштабуються завдяки можливості адаптації їх високорівневої логіки;

– універсальність. Python – це інтерпретована мова, яка використовується для кодингу практично на всіх сучасних платформах. Вона не потребує компіляції та її код можна писати у звичайному текстовому документі;

– світова спільнота. Одним з важливих факторів бурхливої популяризації Python вважається численна спільнота розробників та ентузіастів цієї мови. Його розвиток ведеться на базі регулярно оновлюваної та чітко регламентованої документації PEP (пропозицій щодо розвитку Python).

Недоліки мови Python [36]:

– низька швидкість виконання. Python використовує динамічну типізацію та автоматично керує пам'яттю. У невеликих проектах це доповнює переваги Python. Але у складних та масштабних проектах, де робиться акцент на ефективність та продуктивність, Python програє своїм «побратимам»: C, C++, Java, Go. Але в той же час за швидкістю він обходить PHP, Ruby та JavaScript.

– проблеми динамічної типізації. Як уже говорилося у першому пункті, у невеликих проектах це грає «на руку». І здається, що наявність «динамічної типізації» – це справжній «рай». Але як тільки проект починає «рости», код стає все ширшим – це починає грати злий жарт. Стає просто неможливим встежити за типами даних, що передаються, і це перетворюється на велику проблему. Вони, звичайно, вирішуються всякими доробками та підкручуваннями, але це виглядає так собі. Плюс від цього падає швидкість мови (а вона й так низька).

3.1.2.5 Вибір мови програмування

Серед описаних вище мов програмування більш за все для вирішення поставленої задачі (розробка компонентів системи структурно-параметричного реінжинірингу комп’ютерно-інтегрованих технологічних процесів) підходить мова програмування Java.

Мова Java об’єднує у собі усі переваги вищенаведених мов (C++, C#, Python) та майже не має недоліків.

Java проста й зручна у використанні, має добре пророблену документацію, включає у себе великий набір інструментів розробки, володіє високим рівнем

безпеки та надійності, підтримує багатопоточність та зворотню сумісність, є кросплатформовою мовою, швидкість виконання коду значно більша порівняно з іншими мовами програмування.

3.1.3 Дослідження можливості взаємодії мов імітаційного моделювання з мовою Java

Для структурно-параметричного реінжинірингу комп'ютерно-інтегрованих технологічних процесів планується використовувати мову імітаційного моделювання.

Спочатку за вхідними даними мова Java генеруватиме код на мові імітаційного моделювання, згенерований код оброблюватиметься середою імітаційного моделювання та у якості звіту відправлятиметься до Java, після чого дані будуть передаватися користувачу.

Представлений вище варіант є найкращим рішенням, але може статися так, що зв'язок мов імітаційного моделювання з Java неможливий, тоді доведеться виконувати імітаційне моделювання на стороні Java. Але спочатку проведемо практичне дослідження можливості взаємодії мови Java з GPSS, SIMSCRIPT та GASP IV.

3.1.3.1 Дослідження взаємодії мов Java та GPSS

Розглянемо генерацію GPSS-моделі та її симуляцію. На перший погляд все доволі легко, достатньо просто створити файл з розширенням .gps, записати у нього код, відкрити у GPSSW Editor та запустити симуляцію.

На рисунку 3.1 приведений стандартний спосіб створення та запису даних у файл мовою Java.

```java
public void generateGPSSFile(String fileName) throws IOException {
    File file = new File(fileName);
    // Створення нового файлу
    file.createNewFile();

    // Створення екземпляру класа FileWriter, що надає можливість записувати дані у файл
    FileWriter fileWriter = new FileWriter(file, Boolean.FALSE);
    // Запис даних у файл
    fileWriter.write( str: "GENERATE 15,3\n" +
            "QUEUE 1\n" +
            "SEIZE CHANNEL\n" +
            "DEPART 1\n" +
            "ADVANCE 8,2\n" +
            "RELEASE CHANNEL\n" +
            "TERMINATE 1 \n" +
            "START 200");
    // Гарантує, що дані записані у файл
    fileWriter.flush();
}
```

Рисунок 3.1 – Стандартний спосіб створення та запису даних у файл

Після компіляції Java-коду у папці resources буде створений файл imitation_model.gps, але при спробі відкрити цей файл за допомогою GPSSW Editor буде показано помилку, що приведена на рисунку 3.2.

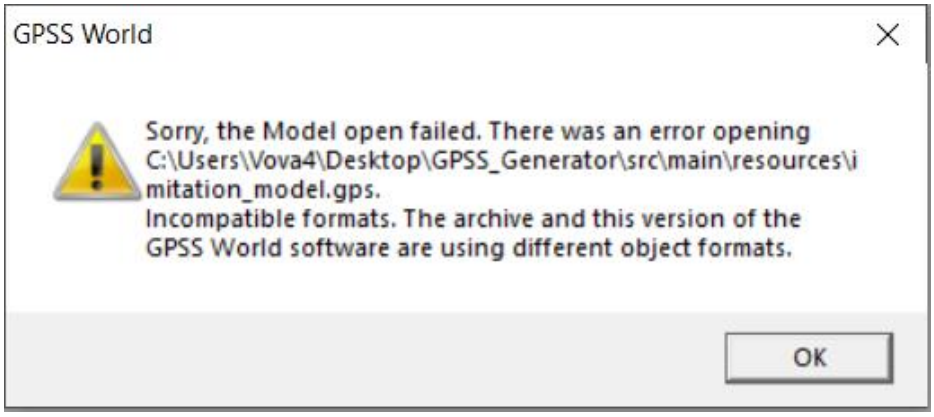

Рисунок 3.2 – Помилка при спробі відкрити згенерований Java-кодом .gps файл

Було проведено пошук інформації і як результат було з’ясовано, що GPSSW Editor може відкривати лише ті моделі, що створені ним самим. Створення .gps

моделей за допомогою інших програм неможливе [18].

Отже для того щоб створити GPSS-модель доведеться відкрити GPSSW Editor за допомогою Java, вставити GPSS-код, зберегти модель, запустити симуляцію та зберегти отриманий звіт.

Для відкриття GPSSW Editor можна скористатися командою:

```
Desktop.getDesktop().open(new File(PATH_TO_GPSSW));
```

де *PATH_TO_GPSSW* – це константа, яка вказує шлях до GPSSW Editora.

Далі за допомогою Java-коду симулюємо натискання гарячих клавіш для створення та збереження моделі. Симуляція натискання клавіш виконується за допомогою методів бібліотеки java.awt, а саме за допомогою методі класу Robot. Копіювання GPSS-коду перед тим як вставити його у GPSSW Editor виконується за допомогою наступного коду:

```
Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
clipboard.setContents(new StringSelection(copied), null);
```

де `copied` – це рядок, що потрібно скопіювати.

Команди для натискання клавіш мають наступний вигляд:

```
Robot bot = new Robot();
bot.keyPress(KeyEvent.VK_ENTER);
bot.keyRelease(KeyEvent.VK_ENTER);
```

Перша команда – це ініціалізація екземпляру класу Robot. Наступні дві команди відповідають за натиск та відпущення відповідної клавіші.

Отже, увесь алгоритм створення GPSS-моделі та її симуляції реалізується за допомогою натискання гарячих клавіш, яке імітується за допомогою Java-коду.

3.1.3.2 Дослідження взаємодії мов Java та SIMSCRIPT

У випадку з SIMSCRIPT все значно легше ніж з GPSS, але доведеться писати власний сервер мовою typescript, так як бібліотеки для роботи з SIMSCIPT існують у мові typescript та недоступні для Java.

Постає питання, чому не розробляти інтерфейс користувача та обчислювальну складову системи одразу ж мовою typescript – насамперед typescript призначений тільки для розробки інтерфейсу користувача, будь-які обчислення виконані цією мовою можуть бути не точними, самі ж обчислення займають набагато більше часу ніж подібні розрахунки, що виконуються мовами високого рівня.

На рисунку 3.3 наведено діаграму, що демонструє процес пошуку оптимального технологічного процесу з використанням мов java та typescript.

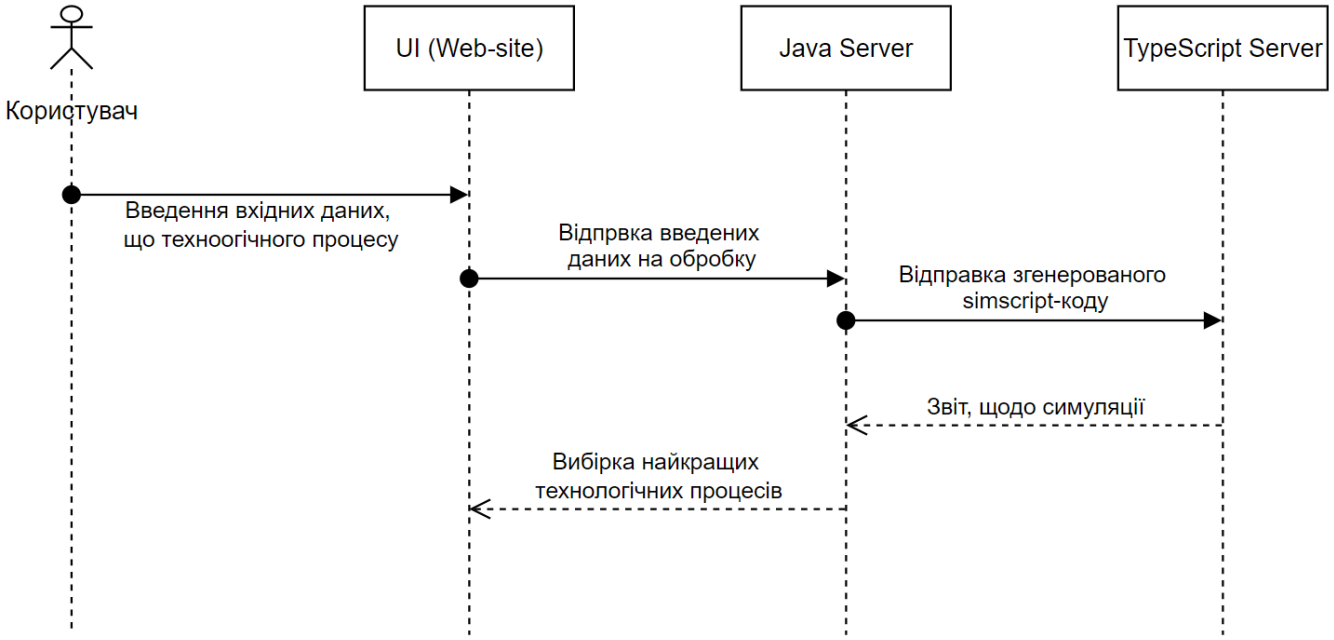

Рисунок 3.3 – Діаграма послідовності взаємодії користувача з розроблюваною системою

Користувач надсилає вхідні дані за допомогою UI до серверу Java, який в свою чергу генерує усі можливі імітаційні моделі (код мовою SIMSCRIPT). На наступному кроці Java-сервер передає згенерований код до TypeScript-сервера, котрий у свою чергу проводить моделювання і повертає результати до Java-сервера у форматі JSON. Результати кожного моделювання зберігаються Java-сервером та після останнього моделювання серед усіх отриманих результатів формується вибірка з найкращих технологічних процесів, що потім відправляється до користувача.

Такий підхід набагато легший ніж у випадку з GPSS, де необхідно за допомогою гарячих клавіш відкривати GPSS вставляти код, зберігати його та виконувати симуляцію. У випадку з GPSS вдасться створити лише десктопний додаток, який працюватиме лише за наявності GPSS World та ще й блокуватиме комп’ютер на дуже великий час, так як поки йде симуляції неможливо виконувати жодні дії. У випадку з SIMSCRIPT можна розробити веб-додаток, який одночасно зможе користуватися велика кількість людей. Також обчислення виконуватимуться набагато швидше через існування бібліотек для SIMSCRIPT.

3.1.3.3 Дослідження взаємодії мов Java та GASP IV

Після проведення досліджень було виявлено, що інтеграція мов Java та GASP IV неможлива ні яким з вище наведених способів. Не існує ніяких бібліотек, що могли б встановити зв’язок мов GASP IV та Java, більш того мову GASP IV неможливо інтегрувати з будь-якою з існуючих мов програмування, вона на це просто не розрахована. Варіант з гарячими клавішами, як у випадку з GPSS, те ж відпадає.

Виходячи з усього вище сказаного можна сміливо стверджувати, що створити додаток для створення / реінжинірингу технологічного процесу з використанням мови GASP IV фізично не можливо.

3.1.3.4 Висновки, щодо вибору способу реалізації додатку

Серед трьох розглянутих вище мов імітаційного GASP IV вибуває так як її інтеграція з мовами програмування технічно неможлива через відсутність бібліотек та інших способів інтеграції.

Інтеграція з GPSS теж не є оптимальним варіантом, так як доводиться спиратися на гарячі клавіші для створення та симуляції імітаційної моделі. Для того щоб варіант додатку працював з GPSS потрібно буде встановлювати програму GPSS World та вказувати у додатку шлях до GPSS World, також симуляції буде виконуватися дуже повільно й неможливо буде виконувати ніякі інші дії з комп'ютером під час симуляції.

Однією з проблем є те, що отримані звіти доведеться читати мовою Java та аналізувати – для цього доведеться створити спеціальний парсер та аналізатор.

Найоптимальнішим варіантом є інтеграція з мовою SIMSCRIPT – для цієї мови існує спеціальна бібліотека, що написана мовою Javascript. Для взаємодії доведеться створити сервер мовою JavaScript, що отримуватиме SIMSCRIPT код, та у відповідь повертатиме результати симуляції. Цей підхід є найкращим з точки зору легкості реалізації та основних принципів програмування.

Існує четвертий спосіб створення додатку, при якому непотрібна жодна з мов імітаційного моделювання. Достатньо створити мовою Java додаток, що буде виконувати ті ж дії, що й мови імітаційного моделювання. Більшість мов імітаційного моделювання написані на високорівневих мовах програмування, Java як раз і є такою мовою.

І нарешті п'ятий спосіб – додаток, що повністю написаний на мові Javascript з використанням React. Такий додаток буде значно точнішим за усі інші та легшим у реалізації.

Отже, з усіх існуючих способів для створення додатку можуть використовуватися або ж зв'язка Java – Javascript – SIMSCRIPT, Javascript, або ж

тільки мова Java, але постає питання, який з варіантів простіший та працює швидше.

3.1.4 Дослідження роботи програми з використання Java, React, Java + React

3.1.4.1 Вхідні дані

Кожні 15 ± 3 хвилин зі складу надходять матеріали для обробки, матеріали проходять чотири рівні обробки:

– перший рівень обробки – оператор шукає браковані матеріали (5 ± 1) хвилин;

– другий рівень обробки – оператор звантажує матеріали до термопласт-автомату, термопласт-автомат обробляє деталі 20 ± 5 хвилин;

– третій рівень обробки – пакувальник шукає браковані деталі серед виготовлених (5 ± 1 хвилин);

– четвертий рівень обробки – пакувальник складає деталі у коробку та відправляє на склад (7 ± 2 хвилин).

Процес роботи триває 12 годин.

3.1.4.2 Дослідження роботи програми з використанням Java

На основі вказаних вище вхідних даних мовою Java був створений консольний додаток, що займається імітаційним моделюванням роботи фабрики.

Програма є універсальною, на її створення було витрачено 1 день.

На рисунку 3.4 наведено код утилітарної функції, яка використовується для розрахунку кількості виготовлених деталей на заданому етапі виробництва.

```java
/**
 * This method imitate simulation level
 *
 * @param workTime - work time
 * @param minProcessTime - min time a diapason
 * @param maxProcessTime - max time a diapason
 * @param maxProcessedDetails - details processed on previous step
 * @return number of delivered materials and time which spent on first delivery
 */
public static int[] processingLevel(int workTime, int minProcessTime,
                                    int maxProcessTime, int maxProcessedDetails) {
    int currentTime = workTime;
    int numberOfProcessingItems = 0;
    int[] statistic = new int[2];
    while (true) {
        int processingTime = TimeGeneratorUtils.generateTimeInDiapason(minProcessTime, maxProcessTime);

        if (currentTime - processingTime < 0
                || (maxProcessedDetails != 0 && maxProcessedDetails == numberOfProcessingItems)) {
            statistic[1] = numberOfProcessingItems;
            return statistic;
        }

        currentTime -= processingTime;
        numberOfProcessingItems++;

        if (statistic[0] == 0) {
            statistic[0] = processingTime;
        }
    }
}
```

Рисунок 3.4 – Утилітарна функція, що відповідає за розрахунок кількості виготовлених деталей

Функція приймає у себе чотири змінні:

– workTime – загальний час роботи на поточному етапі;

– minProcessTime – мінімальний час, що потрібний для завершення однієї ітерації;

– maxProcessTime – максимальний час, що потрібний для завершення однієї ітерації;

– maxProcessedDetails – максимальна кількість деталей, що доступна для обробки (для першого етапу потрібно задати 0).

Функцію у подальшому можна буде розширити, та додати нові дані до статистики, наприклад, навантаженість та кількість бракованих деталей.

Оцінимо швидкість роботи програми, для цього запустимо її десять разів, результати експериментів наведено у таблиці 3.1.

Таблиця 3.1 – Дослідження швидкодії імітаційного моделювання з використання мови програмування Java

Номер експерименту	Час роботи програми, мс
1	55
2	63
3	52
4	58
5	63
6	87
7	60
8	73
9	86
10	62
Середній час	65,9

3.1.4.3 Дослідження роботи програми з використанням Java + React

Розробка програми виявилася дуже складною і зайняла 3 дні. Для розробки програми були використані фреймворки React (Javascript) та Spring (Java). Нажаль бібліотеку simscript можливо використовувати лише з фреймворком React, він не працює ні з чистим Javascript ні з Node JS. Також розроблена програма не є універсальною (потрібна довга доробка, так як бібліотека simscript дуже складна у засвоєнні).

Програма працює наступним чином, користувач натискає кнопку

«SIMULATE FACTORY» (рисунок 3.5), після чого React виконує симуляцію та відправляє дані до Java, далі Java обробляє дані та повертає відповідь. На даному етапі ніякої обробки на стороні Java поки що немає, вона буде додана у майбутньому коли потрібно буде зберігати результати імітації різних схем технологічного процесу і шукати серед усіх схем найкращі за певними критеріями. Така операції некоректно проводити фронтенд мовами, якою є React, бо значно постраждає точність обчислень, на офіційних проектах, будь-які розрахунки проводяться лише мовами високого рівня, такими як Java.

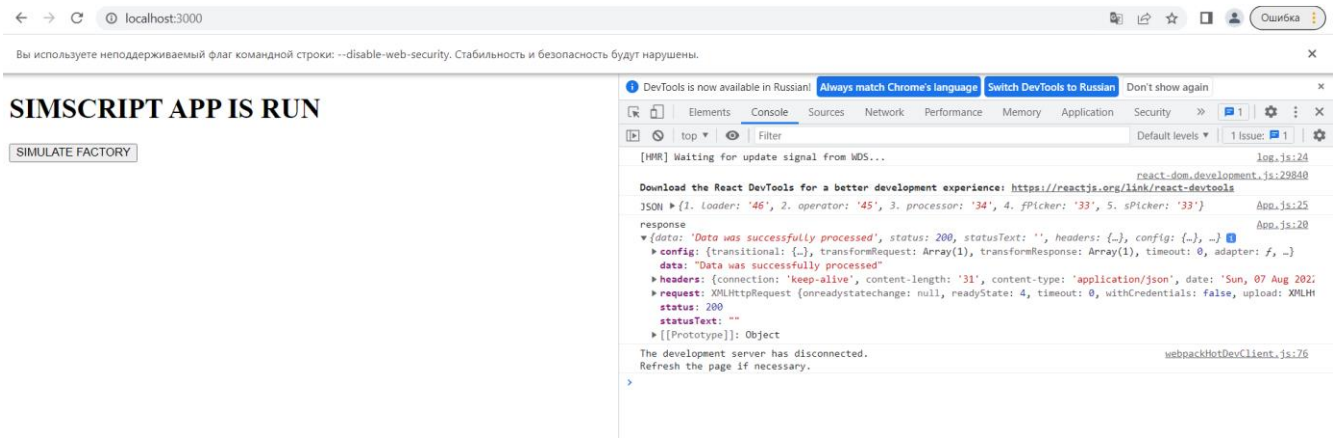

Рисунок 3.5 – UI для симуляції з використанням Java + SIMSCRIPT

Розглянемо код створеної програми. На рисунках 3.6-3.7 наведено код react-класів, щ відповідають за виконання симуляції.

У класі Factory виконується ініціалізація черг до етапів виробництва. Їх потрібно задавати у явному вигляді, динамічне завдання кількості черг неможливе. Також у цьому класу задається час симуляції.

У класі SimulateFactory виконується власне симуляції. Виконуються кроки генерації деталей, постановка їх у чергу, обробка т перехід до наступного етапу. Усі ці дії теж не можуть генеруватися динамічно.

```
class Factory extends Simulation {
    qLoader = new Queue( name: 'Loader', capacity: 1);
    qOperator = new Queue( name: 'Operator', capacity: 1);
    qProcessor = new Queue( name: 'Processor', capacity: 1);
    qPicker1 = new Queue( name: 'Picker #1', capacity: 1);
    qPicker2 = new Queue( name: 'Picker #2', capacity: 1);
    qWait = new Queue( name: 'Wait Area');

    onStarting() {
        super.onStarting();
        this.timeEnd = 720
        this.generateEntities(SimulateFactory, new Uniform( min: 15 - 3, max: 15 + 3));
    }
}
```

Рисунок 3.6 – Клас Factory

```
46      class SimulateFactory extends Entity<Factory> {
47          service = new Uniform( min: 15 - 3, max: 15 + 3);
48
49          async script() {
50              const factory = this.simulation;
51
52              await this.enterQueue(factory.qWait);
53              await this.enterQueue(factory.qLoader);
54              this.leaveQueue(factory.qWait);
55              await this.delay(this.service.sample());
56              this.leaveQueue(factory.qLoader);
57
58              this.service = new Uniform( min: 5 - 1, max: 5 + 1);
59
61              await this.enterQueue(factory.qOperator);
62              this.leaveQueue(factory.qWait);
63              await this.delay(this.service.sample());
64              this.leaveQueue(factory.qOperator);
65
66              this.service = new Uniform( min: 20 - 5, max: 20 + 5);
67
68              await this.enterQueue(factory.qWait);
69              await this.enterQueue(factory.qProcessor);
70              this.leaveQueue(factory.qWait);
71              await this.delay(this.service.sample());
72              this.leaveQueue(factory.qProcessor);
73
74              this.service = new Uniform( min: 5 - 1, max: 5 + 1);
```

Рисунок 3.7 – Клас SimulateFactory, аркуш 1

```
75
76              await this.enterQueue(factory.qWait);
77              await this.enterQueue(factory.qPicker1);
78              this.leaveQueue(factory.qWait);
79              await this.delay(this.service.sample());
80              this.leaveQueue(factory.qPicker1);
81
82              this.service = new Uniform( min: 7 - 2,  max: 7 + 2);
83
84              await this.enterQueue(factory.qWait);
85              await this.enterQueue(factory.qPicker2);
86              this.leaveQueue(factory.qWait);
87              await this.delay(this.service.sample());
88              this.leaveQueue(factory.qPicker2);
89          }
90      }
```

Рисунок 3.7, аркуш 2

Результати виконання програми формуються у відповідь у форматі JSON та передаються до Java-Web додатку. На рисунку 3.8 наведено код, що виконує формування та відправку відповіді у форматі JSON.

```
const listener = () => {
    console.time( label: 'Simulation');
    const sim = new Factory();
    sim.start();

    sim.stateChanged.addEventListener( listener: () => {
        const JSON_RSP = {
            '1. loader': `${format(sim.qLoader.grossDwell.cnt,  decimals: 0)}`,
            '2. operator': `${format(sim.qOperator.grossDwell.cnt,  decimals: 0)}`,
            '3. processor': `${format(sim.qProcessor.grossDwell.cnt,  decimals: 0)}`,
            '4. fPicker': `${format(sim.qPicker1.grossDwell.cnt,  decimals: 0)}`,
            '5. sPicker': `${format(sim.qPicker2.grossDwell.cnt,  decimals: 0)}`
        }
        axios.post( url: 'http://localhost:8080/factory/simulation', JSON_RSP)
            .then(response => {
                console.log('response', response);
            }).catch(error => {
                console.err('error', error)
            });

        console.log('JSON', JSON_RSP)
        console.timeEnd( label: 'Simulation');
    })
}
```

Рисунок 3.8 – Метод, що вмикає симуляцію та надсилає результати у форматі JSON до Java-Web додатку

На стороні Java-Web додатку створено простий сервлет, що відправляє речення «Data was successfully processed» як відповідь до react-додатку (рисунок 3.9).

```
@RestController
public class DataController {

    @PostMapping("/factory/simulation")
    public String index(@RequestBody String JSON) {
        System.out.println("JSON: " + JSON);
        return "Data was successfully processed";
    }
}
```

Рисунок 3.9 – Java-сервлет, що приймає JSON з результатами симуляції

Оцінимо швидкість роботи програми, для цього запустимо її десять разів, результати експериментів наведено у таблиці 3.2.

Таблиця 3.2 – Дослідження швидкодії імітаційного моделювання з використання мови програмування Java + React

Номер експерименту	Час роботи програми, мс
1	46,28
2	41,05
3	43,38
4	44,96
5	40,4
6	66,73
7	37,76
8	35,41
9	35,27
10	35,04
Середній час	42,63

3.1.4.4 Дослідження роботи програми з використанням Javascript

Для розробки програми з використанням Javascript скористаємося тим самим алгоритмом, що й для програми написаної мовою Java. Програму з використанням алгоритмів Java + React не має сенсу створювати, так як вона не є універсальною і має створюватися під заданий технологічний процес, що суперечить меті роботи.

Оцінимо швидкість роботи програми, для цього запустимо її десять разів, результати експериментів наведено у таблиці 3.3.

Таблиця 3.3 – Дослідження швидкодії імітаційного моделювання з використання мови програмування Javascript

Номер експерименту	Час роботи програми, мс
1	2,27
2	2,08
3	2,26
4	3,11
5	2,78
6	1,58
7	3,78
8	2,28
9	1,45
10	3,01
Середній час	2,46

3.1.4.5 Висновки, щодо проведених експериментів

У результаті серії проведених результатів стало зрозуміло, що хоча реалізація використовуючи Java + React швидша за Java, але вона майже у 20 разів повільніша

за програму написану мовою Javascript (з використанням Framework React).

Більш того, реалізація з використанням Java + React (з використанням бібліотеки simscript) не є універсальною на відміну від реалізації мовами Java та Javascript.

Програма написана Javascript є більш простою у реалізації, універсальною та найшвидшою з усіх вище наведених програм.

Отже, виходячи з усього вище написаного прийнято рішення реалізувати програму мовою Javascript.

3.2 Результати експериментів та їх аналіз

Розроблений алгоритм пошуку найефективнішої схеми технологічного процесу зводиться до знаходження діапазону кількості необхідних верстатів.

Найменше значення кількості верстатів показує скільки їх потрібно при найкращих умовах роботи (найменший час на виготовлення кожної деталі), найбільше значення відображає кількість при найгірших умовах роботи (найбільший час на виготовлення кожної деталі).

При стандартному алгоритмі пошуку найефективнішого технологічного процесу зазвичай використовується метод перебору.

У ході виконання роботи було реалізовано обидва алгоритми.

У таблиці 3.4 наведені результати проведення експериментів щодо швидкодії методів.

На рисунках 3.10 і 3.11 наведено графіки залежності зміни часу пошуку найефективнішого технічного процесу від кількості фаз технологічних процесів та графік залежності зміни витрат у залежності від кількості фаз технологічних процесів.

Таблиця 3.4 – Дослідження швидкодії алгоритмів перебору та пошуку у діапазоні

№	Кількість фаз технологічного процесу	Комбінаторний метод		Діапазонний метод	
		t, мс	$S_{вит.}$, од.	t, мс	$S_{вит.}$, од.
1	3	12,73	2265,0	0,39	2205,0
2	4	18,4	2750,0	0,7	2690,0
3	5	24,7	4505,0	0,9	4420,0
4	6	29,8	5350,0	1,19	5105,0
5	7	38,12	6120,0	1,52	5985,0
6	8	46,81	6780,0	1,93	6525,0
7	9	51,23	7900,0	2,01	7655,0
8	10	59,49	8015,0	2,24	7700,0

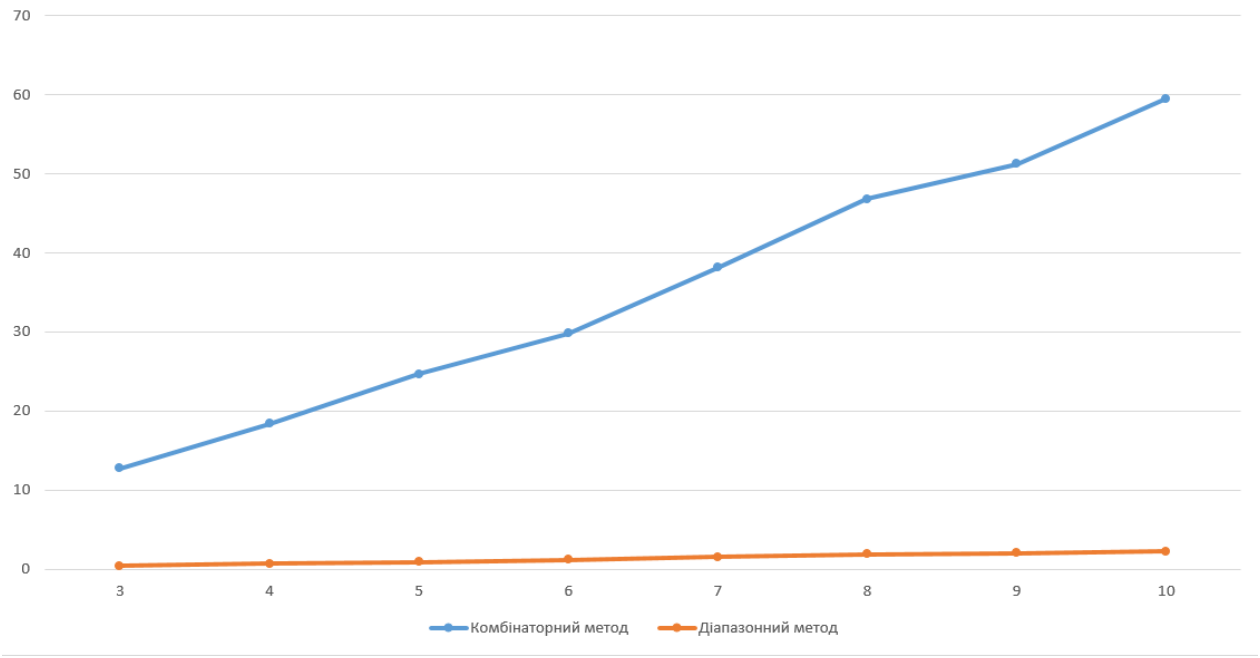

Рисунок 3.10 – Графік залежності зміни часу пошуку найефективнішого технічного процесу від кількості фаз технологічних процесів

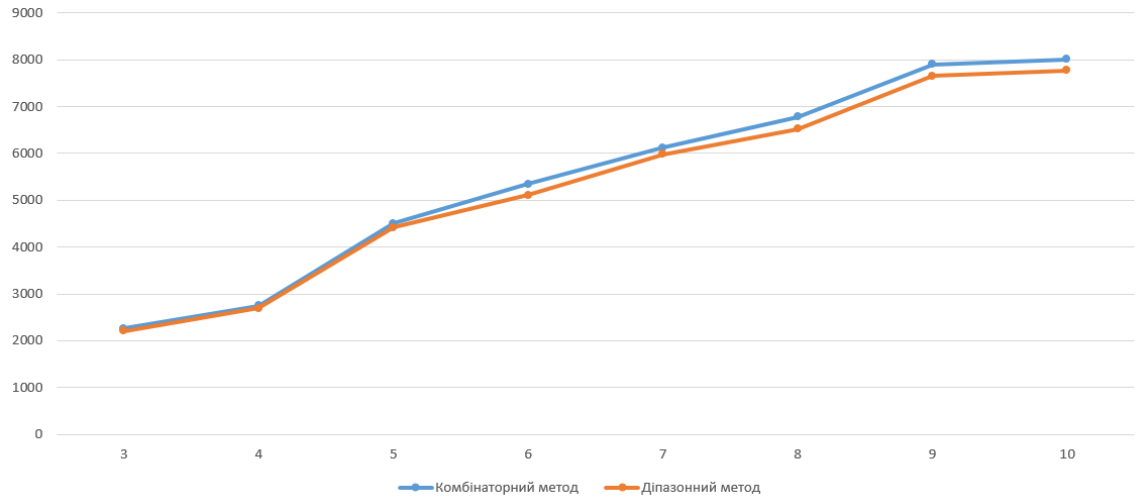

Рисунок 3.11 – Графік залежності зміни витрат у залежності від кількості фаз технологічних процесів

Виходячи з отриманих результатів експериментів можна стверджувати, що чим більша кількість фаз у технологічному процесу, тим більшу перевагу має діапазонний метод.

3.3 Висновки

У третьому розділі були розглянуті сучасні мови програмування. Було проаналізовано способи розробки додатку з використанням різних мов. Найкращим підходом виявилося використання мови React. Додаток розроблений цією мовою у понад 20 разів швидший за додатки розроблені іншими мовами (дані експериментів наведено у таблицях 3.1 – 3.3).

Було проведено порівняння розробленого алгоритму пошуку найефективнішого технологічного процесу з комбінаторним алгоритмом; встановлено, що розроблений алгоритм витрачає набагато ефективніший за базовий комбінаторний алгоритм, перевага розробленого алгоритму збільшується при збільшенні кількості фаз технологічного процесу.

4 ОХОРОНА ПРАЦІ

4.1 Загальні положення

Людина, яка працюватиме з розробленим додатком є технологом з навичками у програмуванні. Програміст-технолог повинен працювати у спеціально обладнаному приміщенні.

До виконання робіт допускаються особи, які пройшли навчання, стажування, інструктаж з питань охорони праці, у тому числі при виконанні робіт з підвищеною небезпекою, ознайомлені з правилами поводження при виникненні аварій та надання першої допомоги потерпілим від нещасних випадків відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 № 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за № 231/10511 (НПАОП 0.00-4.12-05) [37].

Графік роботи програміста встановлюється згідно правил внутрішнього трудового розпорядку.

Програміст зобов’язаний [37]:

– піклуватися про особисту безпеку і здоров’я, а також про безпеку і здоров’я оточуючих людей у процесі виконання будь-яких робіт або під час знаходження на території підприємства;

– знати і виконувати вимоги інструкцій з охорони праці і по видах робіт на своєму робочому місці;

– виконувати роботу відповідно до вимог інструкційно-технологічної карти;

– вміти користуватися засобами індивідуального і колективного захисту;

– знати і виконувати Правила поводження з устаткуванням, інвентарем, користуватися технічним паспортом на устаткування;

– знати і виконувати обов’язки з охорони праці, передбачені колективним договором (трудовим договором), правилами внутрішнього трудового розпорядку підприємства, в тому числі:

– вчасно починати і закінчувати роботу, дотримуватися розкладу технологічної і обідньої перерв;

– не виконувати роботи, що не передбачені змінним завданням;

– не перебувати на роботі в неробочій час без відповідного розпорядження керівника;

– дотримуватись правил корпоративного поводження;

– проходити в установленому порядку медичні огляди;

– вміти надати першу допомогу потерпілому від нещасного випадку;

– перед початком роботи перевіряти справність устаткування, огороджень, інженерно-технічних засобів безпеки, інвентарю, засобів пожежогасіння;

– співпрацювати з роботодавцем у справі організації безпечних і нешкідливих умов праці, особисто вживати можливих заходів щодо усунення будь-якої ситуації, що створює загрозу її життю чи здоров’ю або людям, які її оточують та навколишньому природному середовищу;

– при виявленні недоліків чи небезпеки зобов’язана повідомляти безпосереднього керівника або іншу посадову особу.

4.2 Загальні вимоги перед початком робіт

Перед початком робіт програміст повинен перевірити [37]:

– справність обладнання, інструменту, приладів;

– наявність і справність достатнього освітлення, вентиляції, обладнання тощо;

– перевірити справність рубильників, розеток, штепсельних з’єднань тощо.

У випадку виявлення будь-яких відхилень, несправностей, пошкоджень,

програміст повинен негайно повідомити директора підприємства.

4.3 Загальні вимоги під час виконання роботи

Під час виконання роботи програміст повинен [37]:

– виконувати роботу згідно із своїми посадовими обов’язками;

– не залишати без нагляду своє робоче місце, коли обладнання підключено до електромережі;

– у випадку виявлення будь-яких відхилень, несправностей, пошкоджень негайно повідомити директора підприємства.

4.4 Вимоги безпеки після закінчення роботи

Після закінчення роботи програміст повинен [37]:

– перевірити своє робоче місце;

– відключити від електромережі електрообладнання;

– закрити вікна;

– вжити заходів особистої гігієни: старанно вимити руки, при можливості прийняти душ;

– привести в порядок спеціальний одяг, зняти і прибрати його в окреме місце.

4.5 Вимоги безпеки при аварійній ситуації

При аварійній ситуації програміст зобов’язаний [37]:

– при виявлені небезпечної ситуації (пожежа, землетрус, радіаційна безпека, неполадки в електрогосподарстві тощо) для власного життя та життя співробітників заспокоїти і заспокоїти оточуючих;

– не усувати самому несправностей електромережі та електрообладнання, а

вимкнути загальне електропостачання;

– при виявленні пожежі зобов’язаний негайно викликати пожежну частину;

– вжити заходів згідно з планом евакуації на випадок пожежі, виробничих та природних явищ та вивести працівників у безпечне місце. Організувати роботу ДПД щодо збереження майна та цінних паперів;

– при появі сторонньої особи, яка застосовує протиправні дії щодо безпеки життєдіяльності оточуючих, викликати міліцію;

– у випадку травмування працівників або клієнтів під час роботи підприємства необхідно викликати швидку допомогу або за потреби надати першу долікарську допомогу, за необхідності створити комісію по розслідування нещасного випадку, видати акт встановленого зразка, наказ про підсумки розслідування, повідомлення про наслідки нещасного випадку.

Дії при наданні першої долікарської допомоги [37]:

– надання першої медичної допомоги починати з оцінки загального стану потерпілого і на підставі цього скласти думку про характер пошкодження;

– у разі різкого порушення або відсутності дихання, зупинки серця негайно зробити штучне дихання та зовнішній масаж серця, викликати швидку медичну допомогу.

Дії при ураженні електричним струмом [37]:

– необхідно звільнити потерпілого від дії електричного струму, відключивши електрообладнання від джерела живлення, а при неможливості відключення – відтягнути його від струмоведучих частин за одяг або застосувавши підручний ізоляційний матеріал;

– за відсутності у потерпілого дихання і пульсу необхідно робити йому штучне дихання і непрямий (зовнішній) масаж серця, звернувши увагу на зіниці. Розширені зіниці свідчать про різке погіршення кровообігу мозку. При такому стані оживлення необхідно починати негайно, після чого викликати швидку медичну допомогу.

Дії при пораненні [37]:

– для надання першої допомоги при пораненні необхідно розкрити індивідуальний пакет, накласти на рану стерильний перев'язувальний матеріал і зав'язати її бинтом;

– якщо індивідуального пакету немає, то для перев'язки необхідно використати чисту носову хустинку, чисту полотняну ганчірку тощо. На те місце ганчірки, що приходиться безпосередньо на рану, бажано накапати декілька капель настойки йоду, щоб одержати пляму розміром більше рани, а після цього накласти ганчірку на рану.

Дії при переломах, вивихах, ударах, розтягненні [37]:

– при переломах і вивихах кінцівок необхідно пошкоджену кінцівку укріпити шиною, фанерною пластинкою, палицею, картоном або іншим подібним предметом. Пошкоджену руку можна також підвісити за допомогою перев'язки або хустки до шиї і прибинтувати до тулуба;

– при передбачуваному переломі черепа (несвідомий стан після удару голови, кровотеча з вух або рота) необхідно прикласти до голови холодний предмет (грілку з льодом або снігом, чи холодною водою) або зробити холодну примочку;

– при підозрі перелому хребта необхідно потерпілого покласти на дошку, не підіймаючи його, чи повернути потерпілого на живіт обличчям у низ, наглядаючи при цьому, щоб тулуб не перегинався з метою уникнення ушкодження спинного мозку;

– при переломі ребер, ознакою якого є біль при диханні, кашлю, чханні, рухах необхідно туго забинтувати груди чи стягнути їх рушником під час видиху.

Дії при теплових опіках [37]:

– при опіках вогнем, парою, гарячими предметами ні в якому разі не можна відкривати пухирі, які утворюються, та перев'язувати опіки бинтом;

– при опіках першого ступеня (почервоніння) обпечене місце обробляють ватою, змоченою етиловим спиртом; при опіках другого ступеня (пухирі) обпечене

місце обробляють спиртом, 3 % марганцевим розчином або 4 % розчином таніну;

– при опіках третього ступеня (зруйнування шкіряної тканини) накривають рану стерильною пов’язкою та викликають лікаря.

Дії при кровотечі [37]:

– для того, щоб зупинити кровотечу, необхідно підняти поранену кінцівку вверх, кровоточиву рану закрити перев’язувальним матеріалом (із пакета), складеним у клубочок, придавити її зверху, не торкаючись самої рани;

– при сильній кровотечі, яку не можна зупинити пов’язкою, застосовується здавлювання кровоносних судин, які живлять поранену область, за допомогою згинання кінцівок у суглобах, а також пальцями, джгутом або закруткою.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістранта було запропоновано рішення актуальної науково-прикладної задачі підвищення ефективності комп’ютерно-інтегрованих технологічних процесів за рахунок розробки засобу їх структурно-параметричного реінжинірингу.

За результатами проведеного огляду й аналізу проблемної області була встановлена необхідність створення системи реінжинірингу технологічних процесів, що здійснюватиметься одночасно за множиною показників.

Для досягнення мети було сформульовано постановку задачі структурно-параметричного реінжинірингу ТП за показниками визначити завантаженості обладнання та обсягів виробництва, обрано базовий комбінаторний метод її розв’язання та запропоновано його удосконалення. Після побудови відповідних алгоритмів було обґрунтовано вибір мови програмування, середовища розробки програм та програмування методів розв’язання задачі. Розроблено компоненти математичного та програмного забезпечення системи структурно-параметричного реінжинірингу комп’ютерно-інтегрованих ТП. Створено веб-додаток, що двома методами виконує генерацію та моделювання різних варіантів схем побудови ТП і надає користувачеві найефективніший серед них.

Створений додаток може використовуватися на підприємствах і в організаціях, де вирішуються задачі проєктування чи реінжинірингу ТП.

Практичне використання розробки дозволить без втрати точності рішень скоротити час розв’язання задачі структурно-параметричної оптимізації комп’ютерно-інтегрованих технологічних процесів.

За результатами виконаних досліджень було підготовлено доповіді на: V Міжнародну конференцію «Виробництво & Мехатронні Системи 2021» (м. Харків, 21-22 жовтня 2021 р.) [4]; 11-ту Міжнародну науково-технічну конференцію

«Інформаційні системи та технології» (м. Харків, 22-25 листопада 2022 р.) [5]; Всеукраїнську науково-практичну конференцію здобувачів вищої освіти і молодих учених «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» (м. Харків, 23 листопада 2022 р.) [6].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ханджян В. В., Безкоровайний В.В. Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві // Матеріали всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених. Харків, ХНАДУ, 2021. С. 188-191.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

3. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП "УкрНДНЦ". 2016. 30 с.

4. Безкоровайний В. В., Ханджян В. В. «Математична модель багатокритеріальної задачі структурно-параметричної оптимізації виробничих технологічних процесів» // Виробництво & Мехатронні Системи 2022: матеріали VI Міжнародної конференції, Харків, 2022. С. 133-135.

5. Ханджян В., Безкоровайний В. Формалізація задачі структурно-параметричної оптимізації технологічних процесів // Інформаційні системи та технології: матеріали 11-ї Міжнародної наук.-техн. конф. Ч. 2 (м. Харків, 22-25 листоп. 2022 р.), Х.: ХНУРЕ, 2022. С. 13–14. URL: https://istconf.sedep.online/archive/ist_2022_part_2.pdf (дата звернення 03.12.2022).

6. Ханджян В. В., Безкоровайний В.В. Структурно-параметрична оптимізація технологічних процесів на етапі реінжинірингу // Матеріали всеукраїнської

науково-практичної конференції здобувачів вищої освіти і молодих учених. Харків, ХНАДУ, 2022.

7. Технологічний процес та його структура [Електронний ресурс]. 2017. URL: https://tehnar.net.ua/tehnologicheskiy-protsess-i-ego-struktura/.

8. Pogozhikh M., Pak A. The development of an artificial energotechnological process with the induced heat and mass transfer // Eastern-European Journal of Enterprise Technologies. 2017. С. 50-57.

9. Технологічний процес [Електронний ресурс]. URL: http://opiobjektid.tptlive.ee/Automatiseerimine/3__.html.

10. Реінжиніринг бізнес-процесів як сучасний метод управління стратегічними змінами на підприємстві / Л. М.Сакун, Л. В. Сухомлин, Л. В. Різніченко, Б. О. Вєлькін. 2020. С. 81–82.

11. Безкоровайний В.В., Демська Н.П. Системологічний аналіз проблеми реінжинірингу виробничих технологічних процесів // Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку: Міжнар. науково-практ. конф. Харків, 2022. С. 100.

12. Кулик К.О., Вислоухов С.П. Імітаційне моделювання технологічних процесів // Ефективність інженерних рішень у приладобудуванні: Всеукр. науково-практ. конф. Київ, 2019. С. 140-143.

13. Критерии эффективности производственного процесса [Електронний ресурс]. URL: https://helpiks.org/2-73741.html.

14. Верезуб Н. В., Островерх Е. В., Симонова А. А. Системний аналіз, структурна та параметрична оптимізація технологічних процесів. Харків, 2012.

15. Сідорчук Є.І. Методи структурно-параметричної оптимізації дискретних виробничих технологічних процесів. Харків, 2020. 76 с.

16. Beskorovainyi V., Petryshyn L., Shevchenko O. «Specific subset effective option in technology design decisions», Applied Aspects of Information Technology, Vol. 3., No.1, 2020. pp. 443-455.

17. Beskorovainyi, V. «Combined method of ranking options in project decision support systems», Innovative Technologies and Scientific Solutions for Industries, No 4 (14), 2020. pp. 13-20.

18. Кондрук Н. Е., Маляр М. М. Багатокритеріальна оптимізація лінійних систем. Ужгород: АУТДОР-ШАРК, 2019. 76 с.

19. Кохендерфер М., Уілер Т. Алгоритми оптимізації. 2020. 528 с.

20. Karian, Zaven A., Edward J. Dudewicz. Modern statistical, systems, and GPSS simulation. 2020. 513 p.

21. North, Michael J., Charles M. Macal. Agent-Based Modeling and Computer Languages. 2020. 889 p.

22. Томашевський В. Н., Жданова Е. Г. Імітаційне моделювання у середі GPSS. Київ, 1998. 123 с.

23. Молдабаева М. Н. Автоматизація технологічних процесів та виробництв. Вологда: Инфра-Инженерия, 2019.

24. Popov, George. GPSS language as tool for reliability simulations. 2017. 463 p.

25. Paredis, R., Van Mierlo, S., Vangheluwe, H. Translating process interaction world view models to DEVS: GPSS to (Python (P)) DEVS // Winter Simulation Conference. 2020. pp. 2221-2232.

26. Fakhar F. Comparative study of computer simulation softwares // Journal of Artificial Intelligence in Electrical Engineering. 2019. pp. 1-19.

27. Java [Електронний ресурс]. 2022. URL: https://uk.wikipedia.org/wiki/Java.

28. Еванс Б., Фленеган Д. Java. Посібник розробника. Санкт-Петербург, 2019. 592 с.

29. Лонг Дж. Java у хмарі. 2019. 624 с.

30. Троелсен Е., Джелікс Ф. Мова програмування C# 7 та платформи .NET і .NET Core. 2018. 1330 с.

31. Alls J. Clean Code in C#: Refactor your legacy C# code base and improve application performance by applying best practices. Packt publishing, 2020. 500 с.

32. Плюси та мінуси C++ [Електронний ресурс]. 2015. URLs: http://cjblogr.blogspot.com/2015/02/blog-post.html.

33. Мейєрс С. Ефективний та сучасний C++. Київ, 2016. 304 с.

34. Лутц М. Вивчаємо Python. Санкт-Петербург, 2019. 832 с.

35. Волошин О. Переваги і недоліки мови Python [Електронний ресурс]. 2020. URL: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python.

36. Що таке Python: переваги програмування на Python [Електронний ресурс]. 2020. URL: https://futurenow.com.ua/shho-take-python-piton-perevagy-programuvannya-na-python/.

37. Інструкція з охорони праці для програміста. Актуалізовано на 08.12.2017р. URL: https://www.victorija.ua/blanki-ta-formi-dokumentiv/instruktsiya-z-ohorony-pratsi-dlya-prohramista-aktualizovano-na-08-12-2017r.html.