МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ І ІНФОРМАТИКИ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни "Конструювання програмного забезпечення"
для студентів денної та заочної форм навчання
спеціальностей 121 Інженерія програмного забезпечення,
122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 125 Кібербезпека

Луцьк – 2024

УДК 004.415
М 54

Методичні вказівки до лабораторних робіт з дисципліни "Конструювання програмного забезпечення" для студентів денної та заочної форми навчання спеціальностей 121 Інженерія програмного забезпечення, 122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 125 Кібербезпека / уклад. В.І. Костін. – Луцьк : ДонНТУ, 2024. – 29 с.

У методичних вказівках розглянуто обсяг і зміст лабораторних робіт з дисципліни «Конструювання програмного забезпечення для студентів денної та заочної форми навчання спеціальностей 121 Інженерія програмного забезпечення, 122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 125 Кібербезпека. Наведено основні теоретичні положення, завдання та типові приклади щодо виконання лабораторних робіт, а також перелік рекомендованої літератури.

Укладач: Костін В.І., старший викладач кафедри ПМІ
.

Рецензент: Штепа О.А., к.т.н., доцент, доцент кафедри ЕТКІ

Відповідальний за випуск: Н.О. Маслова, завідувач кафедри ПМІ к.т.н., доцент

Затверджено навчально-методичним відділом ДонНТУ,
протокол № 9 від 30.04.2024 р.

Розглянуто на засіданні кафедри прикладної математики і інформатики,
протокол № 5 від 30.04.2024

ЗМІСТ

ВСТУП

Лабораторні роботи з дисципліни "Конструювання програмного забезпечення" є складовою частиною навчального процесу підготовки бакалаврів денної та заочної форми навчання, що навчаються за спеціальностями 121 Інженерія програмного забезпечення, 122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 125 Кібербезпека.

Основна мета лабораторних робіт - підвищення якості підготовки бакалаврів з програмної інженерії, які повинні знати сучасні способі та методи створення програмних продуктів, володіти методологіями створення програмного забезпечення.

Метою лабораторних робіт є отримання студентами практичних навичок програмування обчислюваних та інформаційних об'єктів.

Лабораторні роботи передбачають попереднє опрацювання лекційного матеріалу та матеріалу для самостійного вивчення.

При виконанні лабораторних робіт студенти набувають навичок приймати інженерні програмні рішення, користуватися технічною та довідковою літературою, що дозволить підвищити рівень проектування програмних продуктів.

Методичні вказівки складаються з чотирьох завдань.

Методичні вказівки розроблено відповідно до вимог кваліфікаційних характеристик бакалавра, призначено для надання допомоги студентам у виконанні і оформленні лабораторних робіт.

Лабораторна робота №1

ПЛАНУВАННЯ ОРГАНІЗАЦІЇ РОБІТ НАД ПРОЕКТОМ ПРОГРАМ

Мета роботи: набути практичних навичок у застосуванні методів мережевого планування розробки великих програмних систем у задані терміни та з оцінкою необхідних ресурсів.

Теоретичні положення

Цілями планування організації робіт над комплексом програм є:

- складання субоптимального плану розробки комплексу,
- раціональний розподіл ресурсів,
- оптимізація термінів виконання комплексу робіт.

У цій лабораторній роботі розглядається мережевий метод, що становить теоретичну основу будь-якої схеми організації робіт над проектом.

Мережеві методи застосовуються для раціонального планування великих програмних комплексів.

Базовим поняттям мережевого планування є робота – витрати, пов'язані з проектуванням, кодуванням та тестуванням одного модуля.

Вихідними даними для планування є:

- тривалість кожної k-ої роботи ($\boldsymbol{T_k}$-витрати на проектування, кодування та тестування модуля, дні),
- інтенсивність розробки модуля ($\boldsymbol{Q_k}$, людина/день),
- кошти на виконання робіт ($\boldsymbol{C_k}$, грн.) тощо.

Ці дані готуються досвідченими програмістами на основі особистого досвіду, статистичних даних та експертних оцінок.

Метою планування є складання мережевого графіка.

Насамперед складається діаграма висхідного чи низхідного проектування. Для кожної роботи задається вага – номер рівня у дереві ієрархії (знизу – при висхідному проектуванні, зверху – при низхідному).

Після цього роботи сортуються за зростанням ваги і для кожної роботи знаходиться безліч безпосередньо попередніх їй робіт і безліч наступних робіт.

Потім визначається ранній термін закінчення кожної роботи за формулою:

$$T'_k = \max_{i \in \omega'_k} (T'_i + \tau_k).$$

Час завершення всього комплексу робіт визначається так:

$$T = \max_k T'_k$$

Після цього визначається пізній термін закінчення кожної роботи:

$$T_k'' = \min_{i \in \varpi_k''} (T_i'' - \tau_i),$$

де $T_N'' = T$.

Ця характеристика визначається від останньої роботи (у ранжованому списку) до першої.

Потім обчислюються резерви часу кожної роботи:

$$\Delta\tau_k = T_k'' - T_k'.$$

Резерви часу є важливою характеристикою, що показує, який термін можна розтягнути виконання роботи, не збільшуючи час виконання всього комплексу робіт.

Роботи з нульовим резервом називаються критичними, їм має приділятися більше уваги, ніж іншим.

Після цього обчислюються ранні початку кожної роботи:

$$\boldsymbol{T_k^0 = T_k' - \tau_k.}$$

Після цього складають мережевий графік та діаграму розподілу людських ресурсів.

Для оптимізації розподілу ресурсів перерозподіляють роботи у межах наявних резервів часу.

Завдання до лабораторної роботи

Скласти раціональний мережевий графік реалізації проекту програмного комплексу відповідно до варіанта.

Варіанти завдань

1. Створити інформаційну систему «Бібліотека», яка виконує функції пошуку книги в каталозі, автоматизацію роботи бібліотекаря (видача/прийом книги, довідка про видання книги, наявність книги у певному відділі).
2. Створити навчальну систему з можливістю тестування знань учня.
3. Створити інформаційну систему «Лікарня», яка виконує функції видачі довідки про графік прийому лікарів, про хворих, які зверталися до лікарні, про процедури, призначені хворим.
4. Створити інформаційну систему «Дипломування», що дозволяє отримати інформацію про викладачів, студентів-дипломників, теми дипломних проектів, рецензентів, оцінки.
5. Створити функціональну модель діяльності банку з огляду на те, що сучасні банки надають своїм клієнтам широкий спектр послуг, починаючи від обслуговування рахунків, прийняття внесків, кредитування й закінчуючи роботою на ринку цінних паперів, роботою з інвестиціями, валютними операціями та іншими можливими напрямками діяльності.
6. Створити інформаційну систему «Книгарня», що виконує функції довідкової системи про наявність книг у магазині і дозволяє зробити покупку через Internet.

7. Створити функціональну модель діяльності великого автосалону, зважаючи на те, що автосалон робить послуги з гарантійного обслуговування клієнтів, має власну автомайстерню, працює безпосередньо з виробниками машин, з клієнтами, надає послуги з оформлення документів.

8. Створити медичну інформаційну системи лікарні.

9. Створити функціональну модель роботи банкомату з огляду на всі можливі транзакції й забезпечення обслуговування самого апарата: завантаження грошей, перевірка справності програмного забезпечення, забезпечення захисту від зловмисників

10. Створити інформаційну систему «Кінопрокат», що дозволяє отримати інформацію про прокат фільмів у кінотеатрах, ціни та наявність квитків.

11. Створити функціональну модель процесу роботи пункту налагодження електронної техніки з огляду на всі можливі види послуг, забезпечення роботи майстрів, закупівлю деталей, роботу з клієнтами.

12. Створити функціональну модель процесу роботи кадрового агентства, з огляду на різні методи підбору персоналу - через Інтернет, за оголошеннями, робота з пошуку клієнтів. Описати роботу агентства.

13. Створити інформаційну систему про наявність ліків в аптеках міста, що дозволяє зробити замовлення на ліки.

14. Створити функціональну модель діяльності бухгалтерії промислового підприємства. Бухгалтерія обробляє рахунки-фактури від постачальників, клієнтів, нараховує заробітну плату співробітникам, обробляє інформацію з контрактів, працює з податковими органами й соціальними фондами.

15. Створити інформаційну систему «Готель», що дозволяє отримати відомості про наявність вільних номерів, ціни та забронювати номер.

16. Створити інформаційну систему «Страхові компанії України», що дозволяє отримати довідку про послуги страхування, ціни на послуги різних компаній, відомості про укладені договори страхування.

17. Створити функціональну модель діяльності великого салону з продажу холодильного обладнання, зважаючи на те, що салон надає послуги з гарантійного обслуговування клієнтів, має власну майстерню, працює безпосередньо з виробниками машин, з клієнтами, надає послуги з оформлення документів.

18. Створити інформаційну систему «Екологічна служба», що дозволяє отримати відомості про підприємства, катастрофи, що сталися на них, завдані збитки.

19. Створити функціональну модель діяльності комп'ютерної фірми, з огляду на те, що фірма торгує комп'ютерами в зібраному вигляді й комплектуючими. Фірма працює як з виробниками комп'ютерної техніки, так і з клієнтами. Фірма надає ряд додаткових послуг: установка програмного забезпечення, підключення до мережі Інтернет, гарантійне обслуговування й т.д.

20. Створити сайт туристичної агенції, що дозволяє ознайомитися з пропонованими турами, забронювати тур або авіаквитки.

21. Створити функціональну модель діяльності торговельної фірми по реалізації косметичної продукції з огляду на роботу фірми із клієнтами, постачальниками, доставку косметики від постачальників і по торговельних місцях клієнтів.

22. Створити інформаційну систему «Музеї світу», що дозволяє отримати відомості про музеї, експонати та їх участь у виставках.

23. Створити функціональну модель діяльності торговельної фірми по реалізації продовольчої продукції з огляду на роботу фірми із клієнтами, постачальниками, доставку продукції від постачальників і по торговельних місцях клієнтів.

24. Створити функціональну модель діяльності кафедри вищого навчального закладу з огляду на наступні напрямки: робота із забезпечення навчального процесу, робота за господарськими договорами, науково-дослідницька робота співробітників і студентів і т.д.

25. Створити функціональну модель роботи аеропорту з огляду на роботу аеропорту з авіакомпаніями, клієнтами, постачальниками й т.д. Урахувати різноманітні роботи аеропорту з технічного обслуговування літаків, обслуговування клієнтів через каси, роботу диспетчерської служби аеропорту.

26. Створити сайт фотостудії, що дозволяє ознайомитись з послугами, розцінками, зразками робіт, зробити замовлення.

27. Створити функціональну модель роботи будівельної фірми. Описати роботу фірми як з постачальниками, так і з клієнтами. Треба відзначити, що в цей час будівельні організації забезпечують повний технологічний процес, починаючи з проведення досліджень ринку, створення проекту, закупівлі матеріалів, до безпосереднього будівництва й остаточного продажу квартир.

28. Створити функціональну модель діяльності вищого навчального закладу з огляду на його роботу як по основних напрямках діяльності (забезпечення навчального процесу, наукова діяльність), так і по додаткових процесах: міжнародна діяльність, робота за договорами, соціальна робота.

Порядок виконання роботи

1. Проаналізувати принципи традиційних стратегій проектування "зверху-вниз", "знизу-вгору" або "вертикальне шарування" і сформувати свій підхід до проектування програми, що поєднує переваги традиційних стратегій, специфіку розв'язуваного завдання, індивідуальний досвід та організаційні стереотипи. Під час планування порядку розробки програмних модулів використовувати ієрархічний, операційний чи комбінований підходи.

2. Планування порядку розробки програмних модулів слід розпочати із складання діаграми майбутніх робіт. Діаграма будується з урахуванням обраного підходу до проектування комплексу, структура якого визначена схемою розкладання.

3. Для кожної роботи, наведеної на діаграмі, необхідно задати такі вихідні дані: тривалість роботи, інтенсивність розробки модуля і т.д. Вихідні дані визначити за допомогою експертних оцінок передбачуваних витрат часу програміста(ів) на розробку та тестування кожного модуля окремо та необхідних для цього обчислювальних ресурсів.

4. Розрахувати параметри мережного графіка.

5. Зобразити мережевий графік, враховуючи ранній початок кожної роботи, із зазначенням наявних резервів часу; визначити критичні роботи та намалювати графік розподілу ресурсів.

6. Побудувати субоптимальний мережевий графік, перерозподіляючи роботи в межах наявних резервів часу, і відповідну йому діаграму розподілу людських ресурсів.

Приклад виконання роботи

Як приклад виконаємо розрахунок мережевого графіка розробки гіпертекстової навчальної системи за одним із розділів програмування.

Схема ієрархії гіпертекстової навчальної системи може мати такий вигляд (рис.1).

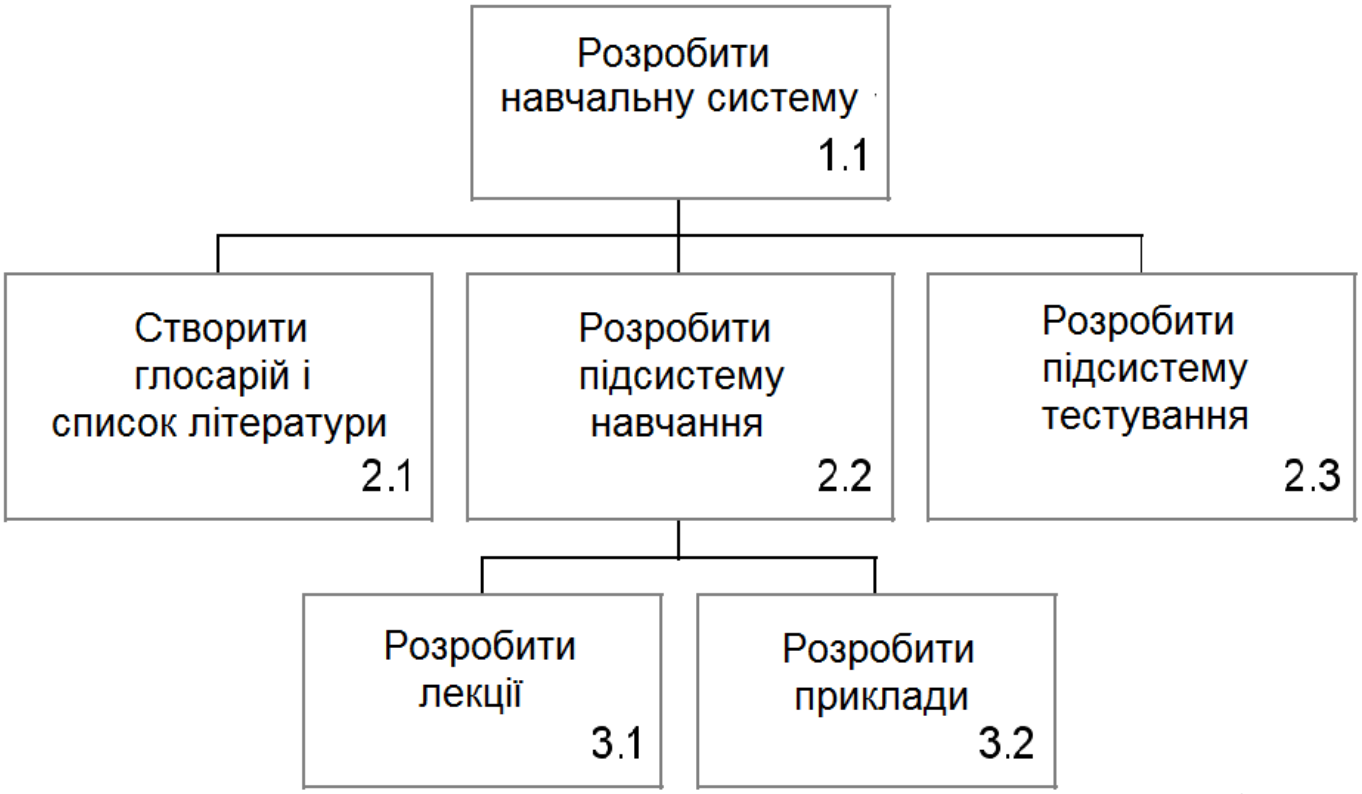

Рисунок 1 - Схема ієрархії довідкової системи

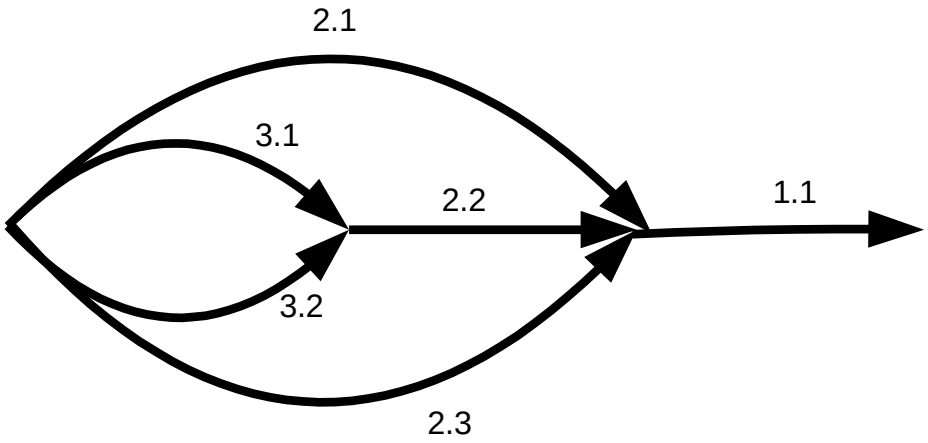

Рисунок 2 - Діаграма висхідного проектування даної системи

Сформуємо вихідні дані розробки мережевого графіка (табл. 1).

Таблиця 1 - Вихідні дані

Робота	Вага	Тривалість τ_k	Інтенсивність Q_k
1.1	3	1	1
2.1	1	2	1
2.2	2	1	1
2.3	1	4	1
3.1	1	3	1
3.2	1	4	1

Виконаємо ранжування робіт з рівнями (табл.2).
Перший рівень – це роботи, які ні на що не спираються.
Другий рівень – це роботи, які спираються на роботи першого рівня.
Третій рівень - це роботи, які спираються на роботи другого рівня і так далі.

Таблиця 2 – Ранжування робіт з рівнями

Робота	Рівень	Тривалість τ_k	Інтенсивність Q_k
2.1	1	2	1
2.3	1	3	1
3.1	1	4	1
3.2	1	3	1
2.2	2	1	1
1.1	3	1	1

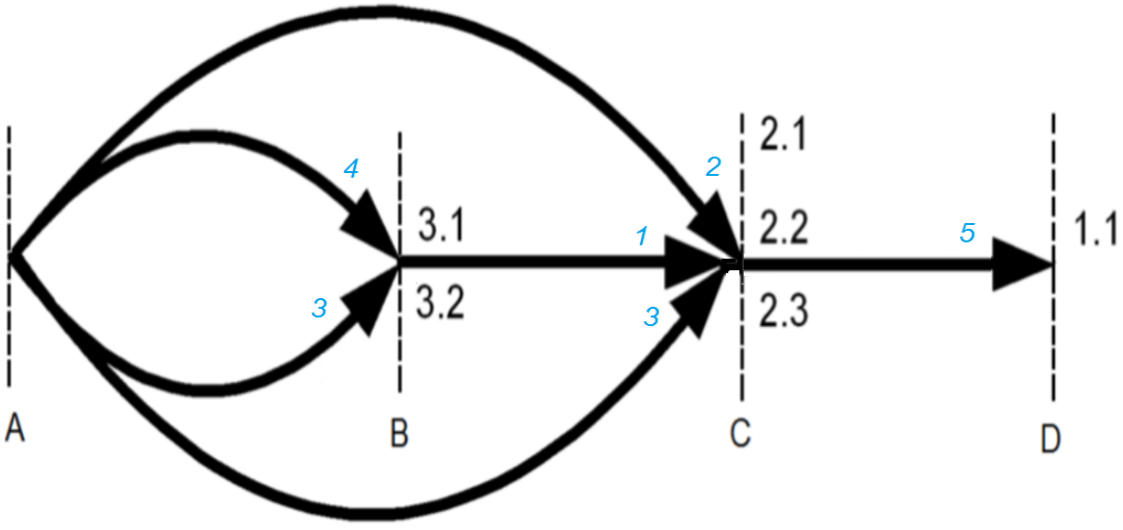

Рисунок 3 - Діаграма висхідного проектування з рівнями

Тепер сформуємо безліч безпосередньо попередніх (ω'_k) і наступних (ω''_k) робіт (табл. 3).

Таблиця 3 – Безліч безпосередньо попередніх (ω'_k) і наступних (ω''_k) робіт

Робота	Вага	ω'_k	ω''_k
2.1	1	–	1.1
2.3	1	–	1.1
3.1	1	–	2.2
3.2	1	–	2.2
2.2	2	3.1, 3.2	1.1
1.1	3	2.1, 2.2, 2.3	–

Визначимо найбільш ранній термін закінчення кожної роботи T'_k (табл. 4) за формулою:

$$T'_k = \max_{i \in \omega'_k}(T'_i + \tau_k)$$

Таблиця 4 – Найбільш ранній термін закінчення робіт

Робота	ω'_k	τ_k	T'_k
2.1	–	2	max(0)+2=2
2.3	–	3	max(0)+3=3
3.1	–	4	max(0)+4=4
3.2	–	3	max(0)+3=3
2.2	3.1, 3.2	1	max(4,3)+1=5
1.1	2.1, 2.2, 2.3	1	max(2,5,3)+1=6

Час завершення проекту T=max (2,4,4,4,5,6) = 6.

Розрахуємо пізній термін закінчення кожної роботи

$$T''_k = \min_{i \in \varpi''_k}(T''_i - \tau_i)$$

де

$$T''_N = T$$.

Будемо обчислювати у такій послідовності (табл. 5): 1.1, 2.2, 3.2, 3.1, 2.3, 2.1 (від останньої роботи до першої) .

Таблиця 5 – Пізній термін закінчення робіт

Робота	ω_k''	τ_k	T_k''
1.1	-	-	6
2.1	1.1	1	min(6)-1=5
2.3	1.1	1	min(6)-1=5
3.1	2.2	1	min(5)-1=4
3.2	2.2	1	min(5)-1=4
2.2	1.1	1	min(6)-1=5
початок	2.1,2.3,3.1,3.2–	1	min(5-2,5-3,4-4,4-3)=0

Потім обчислюються резерви часу для кожної роботи та ранні початку робіт (табл.6) за формулою:

$$\Delta\tau_k = T_k'' - T_k'$$

Таблиця 6 – Резерви часу для кожної роботи та ранні початки робіт

Робота	T_k	T_k'	T_k''	Резерви $\Delta\tau_к$	Ранні початки
2.1	2	2	5	3	0
2.3	4	3	5	2	0
3.1	4	4	4	0	0
3.2	4	3	4	1	0
2.2	1	5	5	0	4
1.1	1	6	6	0	5

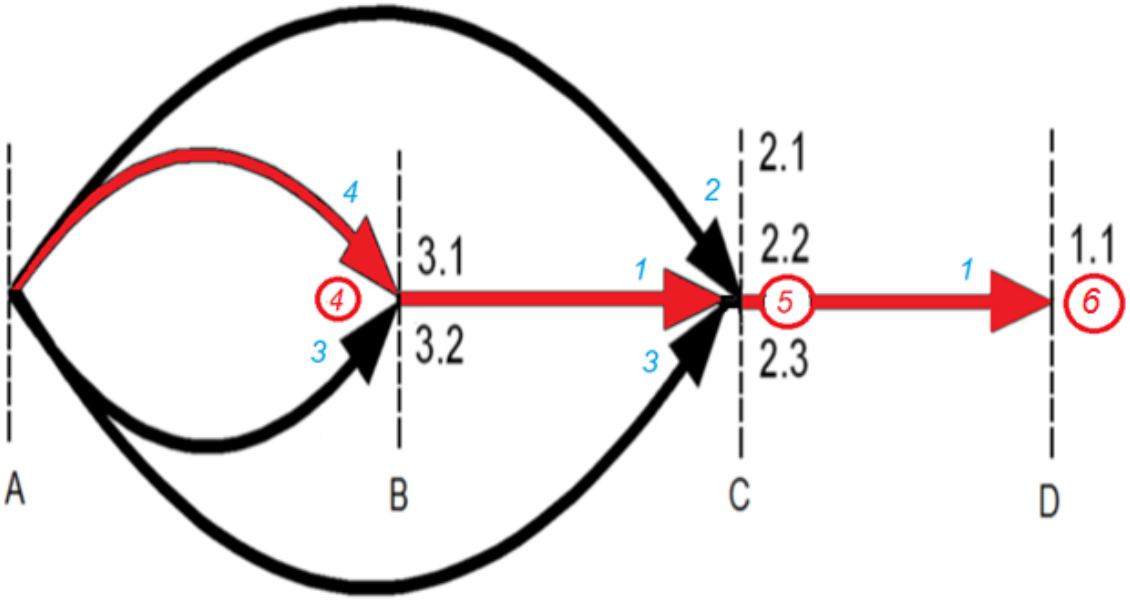

Рисунок 4 - Діаграма висхідного проектування

Складемо мережевий графік робіт (рис. 5).

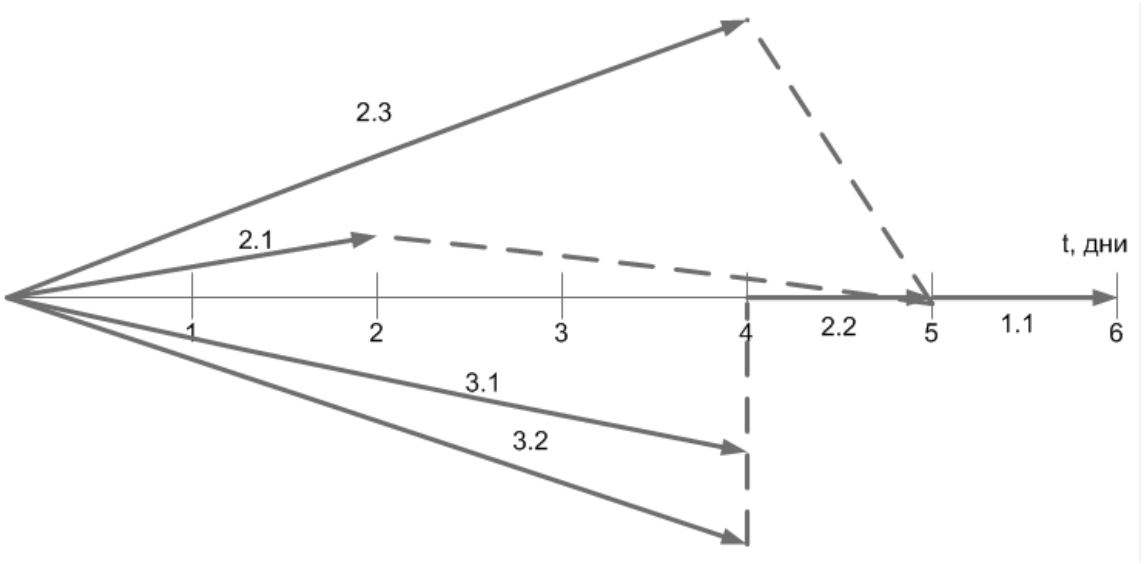

Рисунок 5 - Мережевий графік робіт

Розподіл ресурсів під час виконання робіт наведено на рис.6.

При необхідності початок робіт 2.1 і 2.3 можна зсунути в межах резервів, змінивши розподіл ресурсів на більш зручний.

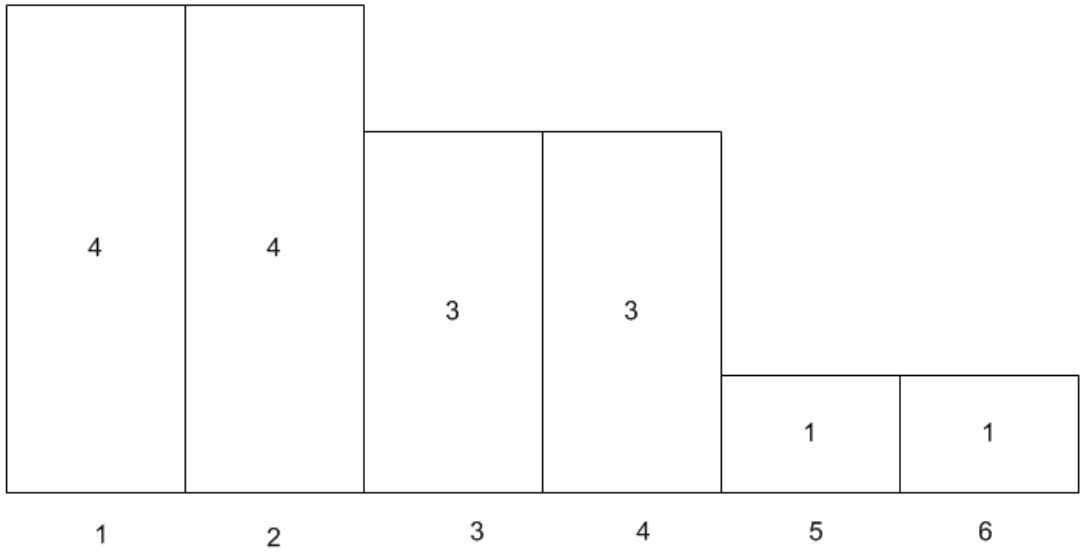

Рисунок 6 - Розподіл ресурсів під час виконання робіт

Це спрощений приклад для створення критичного шляху.

Реально необхідно враховувати, що для кожного модуля фактично виконуються такі етапи (рис.7):

- проектування модуля;
- кодування (програмування) модуля;
- тестування роботи модуля;
- тестування роботи модуля у системі;
- ці дії можуть тривати від одного до кількох днів.

Крім того, необхідно пам'ятати, що творчий колектив має складатися від однієї до трьох осіб.

Усе це треба вставити в діаграму висхідного проектування.

По кожному модулю має бути інформація:

- призначення модуля (що робить модуль, хто з ним працює – користувач, системний працівник, адміністратор);
- основні дії модуля;
- структура модуля (вхідні дані – від кого [користувача або модуля (його назва)], вихідні дані – кому [користувачу або модулю (його назва)]).

Зміст звіту

1. Назва лабораторної роботи, мета роботи.
2. Схема складу програмного комплексу. Опис вибраної стратегії (підходу) проектування комплексу.
3. Вихідні дані, початкова діаграма робіт, що відповідає застосовуваній стратегії проектування.
4. Розрахунок параметрів мережного графіка в табличній та графічній формі.
5. Субоптимальний графік робіт, розподіл ресурсів, критичні роботи, загальний термін виконання проекту.
6. Рекомендації щодо можливої зміни початка робіт.

Контрольні питання

1. У чому перевага мережевого планування розробки великих програмних систем?
2. Що є вихідними даними під час мережного планування?
3. Навіщо потрібен мережевий графік робіт і як складається для програмних комплексів?
4. Яким чином ранжуються роботи?
5. Як визначається критичний шлях на мережевому графіку і що він означає?
6. Завдяки чому можлива оптимізація графіка виконання робіт та необхідних ресурсів на реалізацію проекту?

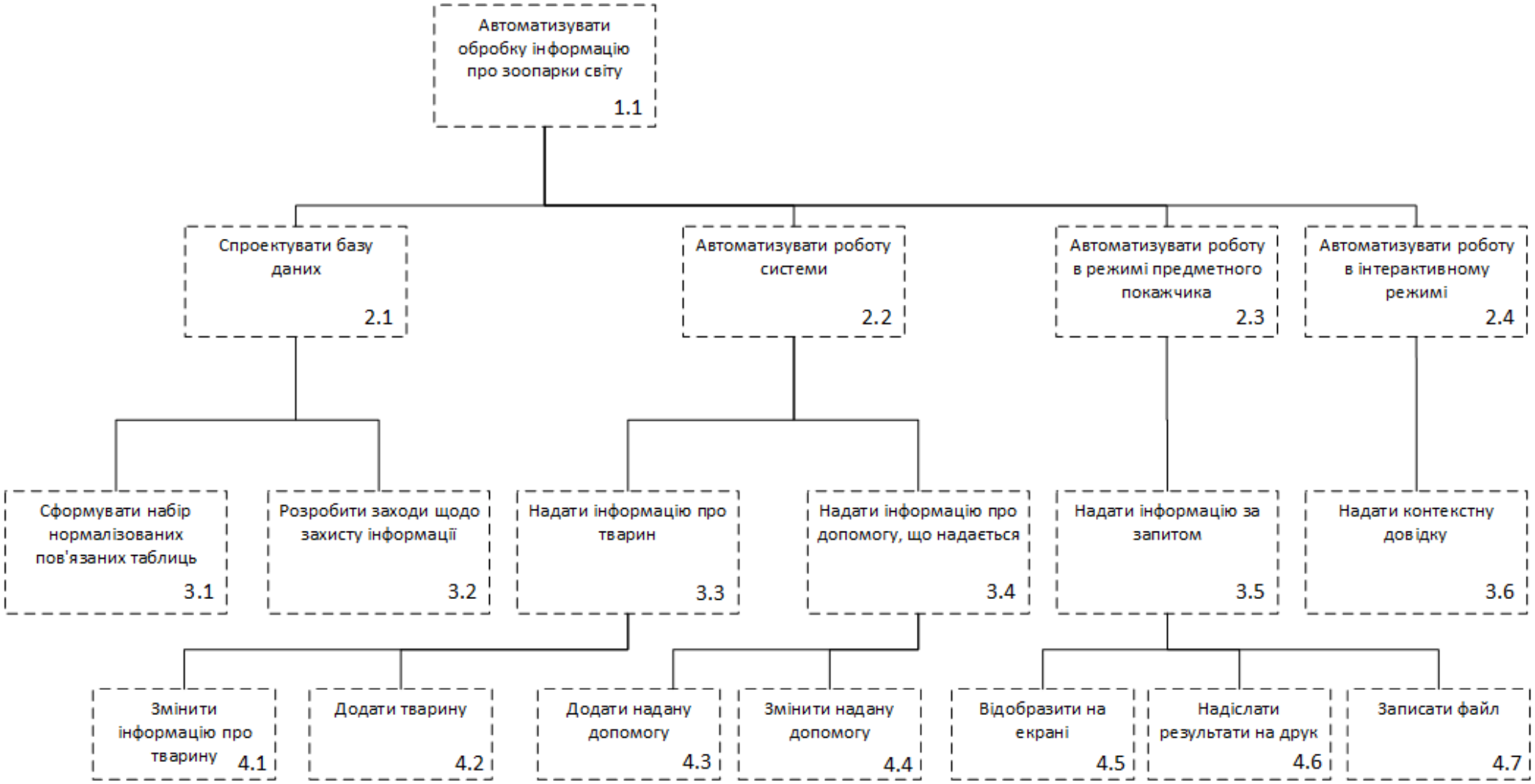

Рисунок 7 - Приклад розробленої схеми

Лабораторна робота №2

РОЗРОБКА ПРОГРАМИ З ВИКОРИСТАННЯМ МЕТОДИКИ ЕКСТРЕМАЛЬНОГО ПРОГРАМУВАННЯ

Мета роботи: набути практичних навичок у застосуванні методики екстремального програмування при вирішенні календарних завдань.

Завдання до лабораторної роботи

Створити програму для вирішення зазначеної у варіанті задачі та набір тестів для застосування екстремального програмування.

Варіанти завдань

1. Визначити кількість днів, що залишилися до кінця поточного місяця від заданої дати.
2. Визначити номер кварталу, до якого належить задана дата (1 квартал-січень-березень, 2 квартал-квітень-червень, 3 квартал-липень-вересень, 4 квартал-жовтень-грудень).
3. Визначити дату та назву дня, що передує заданому.
4. Визначити дату дня, віддаленого від заданого на 1 тиждень.
5. Визначити дату початку наступного тижня (відомі поточна дата та день тижня).
6. Визначити дату найближчого повного місяця, коли відома дата молодика (повний місяць настає через 14 днів після молодика).
7. Визначити дату виходу людини з відпустки, якщо відома дата початку відпустки та її тривалість у днях.
8. Визначити кількість днів у заданому році.
9. Визначити дату, віддалену від заданої на задану кількість тижнів.
10. Визначити пору року, до якої належить введена дата.
11. Визначити дату та назву дня кінця поточного тижня (відомі поточна дата та день тижня).
12. Визначити номер тижня від початку місяця для заданої дати.
13. Визначити, на яку пізню дату можна придбати квиток на поїзд у заданий день (продаж квитків починається за 45 діб до дати відправлення).
14. Визначити дату та назву дня, наступного за заданим.
15. Визначити кількість днів у заданому місяці заданого року.
16. Визначити дату найближчого молодика, якщо відома дата повного місяця (молодик настає через 14 днів після повного місяця).
17. Визначити дату, віддалену від заданої на задане число днів.
18. Визначити дату, яка була один тиждень тому від введеної..
19. Визначити кількість днів у поточному місяці.
20. Визначити дату початку поточного тижня, якщо відома поточна дата та день тижня.

Порядок виконання роботи

1. Написати програму на вирішення завдання штатному наборі даних. Провести тестування. При тестуванні перевірити:

а) правильні значення вхідних даних;

б) неправильні значення вхідних даних;

в) критичні значення вхідних даних.

2. За результатами тестування виділити характеристики вхідних даних, котрим написана програма дає рішення. Внести зміни до програми.

3. Повинен бути перемикач мови інтерфейсу на мінімум три мови взаємодії: англійська, українська та російська. Причому програма має бути одна, а мови інтерфейсу мають бути в структурах даних. При захисті роботи повинні бути готові додати ще одну мову взаємодії.

4. При розробці програмної системи мають бути такі модулі:

i. модуль інтерфейсу користувача з ПС для кількох мов взаємодії (мінімум два плюс можливість додавання ще кількох):
   - введення вихідних даних;
   - виведення результату;
   - додавання мови не повинно впливати на програмну систему.

ii. Модуль розрахунку;

iii. Модуль тестування

   a) правильні дані:
   - усередині області;
   - на межі області:
   - перехід через тиждень;
   - перехід через місяць;
   - перехід через рік;
   - перехід через високосний рік;
   - перехід через століття;
   - перехід через тисячоліття.

   b) неправильні дані:
   - неправильна дата (день, місяць, рік);
   - неправильні символи.

Зміст звіту

1. Назва і мета лабораторної роботи.

2. Початковий варіант програми та опис вхідних даних, для яких вона виконується коректно та не коректно.

3. Остаточний варіант програми та опис вхідних даних.

4. Результати роботи програми (екранні форми).

5. Висновки

6. Перелік джерел посилань.

Контрольні питання

1. Що притаманно методики екстремального програмування?
2. У чому переваги методики екстремального програмування?
3. У чому недоліки методики екстремального програмування?

Лабораторна робота №3

РОЗРОБКА ПРОГРАМИ З ВИКОРИСТАННЯМ МЕТОДИКИ КОМПОНЕНТНОГО ПРОГРАМУВАННЯ

Мета роботи: набути практичних навичок у застосуванні методики компонентного програмування.

Завдання до лабораторної роботи

У середовищі візуального програмування створити програму для вирішення зазначеної у варіанті задачі та компонент з відповідним методом. Інтегрувати компонент у бібліотеку середовища розробки.

Варіанти завдань

1. Визначити кількість днів, що залишилися до кінця поточного місяця від заданої дати.
2. Визначити номер кварталу, до якого належить задана дата (1 квартал-січень-березень, 2 квартал-квітень-червень, 3 квартал-липень-вересень, 4 квартал-жовтень-грудень).
3. Визначити дату та назву дня, що передує заданому.
4. Визначити дату дня, віддаленого від заданого на 1 тиждень.
5. Визначити дату початку наступного тижня (відомі поточна дата та день тижня.)
6. Визначити дату найближчого повного місяця, коли відома дата молодика (повний місяць настає через 14 днів після молодика).
7. Визначити дату виходу людини з відпустки, якщо відома дата початку відпустки та її тривалість у днях.
8. Визначити кількість днів у заданому році.
9. Визначити дату, віддалену від заданої на задану кількість тижнів.
10. Визначити пору року, до якої належить введена дата.
11. Визначити дату та назву дня кінця поточного тижня (відомі поточна дата та день тижня).
12. Визначити номер тижня від початку місяця для заданої дати.
13. Визначити, на яку пізню дату можна придбати квиток на поїзд у заданий день (продаж квитків починається за 45 діб до дати відправлення).
14. Визначити дату та назву дня, наступного за заданим.
15. Визначити кількість днів у заданому місяці заданого року.
16. Визначити дату найближчого молодика, якщо відома дата повного місяця (молодик настає через 14 днів після повного місяця).
17. Визначити дату, віддалену від заданої на задане число днів.

18. Визначити дату, яка була один тиждень тому від введеної.

19. Визначити кількість днів у поточному місяці.

Визначити дату початку поточного тижня, якщо відома поточна дата та день тижня.

Порядок виконання роботи

1. Написати програму для вирішення задачі по варіанту.
2. Для інтерфейсу з користувачем використовувати компоненти, що надаються обраним середовищем розробки або Інтернет.
3. На базі компонента для введення вихідних даних створити свої компоненти з методом, що надає рішення задачі.
4. Інтегрувати розроблені компоненти у бібліотеку середовища розробки.
5. Створити програму, що ілюструє роботу з компонентом, включеним до бібліотеки.

Зміст звіту

1. Назва і мета лабораторної роботи.
2. Текст програми на вирішення завдання з виділенням компонентів середовища розробки.
3. Опис розробленого компонента та послідовності дій щодо його інтеграції до бібліотеки.
4. Текст програми, що ілюструє роботу з компонентом, включеним до бібліотеки.

Контрольні питання

1. У чому суть компонентного програмування?
2. Які сучасні середовища програмування підтримують компонентний підхід?
3. Якою є типова структура компонента?

Лабораторна робота №4

РОЗРОБКА ПРОЕКТУ ПРОГРАМИ З МЕТОДИКИ SCRUM

Мета роботи: набути практичних навичок у застосуванні методики Scrum.

Теоретичні положення

Scrum – це система управління проектами, заснована на принципах Agile. Вона призначена для команд із десяти і менше людей. Вони поділяють роботу на невеликі завдання, які потрібно зробити протягом певного часу. Як і в Agile, ці часові інтервали називаються "спринти". Scrum сприяє самоорганізації команди: вони вирішують складні завдання, аналізують перемоги та поразки та покращують роботу.

Принципи Scrum:

1. *Прозорість*. Команда має працювати у середовищі, де кожен знає, з якими проблемами стикаються інші члени команди.

2. *Перевірка*. Часті перевірки вбудовані у структуру процесу, щоб дати команді можливість обміркувати, як процес працює. До таких пунктів перевірки належать:

• Daily Scrum Meeting та

• Sprint Review Meeting.

3. *Адаптація*. Команда постійно досліджує актуальність продуктів і переглядає пункти, що не мають сенсу.

Команда Scrum:

Product Owner. Це замовник чи призначений їм відповідальний професіонал. Його завдання – розуміти вимоги бізнесу, клієнта та ринку:

• Написання та підтримка Product Backlog.

• Тісна співпраця із замовником та командою Scrum, повідомляючи кожному учаснику значення робочих завдань у Product Backlog.

• Надає команді зрозумілі вимоги, які можна протестувати.

• Відповідає за приймання наприкінці кожної ітерації.

Scrum Master – стежить за застосуванням принципів Scrum у своїх командах. Він навчає команду та замовника Scrum-процесу та намагається оптимізувати застосування цієї практики.

Успіх Scrum-майстра залежить від того, наскільки добре він розуміється на роботі, яку виконує команда, і може допомогти команді підвищити прозорість роботи та оптимізувати процес. Це головний координатор, який складає перелік необхідних ресурсів (кадрових та матеріально-технічних) для зборів із планування спринту та огляду його підсумків, стендапу та ретроспективи спринту.

Development Team – команда, що включає від 3 до 9 максимально кваліфікованих професіоналів, які вміють робити всі етапи розробки ПЗ. Вона самостійна, сама організована та максимально мотивована на результат.

Модель розробки програмного забезпечення Scrum побудована таким чином, щоб допомогти командам природним чином адаптуватися до мінливих умов ринку та потреб користувачів. Водночас короткі цикли дозволяють розробникам бути ефективнішими. Scrum забезпечує структуру, оптимізує розробку і залишається гнучким і враховує бажання власника продукту.

По Scrum команда працює наступним чином (рис. 8).

Плюси

• Легко адаптувати до нестабільних вимог проекту, підходить для динамічного середовища.

• Продукт відповідає потребам власника завдяки його участі у проекті.

• Процес роботи стає прозорим за рахунок регулярних зустрічей та видимого прогресу проекту. Це також полегшує спілкування між учасниками.

• Часті тестування та аналіз допомагають виявити та усунути помилки на ранніх етапах розробки.

Рисунок 8 - Робота команди по Scrum

Мінуси

• Методологія підходить лише для згуртованих досвідчених команд, тому що вона має на увазі постійну взаємодію всіх членів команди.

• Може не підходити для проектів із фіксованими термінами чи бюджетом, оскільки підхід орієнтований на гнучкість та адаптивність.

• Може бути складною для впровадження та вимагати від команд проходження навчання, особливо під час переходу з водоспадної моделі.

• Менеджери, які використовують Scrum, можуть зіткнутися з труднощами під час роботи з дуже великими та складними проектами.

Кому підходить

✓ Scrum підходить для проектів, де потрібні гнучкість та спільна робота. Методологія добре показує себе, якщо вимоги та терміни змінюються або якщо на ринку жорстка конкуренція. Якщо власника продукту зацікавлений у процесі розробки і бере активну участь у ньому - наприклад, дає фідбек на кожному з етапів - такий проект отримає вигоду від клієнто-орієнтованості Scrum.

✗ Проте Scrum може не підійти для проектів, які вимагають суворого дотримання нормативних вимог, та проектів, у яких неможливо поставити навіть короткі тижневі цілі на спринт.

Він також не підійде для проектів без чіткої ідеї та налагодженого конвеєра, а ще якщо у команді є конфлікти та інші проблеми.

Одна чи кілька крос-функціональних само організованих команд створюють продукт інкрементами, тобто поетапно. У команді може бути близько семи осіб.

Scrum - це не лінійний метод розробки; це каскадна модель.

Scrum-продукт розробляють не відразу повністю, а невеликими готовими до релізу частинами, кожну з яких завершують за коротку ітерацію, або спринт.

Структура Scrum працює наступним чином (рис.9).

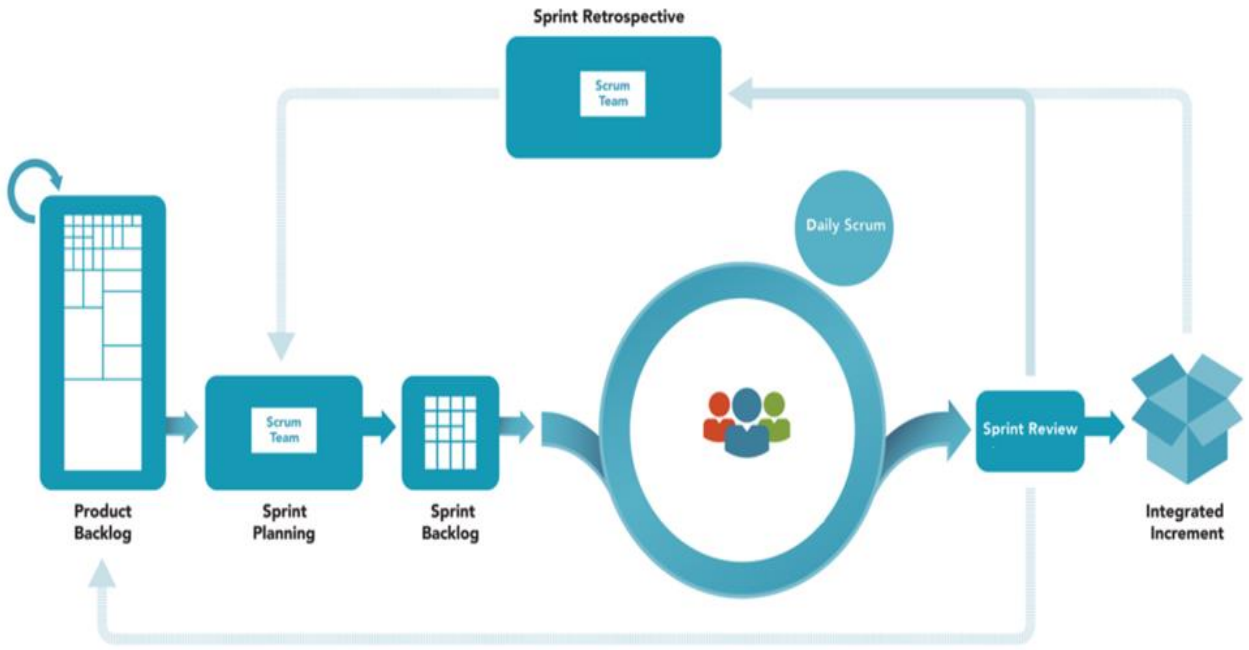

Рисунок 9 - Процес розробки по Scrum

Чотири головні переваги.

1. Відгук потреби клієнта. Компаніям-розробникам дуже добре знайома вимога «розробити на вчора». Традиційні організації, які працюють за методом водоспаду, вбудовують важливі опції та функції у розклад із двома релізами на рік — і в процесі нерідко втрачають клієнтів. Якщо ж клієнти не йдуть, вони можуть залишитися незадоволеними і піти з часом, коли зустрінуть більш відповідального конкурента. Працюючи короткими та частими циклами, ви можете надавати клієнтам продукти майже на вимогу та швидко адаптуватися до нових вимог.

2. Зниження вартості розробки. Аджайл і Scrum довели свою економічність та ефективність. У цих моделях ролі розробників різноманітніші, адже невеликі одиниці можна ефективно тестувати тією самою командою, яка їх розробила. Спеціалізація зникає чи скорочується, зрештою скорочуючи витрати.

3. Насолода від роботи. Поставляючи продукти швидко, команда переживає додаткову радість щоразу, коли робота зроблена та вирушає у світ. Кожна Scrum - команда знає, наскільки приємний реліз. Зі Scrum команда радіє йому не два, а як мінімум дванадцять разів на рік.

4. Більше швидких доходів. Кошти від клієнтів надходять не двічі на рік, а набагато частіше. Нові опції теж можна додавати не двічі на рік, а набагато частіше, таким чином наводячи більше клієнтів та швидше обробляючи їх особливі запити.

Ролі в Scrum

У Scrum є три ролі, які разом утворюють Scrum -команду:

✓ **Власник продукту** – апологет продукту, який повністю розуміє його цінність для бізнесу. Ця людина доносить потреби замовника/стейкхолдера до розробників, але не відповідає за технічний бік процесу. Власник продукту також відповідає за користувальницькі історії та визначає їхню пріоритетність.

Стейкхолдери (*stakeholder*) – це особи, які мають інтереси щодо проекту чи організації або впливають на проект чи організацію. Стейкхолдери бувають внутрішні (співробітники та керівництво) та зовнішні (клієнти, постачальники, громадськість).

✓ **Розробники** виконують усі технічні завдання щодо розробки. Вони крос-функціональні, їхня крос-функціональність залежить від галузі роботи. Тим не менш, розробники завжди відповідають за створення беклог спринту, забезпечують якості згідно специфікації продукту, адаптацію плану відповідно до мети спринту і за власні завдання зі своєї галузі відповідальності.

✓ **Scrum -майстер** виступає організатором роботи Scrum -команди. Scrum -майстер допомагає власнику продукту та розробникам виконувати роботу без перешкод та відволікаючих факторів. Вся комунікація людей поза командою з розробниками проходить через Scrum -майстри. (Іноді Scrum -команди взаємодіють у форматі Scrum of Scrum (скраму скрамів), тобто зборів Scrum -майстрів усіх команд).

Події у Scrum

Є п'ять типів Scrum –подій.

1. **Спринт** (*Sprint*) - саме серце Scrum. Усередині спринту виконується вся робота, необхідна досягнення мети продукту, зокрема планування спринту, дейлі Scrum, огляд і ретроспектива спринту.

Це період часу, протягом якого команда Scrum спільно працює над створенням готового інкременту. Як правило, спринт триває один - два тижні, обсяг спринту в один тиждень вибирається, якщо необхідна швидкість розробки.

Рекомендується планувати спринт тим коротше, що складніше робота і що більше у ній невідомих.

Але останнє слово завжди за командою. Не треба соромитися змінювати тривалість спринту, якщо він вам не підходить. Протягом цього періоду власник продукту та команда розробників можуть переглянути обсяг спринту, якщо це необхідно. Це і є емпірична суть Scrum.

Усі заходи щодо планування до ретроспективи проводяться протягом спринту. Після того, як тимчасовий проміжок для спринту визначений, він повинен залишатися незмінним, допоки ведеться розробка.

2. **Планування спринту** (*Sprint Planning*) - у ньому беруть участь усі члени Scrum -команди протягом тривають максимум 8 годин. На ньому команда розробників під керівництвом Scrum-майстра планує роботу (обсяг спринту), яку

необхідно виконати протягом поточного спринту. Також кожен член команди може висловитися про те, що його цікавить чи турбує. У ході зустрічі призначаються пріоритети та проводяться оцінки термінів

На ньому визначається мета спринту. Потім до Sprint Backlog додаються конкретні User Story з Product Backlog. Ці історії завжди співвідносяться з цілями. При цьому команда Scrum узгоджує такі історії, які можна реалізувати практично під час спринту.

Наприкінці зборів кожен член команди Scrum має чітко уявляти, які завдання можна виконати за спринт та як поставити інкремент.

3. **Щоденний Scrum** (*Daily Scrum meeting*) - це дуже короткі щоденні збори, які для зручності проводяться в один і той же час (зазвичай вранці) і в тому ж місці. Вони короткі (до 15 хвилин) та призначені для того, щоб спланувати денний розклад розробників. Тут можна обговорити робочі складності або прояснити користувальницькі історії. Зустріч є обов'язковою для розробників у повному складі. Scrum -майстер може, але не повинен бути присутнім на ній.

Щоденна Scrum-нарада проводиться, щоб кожен учасник команди був у курсі того, що відбувається, не відхилявся від мети та отримував план роботи на найближчі 24 години. На цих зборах кожен учасник команди відповідає на запитання: що він зробив учора, що він планує робити сьогодні та які у нього виникли проблеми.

4. **Огляд спринту** (*Sprint Review*) – демонстрація діючого продукту, розробленого під час спринту. Цей захід проходить наприкінці спринту і призначений насамперед для того, щоб у подробицях показати досягнутий стейкхолдерам. Команда розробників представляє зацікавленим сторонам та колегам завершені робочі завдання з беклогу, щоб зібрати відгуки. Власник продукту вирішує, чи слід випускати інкремент.

Під час наради ***Product Owner*** змінює ***Product Backlog***, виходячи з результатів останнього завершеного спринту. Цей процес може перейти у планування наступного спринту. Нарада триває 3-4 години.

5. **Ретроспектива спринту** (*Sprint Retrospective*)– проводиться, щоб команда зафіксувала та обговорила всі успіхи та невдачі спринту, проекту, учасників, їхніх взаємин та інструментів. Мета ретроспективи — створити умови, щоб команда могла оцінити все, що вдалося і потрібно покращити наступного разу, та не зациклювалася на невдачах. Триває 1-3 години.

Додатково до цих типів заходів іноді під час спринту команди можуть проводити **уточнення беклогу** (*Backlog Refinement*) – обговорювати елементи беклогу та готуватись до наступного спринту. У рамках цієї зустрічі можна обговорити пріоритетність елементів та розділити елементи беклогу на дрібніші складові.

Артефакти Scrum

Артефакти Scrum є роботою, яку потрібно виконати, щоб завершити проект або спринт. Вони роблять інформацію про проект прозорим для всіх учасників.

У Scrum існує три артефакти, які пстійні присутні у кожній команді Scrum.Вони необхідні, щоб постачати програмне забезпечення, яке буде цінним для замовників. Є й необов'язкові артефакти, які можуть полегшити команду життя.

1. **Беклог продукту** – це головний перелік завдань, які ще називають User Story, які потрібно виконати. Його веде Замовник чи Product Owner. Це перелік функціональних можливостей, вимог, поліпшень і виправлень, що постійно змінюється, що складається із завдання для беклога спринту. Загалом це список завдань команди. Власник продукту постійно звертається до беклогу продукту, змінює у ньому пріоритети та підтримує його актуальність, оскільки може з'явитися нова інформація чи можуть статися зміни над ринком, тому зникне сенс виконувати ці завдання чи з'являться нові способи вирішення проблем.

2. **Беклог спринту** – це список робочих завдань, історій або виправлень багів, відібраних командою розробників із User Story, написаних у Product Backlog. Перед кожним спринтом проводиться збори з планування спринту (його ми обговоримо далі у статті), у якому команда вибирає, які завдання з бэклогу продукту необхідно виконати у межах спринту. Беклог спринту може бути фіксованим і може змінюватися протягом спринту. Однак ніщо не повинно заважати досягненню основної мети спринту - того, чого команда хоче досягти за поточний спринт.

3. **Increment** – це готовий для використання кінцевий продукт за підсумками спринту. Інкремент презентується на демонстрації наприкінці спринту, де команда показує, що вона зробила спринтом. Його часто визначають як критерій готовності продукту, контрольну точку, мету спринту чи навіть повну версію чи поставлений епік. Все залежить від того, якими критеріями готовності керується ваша команда та як вибираються цілі спринту. Наприклад, деякі команди вважають за краще випускати щось для своїх клієнтів наприкінці кожного спринту. Їхнє слово «готово» означає «поставлене». Однак для інших команд це може бути непрактичним. Уявіть, що ви працюєте над серверним продуктом, який можна постачати клієнтам раз на три місяці. Ви, як і раніше, можете розбивати роботу на двотижневі спринти, але ваш продукт буде готовий, коли ви завершите роботу над частиною більшої версії, яку плануєте поставити повністю. Однак не забуватимемо, що чим більше часу йде на випуск ПЗ, тим менше шансів у цього ПЗ досягти успіху.

- ✓ ***Елемент беклогу продукту*** – його потрібно виконати за ітерацію спринту. Зазвичай розбивається кілька менших подзадач.
- ✓ **Мета спринту** - те, що потрібно зробити, щоб виконати елемент беклог продукту.
- ✓ **Берн-даун чат спринту** - робота, яка залишається до повного виконання завдань спринту. Берн-даун чат може бути висхідним або низхідним залежно від того, з чим команда стикається під час виконання завдання. Він служить не звітом про просування команди, а шляхом подолання труднощів і підтримки активності.
- ✓ **Реліз продукту/берн-даун чат продукту** – його наприкінці кожного спринту оновлює Scrum -майстер. Горизонтальна вісь відповідає спринтам, вертикальна показує, скільки роботи (на початку кожного спринту) залишилося до завершення проекту.

Правила Scrum

Ролі, події та артефакти – основні правила Scrum, але є й інші, які також додають процесу ефективності:

- ✓ Scrum -команда складається тільки з власника продукту, Scrum -майстра та розробників.
- ✓ Усі спринти мають бути однаковими за довжиною.
- ✓ Коли завершується один спринт, наступний починається негайно.
- ✓ Кожен спринт починається з планування спринту. Команда розробки щоранку проводить щоденний скрам.
- ✓ Кожному спринті проводять огляд спринту, щоб стейкхолдери могли надати зворотний зв'язок.
- ✓ Доповнювати беклог спринту під час спринту – не найкраща практика.

Обмеження у Scrum

- ✓ Scrum часто призводить до скорочення обсягу робіт через відсутність загального дедлайну (крайній термін виконання завдання чи роботи, певний момент часу, до якого має бути досягнуто мети чи завдання.
- ✓ Високі шанси на провал проекту, якщо люди не дуже залучені чи не готові співпрацювати.
- ✓ Застосовувати структуру Scrum у великих командах складно, але можливо.
- ✓ Якщо будь-який член команди піде в середині проекту, ц е може мати сильний негативний вплив на проект.

Завдання до лабораторної роботи

Сформувати опис проекту розробки програми відповідно до варіанта завдання.

Варіанти завдань

1. Створити інформаційну систему «Бібліотека», яка виконує функції пошуку книги в каталозі, автоматизацію роботи бібліотекаря (видача/прийом книги, довідка про видання книги, наявність книги у певному відділі).
2. Створити навчальну систему з можливістю тестування знань учня.
3. Створити інформаційну систему «Лікарня», яка виконує функції видачі довідки про графік прийому лікарів, про хворих, які зверталися до лікарні, про процедури, призначені хворим.
4. Створити інформаційну систему «Дипломування», що дозволяє отримати інформацію про викладачів, студентів-дипломників, теми дипломних проектів,
5. на те, що сучасні банки роблять своїм клієнтам широкий спектр послуг, починаючи від обслуговування рецензентів, оцінки
6. Створити функціональну модель діяльності банку з огляду рахунків, прийняття внесків, кредитування й закінчуючи роботою на ринку цінних паперів, роботою з інвестиціями, валютними операціями, та інші можливі напрямки діяльності.

7. Створити інформаційну систему «Книгарня», що виконує функції довідкової системи про наявність книг у магазині і дозволяє зробити покупку через Internet.

8. Створити функціональну модель діяльності великого автосалону, зважаючи на те, що автосалон робить послуги з гарантійного обслуговування клієнтів, має власну автомайстерню, працює безпосередньо з виробниками машин, з клієнтами, робить послуги з оформлення документів.

9. Створити функціональну модель роботи банкомату з огляду на всі можливі транзакції й забезпечення обслуговування самого апарата – завантаження грошей, перевірка справності програмного забезпечення, забезпечення захисту від зловмисників

10. Створити інформаційну систему «Кінопрокат», що дозволяє отримати інформацію про прокат фільмів у кінотеатрах, ціни та наявність квитків.

11. Створити функціональну модель процесу роботи пункту налагодження електронної техніки з огляду на всі можливі види послуг, забезпечення роботи майстрів, закупівлю деталей, роботу з клієнтами.

12. Створити функціональну модель процесу роботи кадрового агентства, з огляду на різні методи підбору персоналу - через Інтернет, за оголошеннями, робота з вузами й т. д. і пошуку клієнтів. Описати роботу агентства.

13. Створити інформаційну систему про наявність ліків в аптеках міста, що дозволяє зробити замовлення на ліки.

14. Створити функціональну модель діяльності бухгалтерії промислового підприємства. Бухгалтерія обробляє рахунки-фактури від постачальників, клієнтів, нараховує заробітну плату співробітникам, обробляє інформацію з контрактів, працює з податковими органами й соціальними фондами.

15. Створити інформаційну систему «Готель», що дозволяє отримати відомості про наявність вільних номерів, ціни, забронювати номер.

16. Створити функціональну модель процесу тестування програмного забезпечення. Процес перевірки з використанням автоматичного тестування. Відобразити можливі сценарії повторної перевірки та повернення програмного забезпечення для виправлення знайдених помилок.

17. Створити інформаційну систему «Страхові компанії України», що дозволяє отримати довідку про послуги страхування, ціни на послуги різних компаній, відомості про укладені договори страхування.

18. Створити функціональну модель діяльності великого салону з продажу холодильного обладнання, зважаючи на те, що салон робить послуги з гарантійного обслуговування клієнтів, має власну майстерню, працює безпосередньо з виробниками машин, з клієнтами, робить послуги з оформлення документів.

19. Створити інформаційну систему «Екологічна служба», що дозволяє отримати відомості про підприємства, що сталися на них катастрофах, завданих збитків.

20. Створити функціональну модель діяльності комп'ютерної фірми, з огляду на те, що фірма торгує комп'ютерами в зібраному вигляді й комплектуючими. Фірма працює як з виробниками комп'ютерної техніки, так і з клієнтами. Фірма робить ряд

додаткових послуг: установка програмного забезпечення, підключення до мережі Інтернет, гарантійне обслуговування й т.д.

21. Створити сайт туристичної агенції, що дозволяє ознайомитися з пропонованими турами, забронювати тур або авіаквитки.

22. Створити функціональну модель діяльності торговельної фірми по реалізації косметичної продукції з огляду на роботу фірми із клієнтами, постачальниками, доставку косметики від постачальників і по торговельних місцях клієнтів.

23. Створити інформаційну систему «Музеї світу», що дозволяє отримати відомості про музеї, експонати та їх участь у виставках.

24. Створити функціональну модель діяльності торговельної фірми по реалізації продовольчої продукції з огляду на роботу фірми із клієнтами, постачальниками, доставку продукції від постачальників і по торговельних місцях клієнтів.

25. Створити функціональну модель діяльності кафедри вищого учбового закладу з огляду на наступні напрямки: робота із забезпечення навчального процесу, робота за господарськими договорами, науково-дослідницька робота співробітників і студентів і т.д.

26. Створити функціональну модель роботи аеропорту з огляду на роботу аеропорту з авіакомпаніями, клієнтами, постачальниками й т.д. Урахувати різноманітні роботи аеропорту з технічного обслуговування літаків, обслуговування клієнтів через каси, роботу диспетчерської служби аеропорту.

27. Створити сайт фотостудії, що дозволяє ознайомитись з послугами, розцінками, зразками робіт, зробити замовлення.

28. Створити функціональну модель роботи будівельної фірми. Описати роботу фірми як з постачальниками, так і з клієнтами. Треба відзначити, що в цей час будівельні організації забезпечують повний технологічний процес, починаючи проведення досліджень ринку, створення проекту, закупівлі матеріалів, до безпосереднього будівництва й остаточного продажу квартир.

29. Створити функціональну модель діяльності вищого учбового закладу з огляду на його роботу як по основних напрямках діяльності: забезпечення навчального процесу, наукової праці, так і по додаткових процесах: міжнародна діяльність, робота за договорами, соціальна робота.

Зміст звіту

1. Назва і мета лабораторної роботи.
2. Беклог проекту.
3. Беклоги спринтів.
4. Діаграма згоряння завдань.

Контрольні питання

1. У чому полягає суть методики Scrum?
2. Що таке спринт, беклог?
3. Для чого використовується діаграма згоряння задач?
4. У чому суть планування покеру?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Відмінності та переваги Agile, Scrum та Kanban в управлінні проєктами [Електронний ресурс] : сайт. – Режим доступу: https://online.novaposhta.education/blog/vidminnosti-ta-perevagi-agile-scrum-ta-kanban-v-upravlinni-proektami. – Назва з екрана.

2. Скрам [Електронний ресурс] // Вікіпедія: вільна енциклопедія. - Режим доступу: https://uk.wikipedia.org/wiki/Скрам. – Назва з екрана.

3. Скрам. Що це таке та як цим користуватися [Електронний ресурс] // Brander. - Режим доступу: https://brander.ua/blog/skram-shcho-tse-take-ta-yak-tsym-korystuvatysya. – Назва з екрана.

4. Що таке методологія Scrum і коли варто її використовувати [Електронний ресурс] // Softserve. - Режим доступу: https://career.softserveinc.com/uk-ua/stories/what-is-scrum-methodology. – Назва з екрана.