Додаток 1

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи №2 з дисципліни

«Компоненти інженерії програмного забезпечення.
Вступ у програмну інженерію»

«Дослідження процесу розробки програмного забезпечення. Специфікування вимог»

Варіант 2

Виконали студенти групи ІП-02 Гущін Д.О., Білько О.Є., Демченко О.С.

Перевірила Вітковська І.І.

Київ 2021

Лабораторна робота 2

Мета: отримати навички специфікування вимог до програмного забезпечення.

Завдання:

1. Сформувати робочу групу (3-5 чоловік), розподілити ролі.
2. За узгодженням з викладачем обрати варіант завдання (табл..1) для виконання лабораторних робіт.
3. Провести попередній аналіз предметної області, визначити функції ПЗ, що проектується.
4. Побудувати діаграму прецедентів на основі проведеного попереднього аналізу.
5. Специфікувати вимоги.
6. Обрати та обґрунтувати обрану модель життєвого циклу.
7. Підготувати захист у вигляді мітингу із замовником.

2	Інформаційна система бібліотеки	дозволяє шукати книги у своєму каталозі, враховувати видачу книг на руки й повернення книг.

Хід роботи

Опис предметної області

Інформаційна система бібліотеки – взаємопов’язана система для видачі друкованих інформаційних джерел: книжки, журнали, підручники.

Бібліотека пов’язана з комп’ютером бібліотеки, а той в свою чергу з базою даних, де зберігається інформація про місце, де зберігається книжка та чи є вона зайнята, чи ні. Інформаційну систему можна визначити як сукупність інформаційних елементів введення, обробки, переробки, зберігання, пошуку, виводу й поширення інформації, що перебувають у відносинах і зв'язках між собою та складають певну цілісність.

Глосарій

Бібліотека – місце збереження книжок, в якому клієнт може отримати або віддати вже запозичену книгу.
Книга – джерело інформації, що має номер видання, автора та назву.
Клієнт – користувач однієї або декількох бібліотек.
Інтерфейс – це всі текстові поля, кнопки, смуги прокрутки, з якими взаємодіє користувач.
Інформаційна система - як сукупність інформаційних елементів введення, обробки, переробки, зберігання, пошуку, виводу й поширення інформації, що

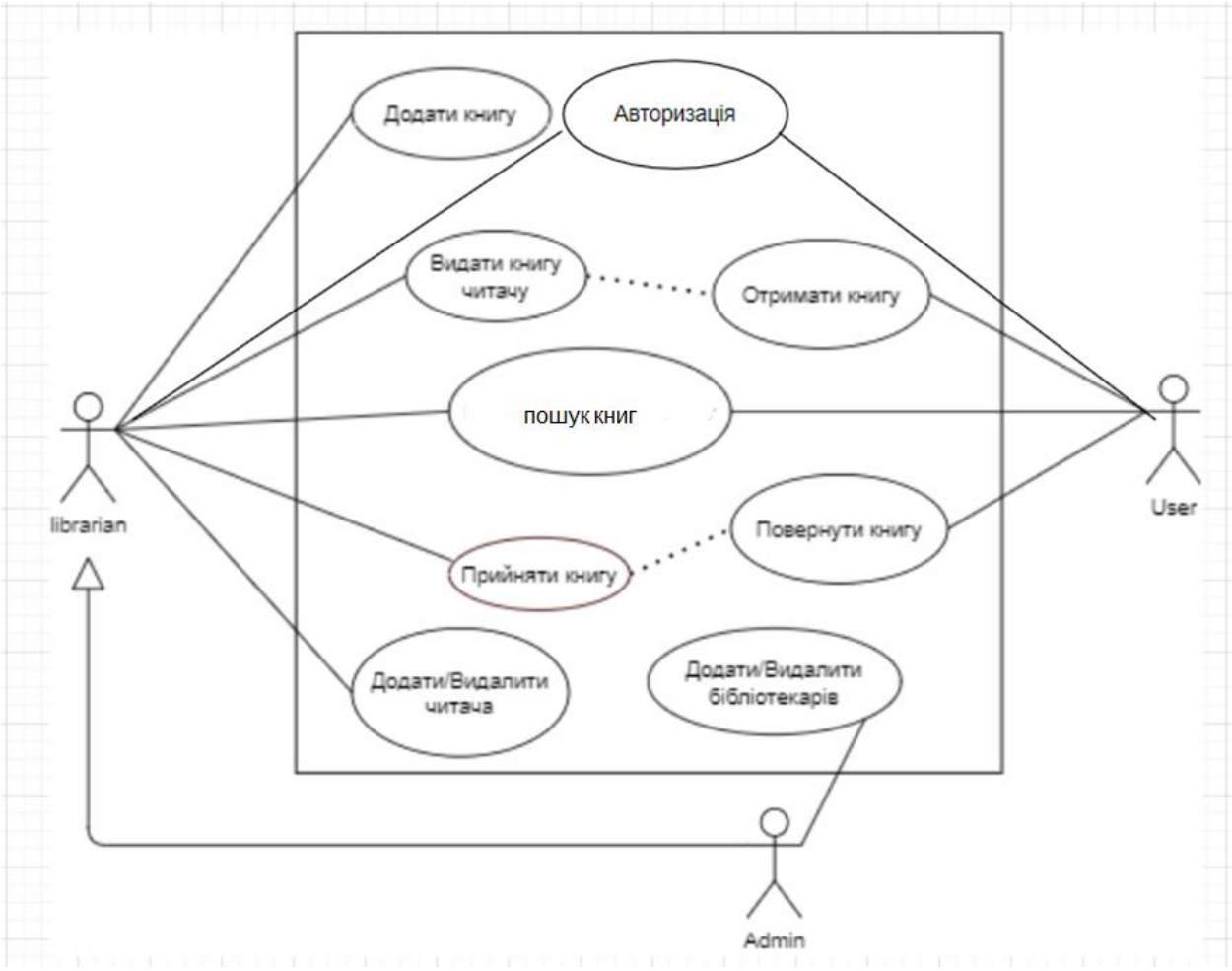
перебувають у відносинах і зв'язках між собою та складають певну цілісність.

База даних — це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів.

СБ – система бібліотеки

Авторизація — процес, в якому користувач «представляється» системі, як правило, вводячи свої особисті дані (логін і пароль)

Use-Case Diagram



	Авторизація користувача
Description	Авторизація користувача за допомогою логіну та паролю
Actor	Користувач, Інтерфейс СБ
Pre-conditions	Останній вхід було здійснено більше години тому
Main flow	1. Система пропонує введення даних для входу (логіну та паролю) користувачеві 2. Користувач вводить дані у відповідні поля

	3. Система перевіряє коректність введених даних 4. Система повідомляє користувача про успішний вхід
Alternative flow	1. а. Користувач відмовляється вводити дані 1. Виведення попередження про неможливість використання программи без авторизації 3. а. Введено некоректні дані 1. Виведення попередження про неправильність введених даних
Post-conditions	Користувач здійснив вхід у систему Користувача було повідомлено

	Отримати список книжок
Description	Отримання інфорції з бази даних
Actor	Користувач, база даних
Pre-conditions	Виконано “Увійти в систему”
Main flow	1. Виконується зчитування списку книг 2. Дані передано у систему
Alternative flow	-
Post-conditions	Дані зчитано і передано у відповідні функції

	Перевірка користувача
Description	Перевірка на кількість вже взятих книжок
Actor	Інтерфейс СБ, користувач
Pre-conditions	Користувач є авторизованим
Main flow	1. Перевірка успішна. Користувач отримав та не повернув менше 10 книг.
Alternative flow	1. Перевірка неуспішна. Користувач отримав та не повернув більше 10 книг.

Post-conditions	Користувач перевірений на кількість взятих книг

	Видача книги
Description	Видача книги корисувачеві
Actor	Користувач, бібліотекар
Pre-conditions	Користувач авторизований та перевірений на кількість взятих книг
Main flow	1. Бібліотекар видає книгу 2. Користувач отримує книгу 3. Бібліотекар фіксує зміни в системі
Alternative flow	-
Post-conditions	Користувач має повернути книгу у визначений час.

	Повернення книги
Description	Повернення книги користувачем
Actor	Користувач, бібліотекар
Pre-conditions	Користувач отримав книгу та є авторизованим
Main flow	1. Користувач повертає книгу у визначений час 2. Бібліотекар фіксує зміни в системі
Alternative flow	1.a. Користувач не повертає книгу у визначений час 1. Користувач сплачує штраф за заборгованість
Post-conditions	Користувач має повернути на одну книгу менше

Специфікація вимог

Функціональні вимоги:

-СБ має перевіряти дійсність логіну і паролю
-СБ має відображати список доступних книг
-СБ має дозволяти користувачу взяти і повернути книжку

Нефункціональні вимоги:

-СБ має бути написана C++
-СБ має перевіряти логін і пароль користувача не більше ніж за 1 секунду
-СБ повинна обмінюватися інформацією з базою даних, використовуючи 256-розрядну кодування

Висновок

Виконавши цю лабораторну роботу ми отримали навички специфікування вимог до програмного забезпечення**,** провели попередній аналіз предметної області, визначили функції та специфікували вимоги до ПЗ, яке проектували, побудувати діаграму прецедентів на основі проведеного попереднього аналізу, обрали найбільш доречну модель життєвого циклу і обґрунтували наш вибір.

Контрольні запитання

1. **Покажіть розвиток інженерії і зокрема інженерії програмного забезпечення**

Періоди розвитку Об'єкти порівняння	1960±5 років програмування *"any-wich-way"*	1970±5 років «програмування в малому»	1980±5 ро-ків «програму-вання у ве-ликому»
Об'єкти	Маленькі програми	Алгоритми і програми	Системна структура
Дані	Неструктурована інформація	Структури даних і типи	Бази даних
Управління	Елементарне ро-зуміння діаграм управління	Програми виконуються і закін-чуються	Програми, що безпере-рвно вико-нуються
Простір етанів	Стан, що погано розуміється окре-мо від управління	Маленькі, прості	Великі, структуризовані
Організаційне управління	Немає	Індивідуальні зусилля	Колективні зусилля, супровід
Інструмент»	Асемблери	Компілятори, редактори, заван-тажувачі	Середовища, інтегровані інстру-менти

2. Наведіть модель Code and fix та вкажіть недоліки

Описати правила цього методу просто:

- отримуємо початкове розуміння потреб замовника;
- починаємо програмувати;
- коли щось буде готово, показуємо «це» замовнику;
- отримавши відгуки, виправляємо наш код;
- повторюємо цикл до повного задоволення замовника (або поки у нього не закінчаться гроші, терпіння ...)

Використовуючи Code-and-Fix, ми тільки пишемо код. Тут все дуже просто: немає необхідності що-небудь планувати, немає необхідності що-небудь документувати. Тому Code-and-Fix вимагає мінімальної кваліфікації розробників, відповідно, їм можна платити менше грошей.

Але, у всього є своя ціна. Як показав досвід, такий підхід дуже скоро приводить до коду, який неможливо підтримувати: виправлення однієї помилки призводить до появи декількох нових; внесення мінімальних змін в одну з частин програми, призводить до руйнування функцій, реалізованих іншими частинами.

3. Поясніть, в чому системо утворююча роль життєвого циклу программного забезпечення в інженерії

Життєвий цикл ПЗ - це стадії, які проходить програмний продукт від появи ідеї до її реалізації в коді, імплементації в бізнес і подальшої підтримки. Моделі життєвого циклу багато в чому зумовлюють і методології розробки ПЗ.

4. Наведіть Step wise модель, поясніть роль її в інженерії програмного забезпечення та вкажіть недоліки

Поступове вдосконалення - це концептуальна основа для поступового розвитку артефакту. Відповідно до поетапного вдосконалення розвиток відбувається в трьох вимірах: від абстрактного до конкретного, від часткового до повного та від неструктурованого до структурованого. Таким чином, розвиток артефакту можна охарактеризувати як змішану послідовність доопрацювань, розширень та реструктуризацій артефакту.

5. Основні складові фаз життєвого циклу, дайте визначення, наведіть приклади

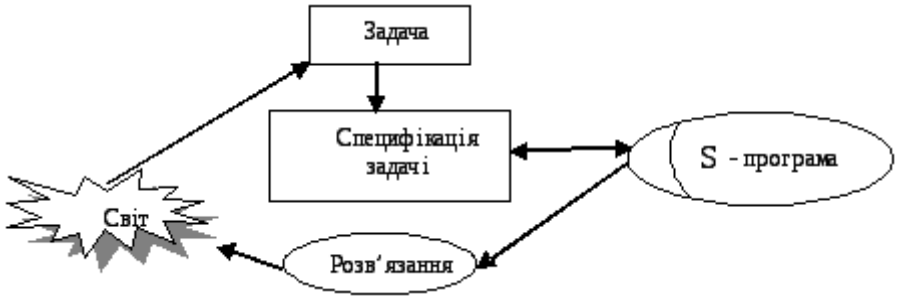
Кожна фаза містить три складових:
− процеси;
− продукти;
− ресурси.

На кожній фазі відбувається певна сукупність процесів, кожен з яких породжує певний продукт, використовуючи певні ресурси.

6. Типи програм за Леманом, загальні визначення

Існує підхід запропонований М. Леманом згідно з яким усі комп'ютерні програми можна поділити на три типи: S (Specification), P (Problem) і E (Environment).

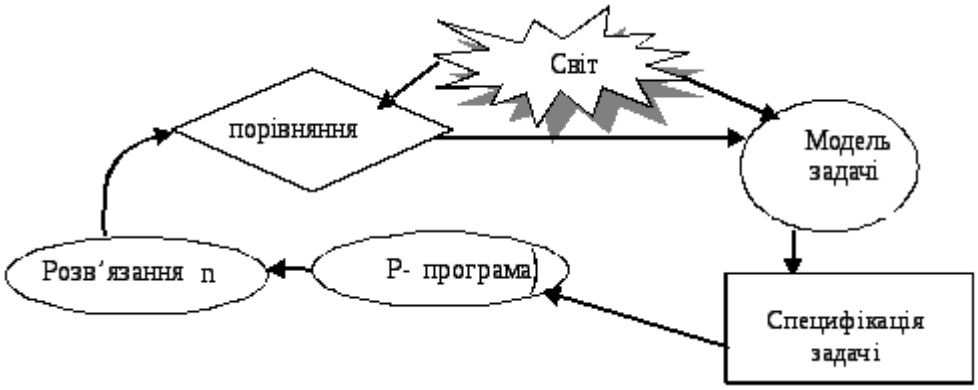
***S*-програма** – це така програма, функція якої відома й визначена однозначно специфікацією задачі. Для *S*-програм характерна повна визначеність вихідної задачі, вимог і значень, а тому S-програми після створення не змінюються. А якщо *S*-програма змінюється, то зміни не повинні порушити відповідності вхід/вихід, оскільки інакше вона розв'язуватиме іншу задачу, і це буде інша програма.



Місце S-програми в реальному світі

***P*-програма** – це така програма яка розв'язує задачу, що не має точної постановки. Тому специфікація задачі та розв'язання наближені, уособлюючи абстрактну модель реальної ситуації, і після порівняння з вимогами реального світу уточнюватимуться через зміну програми. Проте це буде не нова, а стара програма.

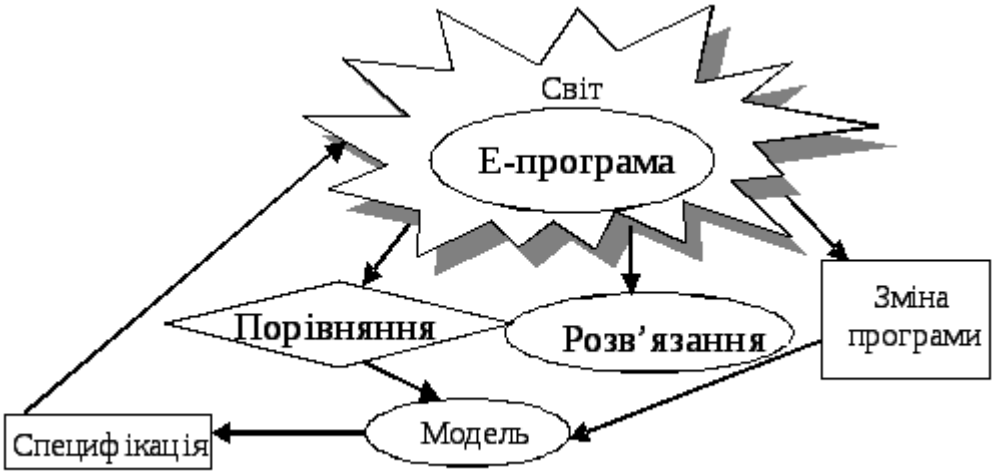
Прикладом є програма що обчислює прогноз погоди. У такій програмі результати розрахунків порівнюються з реальними даними про погоду, а методи прогнозування вдосконалюються зі змінною самої програми.



Місце *P*-програми в реальному світі

***E*-програма** – це програма, яка розв'язує таку задачу, що потребує її присутності в контексті реального світу. У процесі використання *E*-програми в реальному світі становлення до неї зазвичай змінюється і постає потреба змінити програму. При цьому змінена *E*-програма, так само як і *P*-програма,

не буде новою програмою. Прикладом такої програми являється програма керування тренажером для реального об’єкту.



Місце *E*-програми в реальному світі

Зазначені відмінності між типами програм є принциповими й можуть впливати на стосунки між замовником програми і виконавцем – розробником, вибір типу життєвого циклу або обсяг фінансування. Наприклад, формуючу стосунки із замовником розробник залежно від типу програми (у разі *P*- і *E*-програм) має передбачити у проекті неодмінні зміни програм.

Зазвичай P і E програми називають *програмами-застосуваннями*, або *комп’ютерними застосуваннями*, або *програмними системами.*

7. Комп’ютерні програми як системи

Програмна продукція, яка являє собою сукупність програм і (або) підсистем, що мають загальне цільове призначення. Зв'язок між програмами і (або) підсистемами встановлюється розробником, користувачем або іншими фахівцями

8. Типи програмних систем

На сьогоднішній день можна сказати, що більш-менш точно склалися такі групи програмного забезпечення:

- операційні системи та оболонки;
- системи програмування (транслятори, бібліотеки підпрограм, відлагоджувачі тощо);
- інструментальні системи;
- інтегровані пакети програм;
- динамічні електронні таблиці;
- системи машинної графіки;
- системи управління базами даних (СУБД);
- прикладне програмне забезпечення.

Зрозуміло, цю класифікацію не можна вважати вичерпною, але вона більш-менш наочно відображає напрями удосконалення та розвитку програмного забезпечення.

9. Вкажіть користувачів продуктів інженерії програмного забезпечення. Наведіть приклади продуктів інженерії програмного забезпечення - робочих продуктів.

Програмні продукти можуть створюватися як:
- індивідуальна розробка під замовлення;
- розробка для масового поширення серед користувачів.

10. Специфікування вимог, поняття, витоки

– **процеси** зорієнтовано на формулювання та точний опис (специфікування) вимог, яким має відповідати програмне забезпечення з точки зору замовника;
– **продукти** – специфікації вимог;
– **ресурси** – мови специфікування вимог, діаграмери, інженери зі специфікуванням вимог, комунікатори для зв'язку із замовником.

11. Проектування, поняття, витоки, результат

– **процеси** зорієнтовано на створення архітектури та детального проекту програмного забезпечення згідно зі специфікаціями вимог;
– **продукти** – архітектурний та детальний проекти програми;
– **ресурси** – системи автоматизованого проектування, документатори, архітектори, системні програмісти.

12. Конструювання, поняття, витоки, результат

– **процеси** зорієнтовано на кодування програмного забезпечення згідно з детальним проектом і тестування її з метою виявлення та усунення наявних помилок;
– **продукти** – програмне забезпечення, що відповідає вимогам проекту та тести для її тестування;
– **ресурси** – засоби програмування та тестування, програмісти, тестери.

13. Супроводження, поняття, витоки, результат

– **процеси** – коригувальне, адаптувальне, удосконалювальне та відновлювальне супроводження. *Коригувальне супроводження* – це зміна програмного забезпечення з метою виправлення помилок, яких припустилися на попередніх фазах життєвого циклу. *Адаптувальне супроводження* – це

зміна програмного забезпечення з метою пристосування (адаптації) його до змінених вимог замовника. *Удосконалювальне супроводження* – це зміна програмного забезпечення з метою поліпшення його характеристик (метод обчислень, ефективність функціонування, інтерфейс програми). *Відновлювальне супроводження* – це зміна програмного забезпечення з метою відновлення його працездатності або здобуття інформації про його будову та функціонування;
– **продукти** – супроводжуване програмне забезпечення;
– **ресурси** – вимірювачі, реструктуризатори, абстрактори, екстрактори, засоби програмування, програмісти, інженери із супроводження.

14. Доменний аналіз, поняття

– **процеси** зорієнтовано на аналіз доменної інформації (мається на увазі домен розробки програмного забезпечення) із метою виявлення архітектур, конструкцій, методів для використання їх у розроблювальному програмному забезпеченні;

– **продукти** – архітектури, компоненти, методи;

– **ресурси** – інструменти доменного аналізу, доменні експерти, інженери.

15. Утилізація, поняття, витоки

– **процеси** – це утилізація, що передбачає відновлення, переробку, повторне використання та знищення успадкованого програмного забезпечення. *Відновлення* – це подовження життєвого циклу програмного забезпечення завдяки відновленню її працездатності. *Переробка* – це „глибока” зміна програмного забезпечення (архітектури, функцій) або його міграція в іншу операційну систему чи мову програмування. *Повторне використання* – це створення і застосування на основі успадкованого програмного забезпечення таких компонентів, які можна використати в розробці нового програмного забезпечення. *Знищення* – це знищення невикористаного (неутилізованого) програмного забезпечення;

– **продукти** – відновлене або перероблене програмне забезпечення, повторно використовувані компоненти;

– **ресурси** – екстрактори, абстрактори, гранулятори, програмісти, експерти