Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки
Кафедра комп'ютерних технологій та економічної кібернетики

04-05-52М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних і самостійних робіт з навчальної дисципліни

«Технологія створення програмних продуктів»

для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Цифрові технології дистанційної освіти» спеціальності 015.39 «Професійна освіта (цифрові технології)» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІАКОТ
Протокол № 10 від 30.09.2021 р.

Рівне – 2021

Методичні вказівки до виконання лабораторних і самостійних робіт з навчальної дисципліни «Технологія створення програмних продуктів» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Цифрові технології дистанційної освіти» спеціальності 015.39 «Професійна освіта (цифрові технології)» денної та заочної форм навчання [Електронне видання] / Карпович І. М., Гладка О. М. – Рівне : НУВГП, 2021. – 32 с.

Укладачі:
Карпович І. М., к.ф.-м.н., доцент кафедри комп'ютерних технологій та економічної кібернетики;
Гладка О. М., к.т.н., доцент кафедри комп'ютерних технологій та економічної кібернетики.

Відповідальний за випуск:
Грицюк П. М., д.е.н., професор, завідувач кафедри комп'ютерних технологій та економічної кібернетики.

Керівник групи забезпечення спеціальності :
Рощенюк А. М**.,** к.п.н., доцент кафедри комп'ютерних наук та прикладної математики.

ЗМІСТ

Вступ

Останнім часом зростають вимоги до професійної підготовки фахівців із розробки та застосування програмних продуктів. Сучасні технології і інженерія програмного забезпечення – це галузь комп'ютерної науки, яка займається побудовою програмних систем настільки великих і складних, що для цього потрібна участь злагоджених команд розробників різних спеціальностей і кваліфікацій. Зазвичай такі системи удосконалюються протягом тривалого часу, розвиваючись від версії до версії, зазнаючи на своєму життєвому шляху чимало змін, поліпшення існуючих функцій, додавання нових або вилучення застарілих можливостей, адаптацію для роботи в новому середовищі, усунення дефектів і помилок.

Методологія створення складних програмних засобів та стандарти регламентують сучасні процеси управління проектами із розробки складних систем і програмних засобів. Вони забезпечують організацію, освоєння і застосування апробованих високоякісних прийомів проектування, програмування, верифікації, тестування та супроводження програмних засобів і їх компонентів. Вміщені в цьому виданні навчальні матеріали сприятимуть формуванню фахових компетентностей, що дозволять студентам отримувати стабільні передбачувані результати і програмні продукти необхідної якості.

Викладання дисципліни передбачає використання методу проектів. Кожне завдання, вміщене в цих методичних вказівках, являє собою виконання однієї із стадій розробки програмного забезпечення і супроводжується поясненнями з його виконання.

Методичні вказівки містять лише основні теоретичні відомості, необхідні для виконання лабораторних робіт. Перед виконанням завдань та під час підготовки до їх захисту студентові необхідно ознайомитись з конспектом лекцій та опрацювати необхідний матеріал, наведений у переліку рекомендованої літератури, використовуючи електронні джерела та актуальні наукові публікації з комп’ютерних технологій у періодичних виданнях.

Варіанти завдань

Всі лабораторні роботи виконуються студентом за варіантом, визначеним викладачем. Варіант залишається одним і тим же для всіх лабораторних робіт. Під час розробки програми не слід обмежуватися лише функціями, наведеними у варіанті: необхідно запропонувати кілька своїх. Обов'язкове використання структурного та/або об'єктного підходів до програмування.

Завдання

1. Розробити програмний модуль «Облік успішності студентів». Програмний модуль призначений для оперативного обліку успішності студентів в сесію директором інституту, заступниками директора і співробітниками деканату. Відомості про успішність студентів повинні зберігатися протягом всього терміну їх навчання і використовуватися для складання довідок про прослухані курси та додатків до диплому.
2. Розробити програмний модуль «Особові справи студентів». Програмний модуль призначений для отримання відомостей про студентів співробітниками деканату, профкому і відділу кадрів. Відомості повинні зберігатися протягом усього терміну навчання студентів і використовуватися для складанні довідок і звітів.
3. Розробити програмний модуль «Розв’язування комбінаторно-оптимізаційних задач». Модуль повинен містити алгоритми пошуку циклу мінімальної довжини (задача комівояжера), пошуку найкоротшого шляху і пошуку мінімального зв'язує дерева.
4. Розробити додаток Windows «Органiзатор». Додаток призначений для запису, зберігання та пошуку адрес і телефонів фізичних осіб і організацій, а також розкладу, зустрічей та ін. Додаток призначений для будь-яких користувачів комп'ютера.
5. Розробити додаток Windows «Калькулятор». Додаток призначений для будь-яких користувачів і повинен містити всі арифметичні операції (з дотриманням пріоритетів) і (бажано) кілька математичних функцій.
6. Розробити програмний модуль «Кафедра», який містить відомості про співробітників кафедри (ПІБ, посада, науковий ступінь, дисципліни, навантаження, громадська робота, сумісництво та ін.). Модуль призначений для використання співробітниками відділу кадрів і деканату.

7. Розробити програмний модуль «Лабораторія», що містить відомості про співробітників лабораторії (ПІБ, стать, вік, сімейний стан, наявність дітей, посада, науковий ступінь). Модуль призначений для використання співробітниками профкому і відділу кадрів.
8. Розробити програмний модуль «Автосервіс». Під час запису на обслуговування заповнюється заявка, в якій вказуються ПІБ власника, марка автомобіля, вид роботи, дата прийому замовлення і вартість ремонту. Після виконання робіт роздруковується квитанція.
9. Розробити програмний модуль «Облік порушень правил дорожнього руху». Для кожної автомашини (і її власника) в базі зберігається список порушень. Для кожного порушення фіксується дата, час, вид порушення і розмір штрафу. Після оплати всіх штрафів авто видаляється з бази.
10. Розробити програмний модуль «Картотека агентства нерухомості», призначений для використання працівниками агентства. У базі містяться відомості про квартири (кількість кімнат, поверх, площа та ін.). У випадку надходження заявки на обмін (купівлю, продаж) здійснюється пошук відповідного варіанту. Якщо такого немає, клієнт заноситься в клієнтську базу і повідомляється, коли варіант з'являється.
11. Розробити програмний модуль «Картотека абонентів АТС». Картотека містить відомості про телефони і їх власників. Фіксує заборгованості з оплати (абонентської і погодинної). Вважається, що погодинна оплата місцевих телефонних розмов вже введена.
12. Розробити програмний модуль «Авіакаса», що містить відомості про наявність вільних місць на авіамаршрути. У базі повинні міститися відомості про номер рейсу, екіпаж, тип літака, дату і час вильоту, а також вартості авіаквитків (різного класу). Після надходження заявки на квитки програма здійснює пошук відповідного рейсу.
13. Розробити програмний модуль «Книжковий магазин», що містить відомості про книги (автор, назва, видавництво, рік видання, ціна). Покупець оформляє заявку на потрібні йому книги, якщо таких немає, він заноситься в базу і повідомляється, коли потрібні книги надходять в магазин.

14. Розробити програмний модуль «Автостоянка». В програмі міститься інформація про марку автомобіля, його власника, дату і час в'їзду, вартості стоянки, знижки, заборгованості з оплати тощо.
15. Розробити програмний модуль «Кадрове агентство», що містить відомості про вакансії і резюме. Програмний модуль призначений як для пошуку співробітника, що відповідає вимогам керівників фірми, так і для пошуку підходящої роботи.

Лабораторна робота № 1

Тема: Розробка технічного завдання

Мета роботи

Ознайомитися з правилами написання технічного завдання, що є одним з етапів розробки програмного забезпечення.

Основні відомості

Література: [1-5, 7, 8]

Технічне завдання – це документ, в якому сформульовані основні цілі розробки, вимоги до програмного продукту (функціональні і нефункціональні), визначені терміни і етапи розробки, регламентовано процес приймально-здавальних випробувань. В розробці технічного завдання беруть участь як представники замовника, так і представники виконавця. В основі цього документа лежать вимоги замовника, аналіз передових досягнень техніки, результати виконання науково-дослідних робіт, передпроектних досліджень, наукового прогнозування тощо.

Розробка технічного завдання виконується в наступній послідовності.

1. Встановлюється набір функцій, для виконання яких призначена розробка, а також перелік і характеристики вхідних даних. Потім визначаються перелік результатів, їх характеристики і способи подання.

Приклад. Опис вихідних і вхідних повідомлень

Таблиця 1

Параметри вихідного повідомлення «Навчальний план»

Ідентифікатор	Навчальний план
Форма подання	Електронний друкований документ
Періодичність видачі	1 раз в 5 років
Терміни видачі та допустимий час затримки	За вимогою
Атрибути повідомлень	Спеціальність; тематика циклу; тип навчання; форма навчання; номер теми розділу; тема розділу; кількість годин на розділ - всього, в тому числі, годин на теорію, на практику; всього годин на цикл, в тому числі, на теорію і на практику; ПІБ зав. відділенням
Одержувач інформації	Завідувач відділенням

Таблиця 2

Параметри вихідного повідомлення
«Навчально-тематичний план»

Ідентифікатор	Навчально-тематичний план
Форма подання	Електронний друкований документ
Періодичність видачі	1 раз в 5 років
Терміни видачі та допустимий час затримки	За вимогою
Атрибути повідомлень	Вид навчального заняття (теорія / практика); кількість годин на даний вид занять; спеціальність; тип навчання; форма навчання; номер теми розділу; тема розділу; номер підрозділу; тема підрозділу; зміст підрозділу; кількість годин на розділ; кількість годин на підрозділ; ПІБ зав. відділенням
Одержувач інформації	Завідувач відділенням

Таблиця 3

Параметри вхідного повідомлення
«Навчально-виробничий план»

Ідентифікатор	Навчально-виробничий план
Форма подання	Таблиця даних
Періодичність видачі	2 рази на рік
Терміни видачі та допустимий час затримки	11 січня, 31 серпня +1 день
Атрибути повідомлень	Спеціальність; тематика навчання; вид навчання; дата початку та закінчення навчання; контингент студентів
Джерело інформації	База даних системи комплектування

Таблиця 4

Параметри вхідного повідомлення «Список групи»

Ідентифікатор	Список групи
Форма подання	Таблиця даних
Періодичність видачі	2 рази на рік
Терміни видачі та допустимий час затримки	11 січня, 31 серпня +1 день
Атрибути повідомлень	Спеціальність; тематика навчання; вид навчання; дата початку та закінчення навчання; ПІБ слухачів
Джерело інформації	База даних системи комплектування

2. Уточнюється середовище функціонування програмного забезпечення: конкретна комплектація і параметри технічних засобів, версія операційної системи, версії і параметри іншого встановленого програмного забезпечення, з яким буде взаємодіяти програмний продукт, що розробляється.

3. У випадках, коли програмне забезпечення призначене для збирання і зберігання певної інформації або включається в управління якимось технічним процесом, необхідно також чітко регламентувати дії програми в разі збоїв обладнання і енергопостачання.

Вимоги до технічного завдання

1. Загальні положення.

1.1. Технічне завдання оформляють відповідно до стандарту на аркушах формату А4, зазвичай, без заповнення полів. Номери аркушів (сторінок) проставляють у верхній частині аркуша над текстом.

1.2. Аркуш затвердження і титульний аркуш оформляють у відповідності із стандартом [4].

1.3. Для внесення змін і доповнень у технічне завдання на подальших стадіях розробки програмного продукту виготовляють доповнення до нього. Погодження та затвердження доповнень до технічного завдання проводять у тому ж порядку, який встановлений для технічного завдання.

1.4. Технічне завдання має містити наступні розділи:

- вступ;
- найменування та область застосування;
- підстава для розробки;
- призначення розробки;
- технічні вимоги до програми або програмного виробу;
- техніко-економічні показники;
- стадії і етапи розробки;
- порядок контролю і приймання;
- додатки.

В залежності від особливостей програмного продукту допускається уточнювати зміст розділів, вводити нові розділи або об'єднувати окремі з них. За необхідності допускається в технічне завдання включати додатки.

2. Зміст розділів.

2.1. Вступ повинен включати коротку характеристику області застосування програми або програмного продукту, а також об'єкта (наприклад, системи), в якому передбачається їх використовувати. Основне призначення вступу –

продемонструвати актуальність даної розробки і показати, яке місце ця розробка займає серед схожих продуктів.

2.2. У розділі «Найменування і галузь застосування» вказують найменування, коротку характеристику галузі застосування програми і об'єкта, в якому використовують програмний продукт.

2.3. У розділі «Підстава для розробки» повинні бути вказані:

• документ (документи), на підставі яких ведеться розробка. Таким документом може служити план, наказ, договір тощо.

• організація, що затвердила цей документ, і дата його затвердження;

• найменування та (або) умовне позначення теми розробки.

2.4. У розділі «Призначення розробки» має бути вказано функціональне і експлуатаційне призначення програмного забезпечення

2.5. Розділ «Технічні вимоги до програми або програмного виробу» повинен містити такі підрозділи:

• вимоги до функціональних характеристик;
• вимоги до надійності;
• умови експлуатації;
• вимоги до складу і параметрів технічних засобів;
• вимоги до інформаційної та програмної сумісності;
• вимоги до маркування та упаковки;
• вимоги до транспортування і зберігання;
• спеціальні вимоги.

2.5.1. У підрозділі «Вимоги до функціональних характеристик» повинні бути вказані вимоги до складу виконуваних функцій, організації вхідних та вихідних даних, часових характеристик тощо.

2.5.2. У підрозділі «Вимоги до надійності» вказуються вимоги до забезпечення надійного функціонування (забезпечення стійкого функціонування, контроль вхідної і вихідної інформації, час відновлення після відмови тощо).

2.5.3. У підрозділі «Умови експлуатації» повинні бути вказані умови експлуатації (температура навколишнього повітря, відносна вологість і т. і. для обраних типів носіїв даних), за яких

будуть забезпечуватися задані характеристики, а також вид обслуговування, необхідна кількість і кваліфікація персоналу.

2.5.4. У підрозділі «Вимоги до складу і параметрів технічних засобів» вказують необхідний склад технічних засобів із зазначенням їх технічних характеристик.

2.5.5. У підрозділі «Вимоги до інформаційної та програмної сумісності» повинні бути вказані вимоги до інформаційних структур на вході і виході, методів розв'язування, початкового коду, мов програмування. За необхідності повинен забезпечуватися захист інформації і програм.

2.5.6. У підрозділі «Вимоги до маркування та упаковки» в загальному випадку вказують вимоги до маркування програмного виробу, варіанти і способи упаковки.

2.5.7. У підрозділі «Вимоги до транспортування і зберігання» повинні бути вказані для програмного продукту умови транспортування, місця та умови зберігання, терміни зберігання в різних умовах.

2.6. У розділі «Техніко-економічні показники» повинні бути вказані: орієнтовна економічна ефективність, прогнозована річна потреба, економічні переваги розробки в порівнянні з кращими вітчизняними і зарубіжними зразками або аналогами.

2.7. У розділі «Стадії і етапи розробки» встановлюють необхідні стадії розробки, етапи та зміст робіт (перелік програмних документів, які повинні бути розроблені, узгоджені та затверджені), а також, зазвичай, визначають терміни розробки та виконавців.

2.8. У розділі «Порядок контролю і приймання» повинні бути вказані види випробувань і загальні вимоги до приймання роботи.

2.9. У додатках до технічного завдання за необхідності наводять:

- перелік науково-дослідних та інших робіт, які обгрунтовують розробку;
- схеми алгоритмів, таблиці, описи, обгрунтування, розрахунки та інші документи, які можуть бути використані під час розробки;
- інші джерела розробки.

У випадках, якщо якісь вимоги, передбачені технічним завданням, замовник не висуває, то слід у відповідному місці вказати «Вимоги не висуваються».

Порядок виконання завдання

1. Виконати постановку задачі (відповідно до свого варіанту).
2. Розробити технічне завдання на програмний продукт.
3. Оформити роботу відповідно до стандарту.

Вимоги до звіту

Звіт з лабораторної роботи повинен складатися з:

1. Постановки завдання:
   1.1 Функціональних вимог;
   1.2 Нефункціональних вимог;
   1.3 Вихідних повідомлень;
   1.4 Вхідних повідомлень.
2. Технічного завдання на програмний продукт.

Контрольні запитання

1. Назвіть основні етапи розробки програмного забезпечення.
2. Що включає в себе постановка задачі?
3. Назвіть основні розділи технічного завдання.

Лабораторна робота № 2

Тема: Підготовка ескізного проекту

Мета роботи

Навчитися створювати формальні моделі та на їх основі визначати специфікації програмного забезпечення, що розробляється.

Основні відомості

Література: [1, 3, 5, 6-8, 11]

Розробка програмного забезпечення починається з аналізу вимог до нього. В результаті аналізу отримують специфікації програмного забезпечення, що формують загальну модель його

взаємодії з користувачем або іншими програмами і конкретизують його основні функції.

У структурному підході на етапі аналізу і визначення специфікацій розробляють три типи моделей: моделі функцій, моделі даних і моделі потоків даних. Оскільки різні моделі описують проектоване програмне забезпечення з різних сторін, рекомендується використовувати відразу кілька моделей, що розробляються у вигляді діаграм, і пояснювати їх текстовими описами, словниками тощо.

Структурний аналіз передбачає використання наступних видів моделей [7]:

• діаграм потоків даних (DFD – Data Flow Diagrams), що описують взаємодію джерел і споживачів інформації через процеси, які повинні бути реалізовані в системі;

• діаграм «сутність-зв'язок» (ERD – Entity-Relationship Diagrams), які описують бази даних системи, що розробляється;

• діаграм переходів станів (STD – State Transition Diagrams), що характеризують поведінку системи в часі;

• функціональних діаграм (методика SADT);

• специфікацій процесів;

• словника термінів.

Специфікації процесів зазвичай подають у вигляді короткого текстового опису, схем алгоритмів, псевдокоду або flow-форм.

Словник термінів являє собою короткий опис основних понять, які використовуються для складання специфікацій. Він повинен включати визначення основних понять предметної області, опис структур елементів даних, їх типів і форматів, а також усіх скорочень і умовних позначень [1].

За допомогою діаграм переходів станів можна моделювати подальше функціонування системи на основі її попереднього і поточного функціонування. Система, яка моделюється, в будь-який заданий момент часу перебуває в одному із станів. З часом вона може змінити свій стан, але переходи між станами повинні бути точно визначені.

Функціональні діаграми відображають взаємозв'язки функцій програмного забезпечення, що розробляється. Вони створюються на ранніх етапах проектування систем для того,

щоб допомогти проектувальнику виявити основні функції і складові частини проектованої системи і, за можливості, виявити і усунути істотні помилки. Для створення функціональних діаграм пропонується використовувати методологію SADT [1].

Для опису потоків інформації в системі застосовуються діаграми потоків даних (DFD – Data flow diagrams). DFD дозволяє описати необхідну поведінку системи у вигляді сукупності процесів, які взаємодіють за допомогою потоків даних, що їх зв'язують. DFD показує, як кожен з процесів перетворює свої вхідні потоки даних у вихідні потоки даних і як процеси взаємодіють між собою.

Діаграма сутність-зв'язок – інструмент розробки моделей даних, що забезпечує стандартний спосіб визначення даних і відношень між ними. Така діаграма не дуже деталізована. Вона включає сутності і взаємозв'язки, що відображають основні бізнес-правила предметної області, а також вимоги, що пред'являються до інформаційної системи.

На етапі аналізу і визначення специфікацій розробляють: Словник проекту, Модель варіантів використання (use case diagram), Опис потоків подій варіантів використання, які відповідають принципам об'єктно-орієнтованого підходу.

Порядок виконання роботи

На основі технічного завдання з лабораторної роботи №1 підготувати документ з назвою “Ескізний проект”. Для цього:

1. Виконати аналіз функціональних та експлуатаційних вимог до програмного продукту.

2. Визначити основні технічні рішення (вибір мови програмування, структура програмного продукту, склад функцій програмного продукту, режими функціонування) і занести результати в “Ескізний проект”.

Структурний підхід:

3. Визначити діаграми потоків даних для задачі, що розв'язується.

4. Визначити діаграми «сутність-зв'язок», якщо програмний продукт містить базу даних.

5. Визначити функціональні діаграми.

6. Визначити діаграми переходів станів.
7. Визначити специфікації процесів.
8. Додати словник термінів.

Об'єктно-орієнтований підхід[6]:

3. Словник проекту.
4. Діаграма варіантів використання (use case diagram).
5. Опис потоків подій варіантів використання.
6. Діаграми кооперації (collaboration diagram).
7. Діаграми послідовності варіантів використання (sequence diagram).
8. Діаграми класів рівня концепції.
9. Оформити результати у вигляді ескізного проекту.
10. Здати і захистити роботу.

Вимоги до звіту

Звіт з лабораторної роботи повинен складатися з:

1. Постановки завдання.
2. Документа «Ескізний проект», що містить:
   - вибір методу розв’язування і мови програмування;
   - специфікації процесів;
   - всі отримані діаграми;
   - словник термінів.

Контрольні запитання

1. Що таке життєвий цикл програмного забезпечення?
2. У чому суть постановки задачі і передпроектних досліджень?
3. Назвіть функціональні і нефункціональні (експлуатаційні) вимоги до програмного продукту.
4. Перерахуйте складові ескізного проекту.

Лабораторна робота № 3

Тема: Розробка технічного проекту

Мета роботи

Опанувати основи технології і особливості проектування програмного забезпечення.

Основні відомості

Література: [1, 5, 7, 8, 11]

Технічний проект – образ наміченого до створення об'єкта, представлений у вигляді його опису, схем, креслень, розрахунків, обґрунтувань, числових показників. Мета технічного проекту – визначення основних методів, які використовуються для створення інформаційної системи, і остаточне визначення її кошторисної вартості. Технічне проектування підсистем здійснюється відповідно до затвердженого технічного завдання.

Технічний проект програмної системи докладно описує:

• функції, варіанти їх використання і відповідні їм документи;

• структури баз даних, взаємозв'язки даних;

• алгоритми опрацювання даних.

Технічний проект повинен включати дані про обсяги та інтенсивність потоків інформації, кількість користувачів програмної системи, характеристики обладнання та програмного забезпечення, що взаємодіє з програмним продуктом.

Під час розробки технічного проекту оформляються:

• відомість технічного проекту (загальна інформація про проект);

• пояснювальна записка до технічного проекту (вступна інформація, що дозволяє її споживачеві швидко освоїти дані з конкретного проекту);

• опис систем класифікації та кодування;

• перелік вхідних даних (перелік інформації, яка використовується як вхідний потік і служить джерелом накопичення);

• перелік вихідних даних (перелік інформації, яка використовується для аналізу накопичених даних);

• опис програмного забезпечення (перелік програмного забезпечення та СУБД, які планується використовувати для створення інформаційної системи);

• опис технічних засобів (перелік апаратних коштів, на яких планується робота проектованого програмного продукту);

• проектна оцінка надійності системи (експертна оцінка надійності з виявленням найбільш благополучних ділянок програмної системи і її вузьких місць);

• відомість обладнання і матеріалів (перелік обладнання і матеріалів, які будуть потрібні в ході реалізації проекту).

Структурний підхід. Структурною називають схему, яка відображає склад, взаємодію і управління частинами програмного забезпечення. Структурна схема визначається архітектурою програмного забезпечення, що розробляється [1].

Функціональна схема – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках і зазначенням файлів та пристроїв.

Розробка алгоритмів. Метод покрокової деталізації реалізує низхідний підхід до програмування і передбачає покрокову розробку алгоритму.

Структурні карти. Методика структурних карт використовується на етапі проектування програмного забезпечення для того, щоб продемонструвати, яким чином програмний продукт виконує системні вимоги. Структурні карти призначені для опису відношень між модулями. Метод структурного програмування передбачає відповідність між структурою потоків даних і структурою програми. Основна увага в методі зосереджена на відповідності вхідних і вихідних потоків даних.

Об'єктно-орієнтований підхід. Задачі проектування включають в себе дві складові: логічне і фізичне проектування програмних продуктів. Логічне проектування полягає в розробці класів для реалізації їх примірників - об'єктів. Для цього необхідний детальний опис полів і методів класів, а також зв'язків між ними. Тут використовуються як статичні діаграми класів і об'єктів, так і динамічні – послідовностей станів та кооперації. Фізичне проектування передбачає побудову програмних компонентів з раніше визначених класів і об'єктів та розміщення їх на конкретних обчислювальних пристроях. На цьому етапі розробляються діаграми компонентів і розгортання.

Логічна модель програмного забезпечення містить:

1. Діаграму класів (class diagram);
2. Діаграми станів класів (statechart diagram);
3. Діаграму діяльності (activity diagram).

Фізична модель програмного забезпечення (реалізація системи)включає:

1. Діаграму компонентів (component diagram);
2. Діаграму розгортання (deployment diagram);
3. Генерацію коду.

Порядок виконання роботи

Для програмного продукту, для якого формувалося технічне завдання в лабораторній роботі № 1 і ескізний проект в лабораторній роботі № 2, підготувати документ з назвою "Технічний проект". Для цього:

1. Розробити уточнені алгоритми програм, що складають заданий програмний модуль. Використати метод покрокової деталізації.

Структурний підхід:

2. На основі уточнених та доопрацьованих алгоритмів розробити структурну схему програмного продукту.
3. Розробити функціональну схему програмного продукту.
4. Подати структурну схему у вигляді структурних карт.

Об'єктно-орієнтований підхід:

Логічна модель програмного забезпечення:

1) діаграма класів (class diagram);
2) діаграми станів класів (statechart diagram);
3) діаграма діяльності (activity diagram).

Фізична модель програмного забезпечення (реалізація системи):

1) діаграма компонентів (component diagram);
2) діаграма розгортання (deployment diagram);
3) генерація коду.

5. Оформити результати у вигляді технічного проекту.

Вимоги до звіту

Звіт з лабораторної роботи повинен складатися з:

1. Структурної схеми програмного продукту.
2. Функціональної схеми.
3. Алгоритму програми.

4. Діаграми UML логічного і фізичного рівня.
5. Закінченого технічного проекту програмного модуля.

Контрольні запитання

1. У чому суть проектування програмного забезпечення?
2. Перелічіть складові технічного проекту.
3. Охарактеризуйте структурний підхід до програмування.
4. З чого складаються структурна і функціональна схеми?
5. Сформулюйте поняття псевдокоду.
6. Охарактеризуйте всі діаграми UML логічного і фізичного рівня.

Лабораторна робота № 4

Тема: Реалізація програмного продукту

Мета роботи

Розробити програмний продукт відповідно до заданого варіанту.

Основні відомості

Література: [1,3-5,7-9]

Важливим етапом розробки програмного продукту є складання програмної документації. Життєвий цикл програмного забезпечення містить спеціальний процес, присвячений цьому питанню. На кожен програмний продукт повинні складатися два типи документації – для розробників і для різних груп користувачів.

Програмна документація користувачів повинна містити всі необхідні відомості щодо експлуатації програмного забезпечення. Документація розробника повинна містити відомості, необхідні для розробки і супроводу програмного забезпечення.

Види програмних документів

Документування програмного забезпечення здійснюється відповідно до системи вимог до програмної документації [4]. Стандарт містить перелік документів для програмного забезпечення різних типів, серед яких:

• специфікація, яка повинна містити перелік і короткий опис призначення всіх файлів програмного забезпечення, в тому числі, і файлів документації на нього. Вона є обов'язковою для програмних систем, а також їх компонентів, що мають самостійне застосування;
• відомість власників оригіналів повинна містити список підприємств, на яких зберігаються оригінали програмних документів. Необхідність цього документа визначається на етапі розробки і затвердження технічного завдання лише для програмного забезпечення із складною архітектурою;
• текст програми повинен містити текст програми з необхідними коментарями. Необхідність цього документа визначається на етапі розробки і затвердження технічного завдання;
• опис програми повинен містити відомості про логічну структуру та функціонування програми. Необхідність цього документа також визначається на етапі розробки і затвердження технічного завдання;
• відомість експлуатаційних документів повинна містити перелік експлуатаційних документів на програму. Необхідність цього документа також визначається на етапі розробки і затвердження технічного завдання;
• формуляр повинен містити основні характеристики програмного забезпечення, комплектність і відомості про експлуатацію програми;
• опис застосування повинен містити відомості про призначення програмного забезпечення, області застосування, використані методи, класи задач, які можуть бути розв’язані, обмеження для застосування, мінімальну конфігурацію технічних засобів;
• керівництво системного програміста повинно містити відомості для перевірки, забезпечення функціонування і налаштування програми на умови конкретного застосування;
• керівництво програміста повинно містити відомості для експлуатації програмного забезпечення;
• керівництво оператора містить відомості для забезпечення процедури спілкування оператора з обчислювальною системою в процесі виконання програми;
• опис мови – опис синтаксису і семантики мови програми;

• керівництво з технічного обслуговування містить відомості для застосування програми під час обслуговування технічних засобів.

Порядок виконання роботи

Розробити програмний продукт, для якого створені документи в лабораторних роботах № 1-3, і оформити документацію до нього.

1. За результатами лабораторних робіт № 1-3 написати код програм для розв'язування поставленої задачі мовою програмування, обраною на етапі ескізного проектування.

2. Відлагодити програмний модуль.

3. Отримати результати роботи.

4. Оформити документацію до розробленого програмного забезпечення.

Вимоги до звіту

Звіт з лабораторної роботи повинен складатися з:

1. Лістингу програм.
2. Інтерфейсу користувача.
3. Документації до програмного забезпечення (керівництво користувача, керівництво системного програміста, керівництво програміста, керівництво оператора).
4. Результатів роботи програм.

Контрольні запитання

1. У чому полягає етап реалізації та налагодження програми?
2. Які існують інструментальні засоби розробки?
3. Що таке документація до програмного забезпечення?

Лабораторна робота № 5

Тема: Тестування програми

Мета роботи

Вивчити методи тестування логіки програми, формалізовані описи результатів тестування і стандарти із складання схем програм.

Основні відомості

Література: [1, 5, 10]

Тестування програмного забезпечення включає в себе комплекс дій, аналогічних послідовності процесів розробки програмного забезпечення. До нього входять:

- постановка задачі для тесту;
- проектування тесту;
- написання тестів;
- тестування тестів;
- виконання тестів;
- вивчення результатів тестування.

Важливим етапом є проектування тестів. Існують різні підходи до його реалізації. Перший полягає в тому, що тести проектуються на основі зовнішніх специфікацій програм і модулів або специфікацій сполучення модуля з іншими модулями. Програма тут розглядається як «чорна скринька». Сенс тесту полягає в тому, щоб перевірити, чи відповідає програма зовнішнім специфікаціям. Зміст модуля в цьому випадку не має значення. Такий підхід отримав назву – стратегія «чорної скриньки».

Другий підхід – стратегія «білої скриньки», який ґрунтується на аналізі логіки програми. За такого підходу тестування передбачає перевірку кожного шляху, тобто кожної гілки алгоритму, а зовнішня специфікація до уваги не береться.

Жоден з цих підходів не є оптимальним. Реалізація тестування методом «чорної скриньки» зводиться до перевірки всіх можливих комбінацій вхідних даних. Неможливо протестувати програму, подаючи на вхід безліч значень, тому обмежуються певним набором даних. В цьому випадку оцінюють максимальну віддачу тесту у порівнянні з витратами на його створення. Вона вимірюється ймовірністю того, що тест виявить помилки, якщо вони є в програмі. Витрати вимірюються часом і вартістю підготовки, виконання та перевірки результатів тесту.

Тестування методом «білої скриньки» також не дає гарантії того, що модуль не містить помилок. Навіть якщо припустити, що виконані тести для всіх гілок алгоритму, не можна з повною впевненістю стверджувати, що програма відповідає її специфікаціям. Наприклад, якщо потрібно написати програму для обчислення кубічного кореня, а програма фактично обчислює

корінь квадратний, то реалізація буде абсолютно неправильною, навіть якщо перевірити всі шляхи.

Інша проблема – відсутні шляхи. Якщо програма реалізує специфікації у повному обсязі (наприклад, відсутня така спеціалізована функція, як перевірка на від'ємне значення вхідних даних програми обчислення квадратного кореня), ніяке тестування існуючих шляхів не виявить такої помилки. І нарешті, проблема залежності результатів тестування від вхідних даних. Одні дані будуть давати правильні результати, а інші ні. Якщо концентрувати увагу лише на тестуванні шляхів, немає гарантії, що ця помилка буде виявлена.

Таким чином, повне тестування програми неможливе, іншими словами, ніяке тестування не гарантує повну відсутність помилок в програмі. Тому необхідно проектувати тести таким чином, щоб збільшити ймовірність виявлення помилки в програмі.

Стратегія «білої скриньки».

Існують наступні методи тестування за принципом «білої скриньки»:

- покриття операторів;
- покриття розв'язувань;
- покриття умов;
- покриття розв'язувань / умов;
- комбінаторне покриття умов.

Метод покриття операторів. Метою цього методу тестування є виконання кожного оператора програми хоча б один раз.

Метод покриття розв'язувань (покриття переходів). Згідно з цим методом кожен напрямок переходу має бути реалізовано, принаймні, один раз. Цей метод включає в себе критерій покриття операторів, оскільки під час виконання всіх напрямків переходів будуть виконуватися всі оператори, що знаходяться на цих напрямках.

Метод покриття умов. Цей метод може дати кращі результати в порівнянні з попередніми. Відповідно до методу покриття умов записується кількість тестів, достатня для того, щоб усі можливі результати кожної умови в розв'язуванні виконувалися, принаймні, один раз.

Метод покриття розв'язувань / умов. Цей критерій вимагає такого достатнього набору тестів, щоб виконувалися, принаймні, один раз всі можливі результати кожної умови і всі результати кожного розв'язування, а також, щоб кожній точці входу передавалося управління, принаймні, один раз. До недоліків методу відносять те, що не завжди можна перевірити всі умови; неможливо перевірити умови, які приховані іншими умовами, а також недостатню чутливість до помилок в логічних виразах.

Метод комбінаторного покриття умов. Критерій комбінаторного покриття умов задовольняє також і критеріям покриття розв'язувань, покриття умов і покриття розв'язувань / умов. Цей метод вимагає створення такої кількості тестів, щоб усі можливі комбінації результатів умови в кожному розв'язуванні виконувалися, принаймні, один раз.

Порядок виконання роботи

Спроектувати тести і, використовуючи їх, протестувати програму, розроблену в лабораторній роботі № 4.

1. Спроектувати тести за принципом «білої скриньки» і, використовуючи їх, протестувати програму, розроблену в лабораторній роботі № 4. Використовувати схеми алгоритмів, розроблені і уточнені в лабораторних роботах № 2 і № 3.

2. Вибрати кілька алгоритмів для тестування і позначити буквами або цифрами гілки цих алгоритмів.

3. Виписати шляхи алгоритму, які повинні бути перевірені тестами для обраного методу тестування.

4. Записати тести, які дозволять пройти шляхами алгоритму.

5. Протестувати розроблену програму. Результати оформити у вигляді таблиць.

6. Перевірити всі види тестів, зробити висновки про їх ефективність.

7. Оформити звіт про лабораторну роботу.

Вимоги до звіту

Звіт з лабораторної роботи повинен складатися з:

1. Постановки завдання.
2. Блок-схеми програми.

3. Тестів.
4. Таблиць тестування програми.
5. Висновків за результатами тестування (не забувайте, що метою тестування є виявлення помилок у програмі).

Контрольні запитання

1. Які існують види тестування?
2. Назвіть критерії вибору тестів.
3. Перерахуйте властивості тестів.

Лабораторна робота № 6

Тема: Використання технології RUP

Мета роботи

Сформувати навички: роботи з реальними замовниками програмних систем; ідентифікації зацікавлених осіб та інтерв'ю з ними; аналізу отриманого матеріалу; формулювання проблеми, її актуальності і потреб зацікавлених осіб.

Основні відомості

Література: [1, 5, 11, 12]

Rational Unified Process (RUP) – це ітеративний процес створення програмного забезпечення. Кожна ітерація на основі спіральної моделі життєвого циклу розробки створює фрагмент системи або її нову версію. Основна мета – гарантувати високу якість програмного продукту в межах відведеного графіка і передбаченого бюджету.

Технологія RUP (Rational Unified Process) ґрунтується на трьох ключових ідеях.

• Весь хід робіт підпорядковується підсумковим цілям проекту, вираженими у вигляді варіантів використання (use cases) – сценаріїв взаємодії завершеної програмної системи з користувачами або іншими системами, при виконанні яких користувачі отримують важливі для них результати і послуги. Розробка починається з виділення варіантів використання і на кожному кроці контролюється мірою наближення до їх реалізації.

• Основним рішенням, прийнятим у ході проекту, є архітектура завершеної програмної системи. Архітектура встановлює набір компонентів, з яких буде побудоване програмне забезпечення, відповідальність кожного з компонентів (які підзадачі він розв'язує в рамках загальних задач системи), визначає інтерфейси, через які вони можуть взаємодіяти, а також способи взаємодії компонентів один з одним.

Архітектура є одночасно основою для отримання якісного програмного забезпечення і базою для планування робіт та оцінок проекту в термінах часу і ресурсів, необхідних для досягнення певних результатів. Вона оформляється у вигляді набору графічних моделей мовою UML.

• Основою процесу розробки є заплановані і керовані ітерації, обсяг яких (реалізована в рамках ітерації функціональність і набір компонентів) визначається на основі архітектури.

Етап Початок (Inception). Головне призначення етапу – запустити проект. Основна мета цієї фази – досягти компромісу між усіма зацікавленими особами щодо завдань проекту та виділених на нього ресурсів. На цій стадії визначаються основні цілі проекту, керівник і бюджет, основні засоби виконання – технології, інструменти, ключові виконавці. Тут може проводитися апробація вибраних технологій, щоб переконатися у спроможності досягнення цілей з їх допомогою, а також складаються попередні плани проекту. На цю фазу може витрачатися близько 10% часу і 5% трудомісткості одного циклу.

Цілі етапу *Початок*:

• визначити область застосування системи, що проектується (її призначення, межі, інтерфейси з зовнішнім середовищем, критерій визнання - приймання);

• визначити елементи Use Case, критичні для системи (основні сценарії поведінки, які визначають її функціональність і покривають головні проектні рішення);

• визначити загальні риси архітектури, що забезпечує основні сценарії, створити демонстраційний макет;

• визначити загальну вартість і план всього проекту і забезпечити деталізовані оцінки для етапу розвитку;

• ідентифікувати основні елементи ризику.

Основні дії етапу *Початок*:

• формулювання області застосування проекту – виявлення вимог і обмежень, що розглядаються як критерій визнання остаточного продукту;

• планування та підготовка бізнес-варіанту і альтернатив розвитку для управління ризиком, визначення персоналу, проектного плану, а також виявлення залежностей між вартістю, плануванням і корисністю;

• синтезування попередньої архітектури, розвиток компромісних рішень проектування; визначення рішень розробки, купівлі і повторного використання, для яких можна оцінити вартість, планування і ресурси.

В результаті етапу *Початок* створюються такі артефакти:

• концепція (Vision) – специфікація з переліком основних проектних вимог, ключових характеристик і головних обмежень;

• початкова модель Use Case (20% від повного уявлення);

• початковий словник проекту (глосарій);

• початковий бізнес-варіант (зміст бізнесу, критерій успіху – прогноз доходу, прогноз ринку, фінансовий прогноз);

• початкове оцінювання ризику;

• проектний план, в якому показані етапи і ітерації.

Приклади оформлення документів RUP (артефактів) можна переглянути, наприклад, за посиланням:
https://uadoc.zavantag.com/text/35/index-1.html?page=4

Етап Розвиток (Elaboration). Головне призначення етапу – створити архітектурний базис системи. Основна мета цієї фази – на базі основних найбільш істотних вимог розробити стабільну базову архітектуру продукту, яка дозволяє розв'язувати поставлені перед системою задачі і в подальшому використовується як основа розробки системи. На цю фазу може витрачатися близько 30% часу і 20% трудомісткості одного циклу.

Цілі етапу *Розвиток*:

• визначити остаточні вимоги, функціональні вимоги сформулювати як елементи Use Case;

• визначити архітектурну платформу системи;

• відслідковувати ризик, усунути джерела найбільшого ризику;
• розробити план ітерацій етапу *Реалізація*.

Основні дії етапу *Розвиток*:

• розвиток специфікації уявлення, повне формування критичних елементів Use Case, які задають подальші рішення;
• розвиток архітектури, виділення її компонентів.

В результаті етапу *Розвиток* створюються такі артефакти:

• модель Use Case (80% від повного уявлення);
• додаткові вимоги (нефункціональні вимоги, а також інші вимоги, які не пов'язані з конкретним елементом Use Case);
• опис програмної архітектури;
• архітектурний макет;
• переглянутий список елементів ризику і переглянутий бізнес-варіант;
• план розробки для всього проекту, що включає великоблочний проектний план і показує ітерації і критерій еволюції для кожної ітерації.

Етап Реалізація (Construction). Головне призначення етапу – створити програмний продукт, який забезпечує початкові операційні можливості.

Цілі етапу *Реалізація*:

• мінімізувати вартість розробки шляхом оптимізації ресурсів і усунення необхідності доробок;
• домогтися швидкого отримання прийнятної якості;
• домогтися швидкого отримання контрольних версій (альфа, бета і т. д.).

Основні дії етапу *Реалізація*:

• управління ресурсами, контроль ресурсів, оптимізація процесів;
• повна розробка компонентів і їх тестування (за сформульованим критерієм еволюції);
• оцінювання реалізацій продукту (із врахуванням критеріїв із специфікації).

В результаті етапу *Реалізація* створюються такі артефакти:

• програмний продукт, готовий для передачі користувачам;
• опис поточної реалізації;
• керівництво користувача.

Етап Впровадження (Transition). Головне призначення етапу – застосувати програмний продукт у середовищі користувачів і завершити реалізацію продукту. Етап починається з пред'явлення користувачам бета-версії продукту. У ній виявляються помилки, які коригуються у наступних бета-реалізаціях. Паралельно вирішуються питання розміщення, пакування та супроводу продукту. Після завершення бета-періоду тестування продукт вважається реалізованим.

Порядок виконання роботи

Реалізувати проект розробки програмного забезпечення згідно з виданим завданням. Сформувати команду, розподілити ролі. Виконати основні дії і створити артефакти, які відповідають певному етапу.

Література

1. Гагарина Л. Г., Кокорева Е. В., Виснадул Б. Д. Технология разработки программного обеспечения : учебное пособие / под ред. Л. Г Гагариной. М: ИД «ФОРУМ»-ИНФРА-М, 2008. 400 с. URL: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/TekhnologiyaRazrabotGagarina2008.pdf (дата звернення: 30.07. 2021).
2. Техническое задание на создание автоматизированной системы. ГОСТ 34.602-89. [Чинний від 1990-01-01]. 12 с. (Міждержавний стандарт).
3. Автоматизированные системы. Стадии создания. ГОСТ 34.601-90. [Чинний від 1991-01-01]. 10 с. (Міждержавний стандарт).
4. Виды, комплектность и обозначение документов при создании автоматизированных систем. ГОСТ 34.201-89. [Чинний від 1990-01-01] 8 с. (Міждержавний стандарт).
5. Технології програмування та створення програмних продуктів: конспект лекцій /укладач О. В. Алексенко. Суми : Сумський державний університет, 2013. 133 с.
6. Вендров А. М. Практикум по проектированию программного обеспечения экономических информационных систем : учеб. пособие. М. : Финансы и статистика, 2006. 192 с.
7. Орлов С. А., Программная инженерия. Учебник для вузов. 5-е издание обновленное и дополненное. Стандарт третьего поколения. СПб : Питер, 2016. 640 с.
8. Модели жизненного цикла программного обеспечения. URL: http://swebok.sorlik.ru/software_lifecycle_models.html (дата звернення: 10.06. 2021).
9. Spiral model. URL: http://en.wikipedia.org/wiki/Spiral_model (дата звернення: 10.06. 2021).
10. Канер Сэм, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. К. : «ДиаСофт», 2001. 544 с.

11. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002.492 с.
12. Кролл П., Крачтен Ф. Rational Unified Process – это легко. Руководство по RUP для практиков: Пер. с англ. М. : КУДИЦ-ОБРАЗ, 2004. 432 с.