**Лабораторна робота №5-1 Процес розробки програмного забезпечення. Оформлення результатів**

**На період** роботи в дистанційній формі навчання на надані запитання потрібно надати письмові відповіді, надіславши їх разом з лабораторною роботою на електронну адресу викладача **t.i.lumpova@gmail.com**.. Файл надавати з іменем у форматі

**ОPI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, О**PI3104L**buts.doc. Відповіді на запитання повинні бути не довгими і змістовними. Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності відповідей-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31 23.03.2023**

**Мета:** Навчиться оформляти результати проектування з погляду на вимоги до програмного забезпечення.

**Завдання:** узагальнити результати своєї роботи над проектом в одному документі, доповнивши його новими розділами.

**Методичні вказівки**

1. Продовжуємо працювати з постановкою задачі з минулих ЛР та ПР з **теми власної курсової роботи**.

Очікуваний підсумковий результат щодо проведеної роботи (що повинно бути зроблено) надано в таблиці:

| ЛР/ПР | Результат |
| --- | --- |
| ПР№1-2 | – Назва проекту, його мета, результати і вхідні дані, основні функції, які реалізовує ПЗ у вигляді структурної схеми – Модель ЖЦ ПЗ та обґрунтування вибору – Схема реалізації моделі ЖЦ ПЗ – Склад бізнес-процесів основних та допоміжних, вказавши перелік допоміжних процесів на кожному етапі ЖЦ ПЗ |
| ПР№1-3 | - |
| ПР№1-4 | - |
| ЛР№1-2 | –Діаграма прецедентів (варіантів / випадків використання) –Таблиця з описом вимог (табл.1 ) |
| ПР№2-2 | Діаграма класів |
| ЛР№2-1 | Діаграма послідовностей для кожного раніш створеного прецеденту |
| ПР№2-3 | Уточнені та узгоджені діаграми прецедентів, класів та послідовностей |
| ЛР№2-2 | Діаграма кооперації, уточнені діаграми прецедентів та послідовностей |
| ПР№2-4 | –Діаграма діяльності –Доповнена таблиця з описом вимог (табл.1 ) |
| ЛР№3-1 | - |
| ЛР№3-2 | - |
| ПР№3-1 | –Документ/розділ „Вимоги користувача”,  –Документ/розділ „Функціональні вимоги до програмного забезпечення” |
| ЛР№3-3 | –Таблиця "Способи перевірки нефункціональних вимог"  –Діаграми прецедентів та кооперації з прив’язкою об’єктів та дій до конкретної вимоги, внесені до документу/розділу „Вимоги користувача” –Доопрацьований документ/розділ „Функціональні вимоги до програмного забезпечення” |
| ЛР№4-1 | Доопрацьовані документи/розділи „Вимоги користувача”, „Функціональні вимоги до програмного забезпечення” |
| ПР№3-2 | – Окремий розділ "Основні технічні рішення "– Окремий розділ "Словник термінів " – Доопрацьовані документи/розділи „Вимоги користувача”, „Функціональні вимоги до програмного забезпечення” |
| ПР№3-3 | - графічні схеми бізнес-процесів та логіки програмної системи  – Окремий розділ "Специфікації процесів"  – Доопрацьовані документи/розділи „Вимоги користувача”, „Функціональні вимоги до програмного забезпечення” |

1. Всі матеріали оформіть одним документом, перевірте узгодженість всіх частин. Матеріали ПР№1-2 оформлюйте як перший розділ Загальні відомості" та другий "Модель ЖЦ ПЗ" з відповідними підрозділами, решту – за схемою складання технічного завдання (ПР№3-1). Цей документ буде вами в подальшому використовуватися в ході вивчення програмної інженерії та конструювання програмного забезпечення, доповнюватися та удосконалюватися.
2. Надайте письмові відповіді на запитання
3. Що в серед наданих матеріалів (див. п.1) відноситься до поведінкових нотацій, а що до структурних?
4. Які на Вашу думку дії з конструювання Ви вже реалізували в своєму проекті?
5. До роботи повинен надаватися звіт з титульним листом із визначенням «Лабораторна робота № », після цього написати назву системи / застосунку для якого відносяться матеріали.
6. По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою. Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

**Строк відсилки ЛР ІПЗ-31 09.05.2024**

**ІПЗ-32 10.05.2024**

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

**Проектування ПЗ** - це процес визначення архітектури, набору компонентів, їх інтерфейсів, інших характеристик системи і кінцевого складу програмного продукту.

Область знань «Проектування ПЗ (Software Design)» складається з таких розділів:

- базові концепції проектування ПЗ (Software Design Basic Concepts),

- ключові питання проектування ПЗ (Key Issue in Software Design),

- структура й архітектура ПЗ (Software Structure and Architecture),

- аналіз і оцінка якості проектування ПЗ (Software Design Quality Analysis and Evaluation),

- нотації проектування ПЗ (Software Design Notations),

- стратегія і методи проектування ПЗ (Software Design Strategies and Methods).

**Базова концепція проектування ПЗ** - це *методологія* проектування архітектури за допомогою різних методів (об'єктного, компонентного й ін.), процесів ЖЦ (стандарт ISO/IEC 12207) і *техніки* - декомпозиція, абстракція, інкапсуляція й ін. На початкових стадіях проектування предметна область декомпозується на окремі об'єкти (при об'єктно-орієнтованому проектуванні) або на компоненти (при компонентному проектуванні). Для подання архітектури програмного забезпечення вибираються відповідні артефакти (нотації, діаграми, блок-схеми і методи).

**Ключові питання проектування** - це декомпозиція програм на функціональні компоненти для незалежного і одночасного їхнього виконання, розподіл компонентів у середовищі функціонування і їх взаємодія між собою, забезпечення якості і живучості системи й ін.

**Проектування архітектури ПЗ** проводиться архітектурним стилем, заснованим на визначенні основних елементів структури - підсистем, компонентів, об'єктів і зв'язків між ними.

**Архітектура проекту** - високорівневе подання структури системи і специфікація її компонентів. Архітектура визначає логіку системи через окремі компоненти системи настільки детально, наскільки це необхідно для написання коду, а також визначає зв'язки між компонентами. Існують і інші види подання структур, засновані на проектуванні зразків, шаблонів, сімейств програм і каркасів програм.

Один з інструментів проектування архітектури - **патери (шаблон).** Це типовий конструктивний елемент ПЗ, що задає взаємодію об'єктів (компонентів) проектованої системи, а також ролі і відповідальності виконавців. Основна мова опису - UML. Патерн може бути **структурним,** що містить у собі структуру типової композиції з об'єктів і класів, об'єктів, зв'язків і ін.; **поведінковим,** що визначає схеми взаємодії класів об'єктів і їх поведінку, задається діаграмами взаємодії, потоків керування й ін.; **погоджувальним,** що відображає типові схеми розподілу ролей екземплярів об'єктів і способи динамічної генерації структур об'єктів і класів.

**Аналіз і оцінка якості проектування ПЗ** - це заходи щодо аналізу сформу­льованих у вимогах атрибутів якості, функцій, структури ПЗ, з перевірки якості ре­зультатів проектування за допомогою метрик (функціональних, структурних і ін.) і методів моделювання і прототипування.

**Нотації проектування** дозволяють представити опис об'єкта (елемента) ПЗ і його структуру, а також поведінку системи. Існує два типи нотацій: структурна, поведінкова, та множина їх різних представлень.

**Структурні нотації -** це структурне, блок-схемне або текстове подання аспектів проектування структури ПЗ з об'єктів, компонентів, їх інтерфейсів і взаємозв'язків. До нотацій відносять формальні мови специфікацій і проектування: ADL (Architecture Description Language), UML (Unified Modeling Language), ERD (Entity-Relation Diagrams), IDL (Interface Description Language) тощо. Нотації містять y собі мовний опис архітектури й інтерфейсу, діаграм класів і об'єктів, діаграм сутність-зв'язок, конфігурації компонентів, схем розгортання, а також структурні діаграми, що задають у наочному вигляді оператори циклу, розгалуження, вибору і послідовності.

**Поведінкові нотації** відбивають динамічний аспект роботи системи та її ком­понентів. Ними можуть бути діаграми потоків даних (Data Flow), діяльності (Activity), кооперації (Colloboration), послідовності (Séquence), таблиці прийняття рішень (Décision Tables), передумови і посту мови (Pre-Post Conditions), формальні мови специфікації (Z, VDM, RAISE) і проектування.

**Стратегія і методи проектування ПЗ.** До стратегій відносять: проектування вгору, вниз, абстрагування, використання каркасів і ін. Методи є функціонально-орієнтовані, структурні, які базуються на структурному аналізі, структурних картах, діаграмах потоків даних й ін.

Вони орієнтовані на ідентифікацію функцій і їх уточнення знизу-вгору, після цього уточнюються діаграми потоків даних і проводиться опис процесів.

В об'єктно-орієнтованому проектуванні ключову роль відіграє спадкування, поліморфізм й інкапсуляція, а також абстрактні структури даних і відображення об'єктів. Підходи, орієнтовані на структури даних, базуються на методі Джексона[[1]](#footnote-1) і використовуються для подання вхідних і вихідних даних структурними діаграмами. Метод UML призначений для опису сценаріїв роботи проекту у наочному діаграмному вигляді. Компонентне проектування ґрунтується на використанні готових компонентів (reuse) з визначеними інтерфейсами і їх інтеграції в конфігурацію, як основи розгортання компонентної системи для її функціонування в операційному середовищі.

Формальні методи опису програм ґрунтуються на специфікаціях, аксіомах, описах деяких попередніх умов, твердженнях і посту мовах, що визначають заключну умову одержання правильного результату програмою. Специфікація функцій і даних, якими ці функції оперують, а також умови і твердження — основа доведення правильності програми.

Конструювання програмного забезпечення (software construction) - детальне створення робочої програмної системи з конструкцій (блоків, операторів, функцій) за допомогою комбінації кодування, верифікації (перевірки), модульного тестування (unit testing), інтеграційного тестування та відлагодження.

З конструюванням програмного забезпечення пов’язані конкретні задачі:

* Перевірка виконання умов, необхідних для успішного конструювання.
* Проектування і написання класів та методів.
* Створення та присвоєння імен змінним та іменованим константам.
* Вибір управляючих структур та організація блоків команд.
* Модульне тестування, інтеграційне тестування та відлагодження власного коду.
* Взаємний огляд коду та низькорівневих програмних структур членами групи.
* Інтеграція програмних компонентів, створених окремо.
* Оптимізація коду, направлена на підвищення його швидкодії і зниження степені використання ресурсів.

Конструювання займає у середньому 30-80% від загального часу, затраченого на роботу і являється центром процесу розробки програмного забезпечення. Це єдиний процес, що в будь-якому випадку виконується при написанні програмного забезпечення.

Область знань «Конструювання ПЗ (Software Construction) містить у собі такі розділи:

- зниження складності (Reduction in Complexity),

- попередження відхилень від стилю (Anticipation of Diversity),

- структуризація перевірок (Structuring for Validation),

- використання стандартів (Use of External Standards).

**Зниження складності** - це мінімізація, зменшення і локалізація складності конструювання.

**Мінімізація складності** - це обмеження на обробку складних структур і великих обсягів інформації протягом тривалого періоду часу. Вона досягається, зокрема, використанням у процесі конструювання простих елементів, а також рекомендацій стандартів.

**Зменшення складності** в конструюванні ПЗ досягається шляхом створення простого коду, що легко читається і спрощує тестування, підвищує продуктивність і впливає на досягнення інших характеристик і обмежень проекту. Зменшення складності спрощує процеси верифікації і тестування результатів конструювання елементів ПС.

**Локалізація складності —** це спосіб конструювання з застосуванням об'єктно-орієнтованого підходу, що лімітує інтерфейс об'єктів, спрощує їхню взаємодію, перевірку правильності самих об'єктів і зв'язків між ними. Локалізація призначена для внесення змін, пов'язаних з виявленими помилками в коді, або коли джерелом помилок є середовище, у якому виконується код.

**Попередження відхилень від стилю.** Для розв'язання різних задач конструювання застосовуються різні стилі конструювання (лінгвістичний, формальний, візуальний).

**Лінгвістичний стиль** заснований на використанні словесних інструкцій і виразів для подання окремих елементів (конструкцій) програм. Він призначений для конструювання нескладних конструкцій і приводиться до вигляду традиційних функцій і процедур або реалізується методами логічного і функціонального програмування й ін.

**Формальний стиль** використовується для точного й однозначного визначення компонентів системи, мінімальної кількості помилок, що можуть виникнути в зв'язку з неоднозначністю визначень або невдалих узагальнень об'єктів конструювання ПЗ.

**Візуальний стиль** - найбільш універсальний для конструювання прикладного ПЗ. Він дозволяє представляти елемент конструювання у наочному вигляді. Візуальна мова проектування UML надає розробнику набір діаграм для подання статичної і динамічної структур ПЗ. При його застосуванні створюється-текстовий і діаграмний опис конструктивних елементів ПЗ, який виводиться на екран дисплея для перегляду і коригування.

**Структуризація перевірок** припускає, що побудова ПС структурована таким чином, що спрощується пошук помилок, дефектів і різних збоїв у процесі перевірок як на стадії незалежного тестування, так і в процесі експлуатації. Структуризації перевірок сприяють огляд, інспектування, спільний перегляд, модульне тестування із застосуванням автоматизованих засобів тестування й ін.

**Використання зовнішніх стандартів.** Конструювання ПЗ залежить від застосовних зовнішніх стандартів, пов'язаних з мовами програмування, інструментальними засобами й інтерфейсами. При конструюванні має бути визначений достатній набір стандартів для керування і забезпечення координації між визначеними видами діяльності і групами операцій, мінімізації складності, внесення змін, аналізу ризиків тощо.

До таких стандартів відносять: мови програмування (Java, Ада 95, C++ і ін.), інтерфейси мов програмування (МП) і прикладні інтерфейси платформ Windows (COM, DCOM), CORBA і ін. При конструюванні використовують стандарти мов опису даних (XML, SQL і ін.), засобів комунікації (COM, CORBA. і ін.), інтерфейсних мов (POS1X, IDL, APL), UML і ін.

Перелічені вище розділи області знань «Конструювання ПЗ» у ядрі знань SWEBOK об'єднуються в групу «Основи конструювання». Крім того, розглядаються групи розділів «Керування конструюванням» та «Практичні міркування». Опишемо першу з них детальніше.

**Керування конструюванням** - це керування процесом конструювання ПЗ, планування, оцінка виконання плану і розроблення заходів щодо внесення змін.

**Моделі конструювання** містять у собі набір операцій, послідовність дій і результатів. Види моделей визначаються стандартом ЖЦ, методологіями і практиками. Деякі стандарти ЖЦ за своєю природою орієнтовані на конструювання типу екстремального програмування і раціонального уніфікованого процесу - RUP (Rational Unified Process).

**Планування** - це визначення порядку операцій, термінів і рівня виконання заданих умов у процесі конструкторської діяльності за моделлю ЖЦ, що містить у собі задачі і дії зі створення, перевірки й оцінки показників якості. Виконавці роз­поділяються за процесами і виконують відповідні задачі з реалізації проміжного і кінцевого продукту. Остаточний результат виміряється за обсягом коду, ступенем повторного використання, кількістю помилок і дефектів, а також оцінюються пока­зники якості ПЗ.

**Внесення змін** пов'язане з помилками, виявленими при перевірці і тестуванні, проводиться з метою збереження функціональної цілісності системи. У випадку виявлення помилок на етапі супроводження приймається рішення про внесення змін або заміну коду у цілому.

**Контрольні питання**

1. В чому полягає процес створення ПЗ?
2. Які причини вдосконалення процесу створення ПЗ?
3. Визначтестратегії впровадження інновацій в розробці ПЗ.
4. Визначте загальносистемні принципи розробки програмного забезпечення.
5. Визначте основністадії типового процесу створення програмного продукту та його основні етапи.
6. Для чого потрібні контрольні точки процесу розробки програмного забезпечення?
7. З яких дій складається етап проектування ПЗ?
8. Чому проектування **ПЗ визначається як стратегічний етап?**
9. Призначення специфікацій програмного продукту.
10. Призначення та сутність функціональних діаграм при аналізі вимог до програмного продукту.
11. Призначення словника термінів.
12. В чому полягає важливість процесу збирання вимог ?
13. Надайте визначення вимоги та конкретизуйте його стосовно ПЗ.
14. Визначте класифікацію вимог.
15. Яким чином визначаються зацікавлені в програмному продукті особи на стадії збирання вимог?
16. Визначте джерела вимог.
17. Які Ви знаєте методи виявлення вимог
18. Які перешкоди виникають при виявленні вимог?
19. Для чого встановлюють пріоритети вимог?

1. Методологія Джексона розглядає ПЗ, як процес обробки послідовного потоку записів, тобто передбачає існування на вході і виході послідовних файлів, структура записів яких повністю визначає структуру програми. Методологія Джексона використовує три засоби для документування проекту програми:

   * Загальна мережна схема потоків даних в системі;
   * Схеми ієрархічної структури для представлення програми і даних;
   * Структурний текст.

   Структурний текст – це формальний псевдокод, який містить правила для запису базових конструкцій, логічних умов і елементарних програмних інструкцій. Для перевірки коректності проекту застосовується контроль того, що всі виходи створюються, а входи використовуються. [↑](#footnote-ref-1)