ВСТУП

Лабораторні роботи виконуються згідно з робочою програмою дисципліни «Технологія створення програмних продуктів» підготовки бакалаврів спеціальності 122 «Комп'ютерні науки».

Мета практикуму — набуття студентами практичних навичок, поглиблення та закріплення теоретичних знань з питань процесного та об'єктно-орієнтованого підходів до проектування інформаційних систем (ІС), застосування методів аналізу для формування моделей логіки бізнес-процесів ІС та оволодіння методами проектування з використанням CASE-засобів ARIS і Rational Rose UML.

У результаті виконання лабораторних робіт студенти здобувають знання про інструменти і методи документування існуючих бізнес-процесів організації замовника програмних систем (ПС), підходи до тестування програмних комплексів ІС, а також про методи оцінювання характеристик якості ПС та перевірки їх відповідності вимогам.

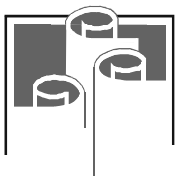
Студенти оволодівають такими базовими професійними компетенціями, як здатність до створення ІС; здатність до вибору стратегій планування життєвого циклу (ЖЦ) програмних продуктів; здатність до синтезу і розроблення вимог та специфікацій компонентів ІС; здатність до концептуального проектування компонентів ІС.

Лабораторні роботи структуровано за модульним принципом. Передбачено виконання трьох лабораторних робіт у модулі 1 «Методи та засоби розроблення програмних продуктів і стандарти програмної інженерії» і двох лабораторних робіт у модулі 2 «Гнучкі технології розроблення програмного забезпечення, тестування та документування процесів розроблення програмних продуктів». У процесі виконання робіт студенти ознайомлюються з технологіями процесного та об'єктно-орієнтованого моделювання і створюють проект ІС.

Кожну роботу студент виконує особисто згідно з індивідуальним номером варіанта завдання, який видає викладач. Порядок виконання роботи включає ознайомлення з теоретичним матеріалом, виконання передбаченого варіанта завдання, у тому числі: розроблення концепції та вимог до ІС, побудову діаграм бізнес-процесів

моделі проекту ІС, застосування методів та засобів тестування web-додатків, оформлення звіту і відповідей на запитання та завдання для самоперевірки. Звіт про виконання лабораторної роботи містить титульний аркуш, мету роботи, послідовність виконання, висновки.

До оформлення звіту ставляться такі вимоги: роботу оформляти на аркушах формату А4, на титульному аркуші вказати тему лабораторної роботи; назву дисципліни та номер варіанта; ким виконано та ким прийнято роботу. Хід роботи повинен містити вихідні дані та основні етапи їх перетворення; перелік специфікацій вимог до ІС; концептуальну, об’єктно-орієнтовану та процесну моделі ІС та висновки за результатами роботи.



Модуль 1

МЕТОДИ ТА ЗАСОБИ РОЗРОБЛЕННЯ ПРОГРАМНИХ ПРОДУКТІВ І СТАНДАРТИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Лабораторна робота 1.1

ВИЗНАЧЕННЯ, АНАЛІЗ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ В МЕЖАХ ЗАДАНОЇ ПРЕДМЕТНОЇ ОБЛАСТІ

Мета: оволодіння основними принципами визначення та моделювання вимог до програмного забезпечення (ПЗ) інформаційної системи роботи компанії та вивчення методів застосування діаграм варіантів використання в процесі постановки функціональних вимог до проектованої системи.

Основні теоретичні відомості

Збирання та аналіз вимог до програмного забезпечення

Вимоги — це властивості, які повинно мати ПЗ ІС для адекватного визначення функцій, умов та обмежень виконання, а також обсягів даних, технічного забезпечення і середовища функціонування.

Методи визначення вимог:

1) інтерв'ю, опитування, анкетування;
2) «мозковий штурм», семінар;
3) спостереження за виробничою діяльністю, «фотографування» робочого дня;
4) аналіз нормативної документації;
5) аналіз моделей діяльності;
6) аналіз конкурентних продуктів;
7) аналіз статистики попередніх версій системи.

Основна мета робочого процесу визначення вимог базується на тому, щоб направити процес розроблення на створення та отримання ІС, що відповідає вимогам замовника.

Використовують такі кроки робочого процесу визначення вимог:

– усвідомлення контексту ІС;

– визначення функціональних вимог;

– створення переліку можливих вимог;

– визначення нефункціональних вимог.

Перелік можливих вимог. Спочатку необхідно зібрати всі можливі вимоги та ідеї щодо майбутньої системи. Кожна пропозиція в переліку повинна мати: номер, коротку назву і зміст вимоги, опис вимоги, який може складатися зі стану пропозиції (наприклад, запропоновано, затверджено, відібрано тощо); вартість реалізації; пріоритет (наприклад, критичний, важливий або допоміжний); рівень ризику, зумовлений реалізацією пропозиції (наприклад, критичний, значущий або звичайний). Перелік вимог може змінюватися.

Усвідомлення контексту системи та побудова концепції ІС. Для того щоб правильно визначити вимоги, розробники системи повинні розуміти контекст (умови предметної області — ПрО), у якому працює система. Є два підходи до опису контексту системи: *моделювання предметної області* та *бізнес-моделювання*.

Модель предметної області описує важливі поняття області і їх зв'язки між собою. Не можна плутати модель ПрО з логічною або фізичною моделлю системи. Модель ПрО описує тільки її об'єкти, але не показує як ПС буде з ними працювати. Модель також дозволяє скласти глосарій системи для кращого її розуміння замовниками, користувачами та розробниками.

Бізнес-модель описує процеси (існуючі чи майбутні), які повинна підтримувати система. Кожну бізнес-модель можна подати як одну з підмоделей ПрО. Крім визначення бізнес-об'єктів, які задіяні в проекті, ця модель визначає працівників, їх обов'язки, а також дії, які вони повинні виконувати.

У результаті отримаємо *концепцію системи*, яку формулюють у вигляді опису призначення системи, об'єктів автоматизації та основних бажаних функцій ІС.

Визначення функціональних вимог. Підхід до визначення системних вимог ґрунтується на застосуванні варіантів використання системи (Use Cases), котрі охоплюють як функціональні, так і нефункціональні вимоги, що специфічні для конкретного варіанта використання.

Для користувача важливо, щоб система виконувала певні дії, при цьому користувач певним чином взаємодіє із системою, використовуючи її для своїх цілей. Таким чином, якщо визначити всі

можливі варіанти використання системи користувачем або іншими зовнішніми акторами чи процесами, то отримаємо множину функціональних вимог до неї.

Визначення нефункціональних вимог. До нефукціональних вимог належать такі властивості системи, як обмеження середовища і реалізації, ефективність і продуктивність, супроводжуваність, залежність від платформи (переносність), зручність використання (зрозумілість, виучуваність, керованість привабливість), точність, масштабованість, надійність, захищеність. Під надійністю розуміють такі характеристики, як середній час напрацювання на відмову, відновлюваність, кількість дефектів (помилок) на тисячу рядків програми, кількість помилок на клас та ін.

Вимоги до продуктивності — це швидкодія, пропускна здатність, час відгуку, використана пам'ять у процесі роботи ІС. Багато вимог до продуктивності можуть бути описані в конкретних варіантах використання (сценаріях), або в розділах, у яких описується робота всієї системи в цілому.

Для накладання нефукціональних обмежень на вимоги слід використовувати атрибути характеристик та підхарактеристик якості стандарту ISO/IEC 9126 (part 1).

Часто нефункціональні вимоги (обмеження) не можуть стосуватися конкретного варіанта використання (функції ІС) і мають бути занесені в окремий список додаткових вимог до системи (стандарт ISO/IEC 9126).

Під час процесу визначення вимог визначаються учасники процесу та розподіляються ролі. Типові приклади ролей:

Користувачі (Users) — це група, яка охоплює тих людей, які будуть безпосередньо використовувати ПС.

Користувачі можуть описати задачі, які вони вирішують чи будуть вирішувати за допомогою ПС, а також сподівання щодо підвищення атрибутів якості, які відображаються в користувацьких вимогах.

Замовники (Customers) — це особа або організація, що відповідають за замовлення програмного продукту, а також безпосередньо використовують ПП.

Аналітики (Market analysts) — особи, що відповідають за маркетинг, збирання та написання вимог.

Регулятори (Regulators) — особи, що регулюють області використання відповідно до керувальної документації, а також контролюють етапи та процеси формування вимог і розроблення програмного продукту.

Інженери-програмісти (Software Engineers) — особи, що розробляють програмний продукт, а є відповідальними за технічну оцінку шляхів вирішення поставлених задач та реалізацію вимог замовників.

Більшу частину користувацьких вимог визначають на стадії моделювання. Спочатку рекомендується виявити всіх зацікавлених осіб (акторів або Actors), потім визначити їх дії (варіанти використання, прецеденти або Use cases), а за необхідності — описати сценарії цих дій на відповідних діаграмах. Ці варіанти використання, по суті, і будуть вимогами до системи.

Уніфікована мова моделювання UML являє собою мову для визначення концепції, проектування і документування програмних, інформаційних, корпоративних та технічних систем.

Приклад формування опису вимог за допомогою UML

Виконаємо аналіз ПрО та побудуємо комплект Use Case діаграм для пояснення вимог до діяльності ІС у межах заданої у прикладі предметної області.

Предметна область «Діяльність поліклініки». У кожному районі міста є поліклініка, яка обслуговує пацієнтів. Поліклініка має номер ліцензії, назву, адресу, телефон, головного лікаря. Пацієнти записуються на прийом до лікарів-спеціалістів, а також до медсестер на процедури на певну дату і час у деякі кабінети. Пацієнти характеризуються прізвищем, ім’ям, по батькові, номером медичної картки, номером медичної страховки, адресою. Лікарі та медперсонал характеризуються спеціальністю, ПІБ, посадою, медичним стажем. Кабінети мають номер, назву, телефон, перелік медичного обладнання. Лікарі назначають пацієнтам лікування у вигляді ліків та процедур. Ліки мають назву, номер ліцензії Міністерства охорони здоров’я, ціну, правила вживання, термін придатності. Процедури мають назву, номер кабінету, список необхідних ліків, тривалість.

Загальна концепція системи. У ході детального аналізу ПрО стає очевидним, що основним завданням корпоративної ІС для організації є автоматизація керування поліклінікою, що включає в

себе реєстрацію відвідувачів, їх обслуговування (діагностування та лікування), формування звітності та наказів щодо діяльності установи. Такі висновки можна зробити з огляду на те, що специфіка роботи такої установи, як поліклініка, передбачає високий рівень деталізації даних та обсягів документування діяльності.

Очевидно, що оброблення інформаційних потоків даних і документації та її зберігання є пріоритетними завданнями з розроблення ІС, а тому доцільним є проектування системи, котра здатна автоматизувати процеси збирання та оброблення відповідних даних, їх форматування і подання для керівних органів організації.

Більшість документації, яка формується в ході діяльності поліклініки, має стандартизований вигляд і вимагає дотримання певних визначених законодавством правил, що має бути враховано в розробленні клієнтського ПЗ, що відповідає за структуру і форму відображення шаблонів під час заповнення їх користувачем.

З огляду на те, що основною сутністю процесу надання послуг поліклінікою є пацієнт, і саме він є об’єктом формування документів системи, то важливим є супровід системою документування діяльності установи, починаючи від моменту звернення пацієнта до поліклініки і до закінчення надання медичних послуг.

Необхідно враховувати, що медична установа потребує швидкого та ефективного реагування керівництва на відхилення в показниках діяльності, а отже ІС зобов’язана надавати вичерпну інформацію для формування керівництвом рішень впливу на діяльність установи.

Дані, якими оперує система, поділяються на два основні види: динамічні, які з’являються у процесі діяльності поліклініки (наприклад, картки пацієнтів, записи про виконані процедури тощо) та статичні, які відображають медичні послуги і не змінюються в ході діяльності. Тому необхідно передбачити в системі наявність розподілу між словниковим забезпеченням та даними, призначении для формування звітності.

Звітність має формуватися у вигляді стандартизованих відповідно до нормативних актів документів, а також у формі, необхідній для розроблення на підставі поданих даних рішень і наказів щодо діяльності; таким чином, система формування звітності потребує наявності шаблонів та можливості формування звітів змінної структури.

Оскільки певні дані в системі мають конфіденційний характер, важливою є наявність механізмів захисту даних та обмеження доступу до них, а відповідно система авторизації та аутентифікації і захисту даних від стороннього доступу.

Розгортання має бути запроектованим з розрахунку на те, що апаратна база може бути обмеженою, а отже, окремі логічні модулі та компоненти ІС можуть міститися на одній фізичній платформі.

Розроблення комплекту діаграм Use Case. Моделювання системи проводиться як спуск по рівнях: від концептуальної моделі до логічної і далі до фізичної моделі ПС.

Концептуальна модель системи виражається у вигляді діаграм варіантів використання (use case diagram). Цей тип діаграм слугує для проведення ітераційного циклу загальної постановки завдання разом із замовником.

Діаграми варіантів використання є основою для досягнення взаєморозуміння між програмістами, що розробляють проект системи, і замовниками проекту ІС.

Діаграми варіантів використання (Use case) іноді називають *діаграмами прецедентів* або *сценаріїв*. Варіант використання являє собою типову взаємодію користувача та проектованої системи і охоплює деяку очевидну для користувачів функцію, може бути як невеликим, так і досить великим, а також варіант вирішує деяке дискретне завдання користувача.

У найпростішому випадку варіант використання створюється у процесі обговорення з користувачами тих функцій, які вони хотіли б отримати від системи. Кожній окремій функції привласнюється ім’я і записується її короткий текстовий опис. Це все, що необхідно для фази аналізу.

Знання деяких деталей може знадобитися, якщо передбачається, що варіант використання містить важливі архітектурні відгалуження. Більшість варіантів використання можна деталізувати під час конкретної ітерації у процесі проектування.

Для проекту *«Діяльність поліклініки»* запропоновано варіанти використання, які подані основною діаграмою (рис. 1.1) та деталізацією певних варіантів використання, які можуть бути автоматизовані і тому потребують детального аналізу: реєстрація (рис. 1.2), діагностування (рис. 1.3), лікування (рис. 1.4), ведення звітності (рис. 1.5) та формування рішення (рис. 1.6).

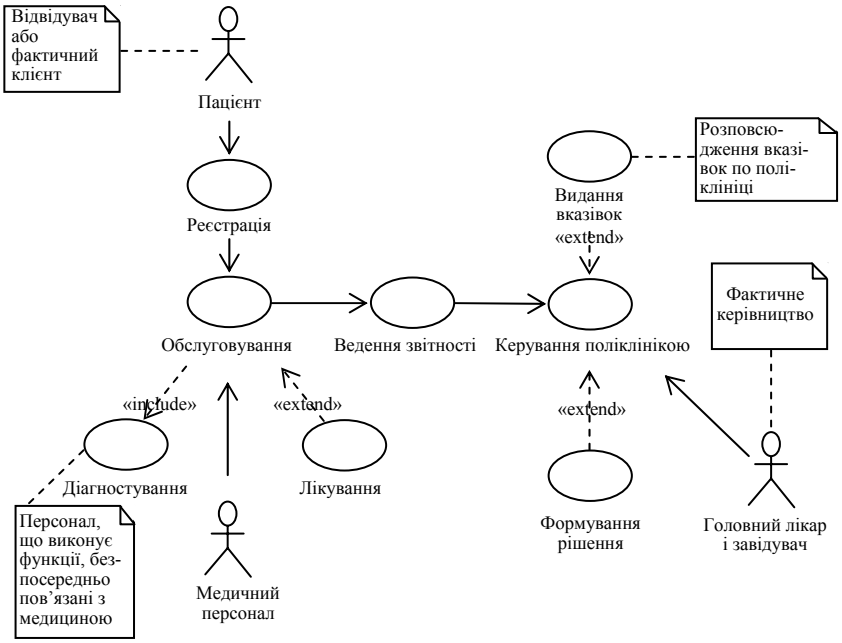


Рис. 1.1. Загальна діаграма моделі ІС

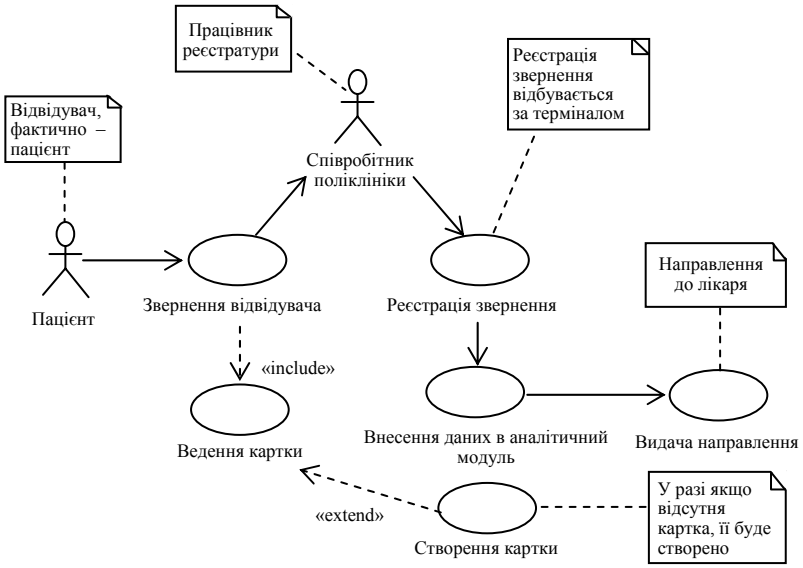


Рис. 1.2. Діаграма реєстрації звернень

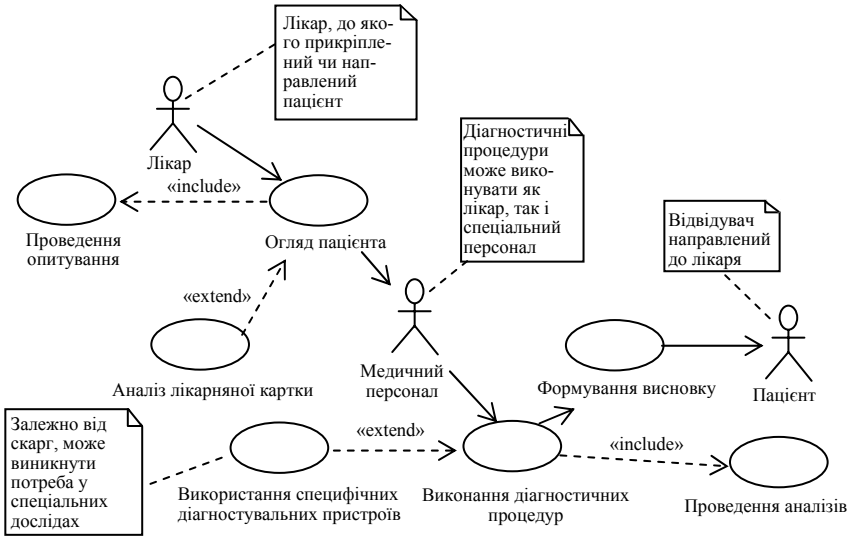


Рис. 1.3. Діаграма діагностування пацієнта

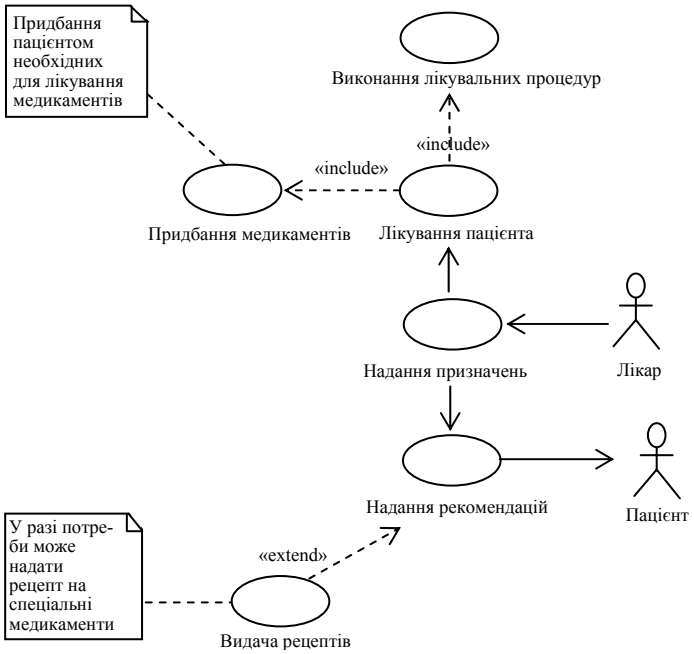


Рис. 1.4. Діаграма лікування пацієнтів

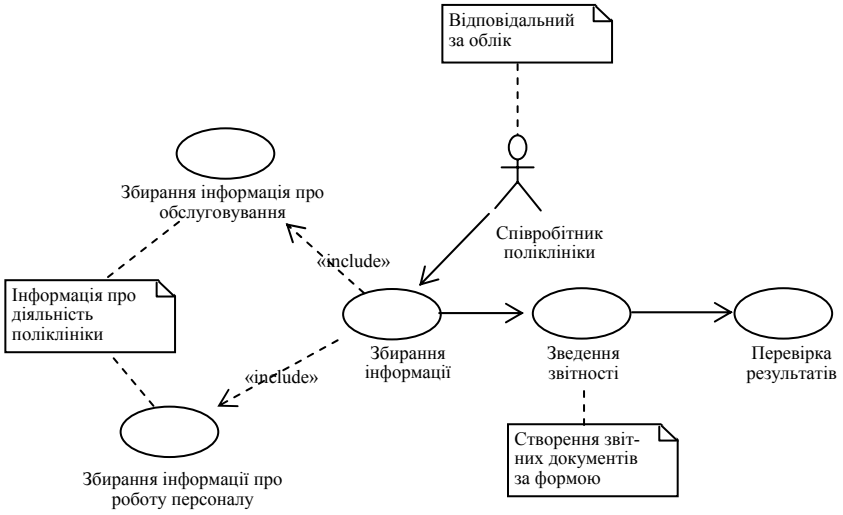


Рис. 1.5. Діаграма ведення звітності

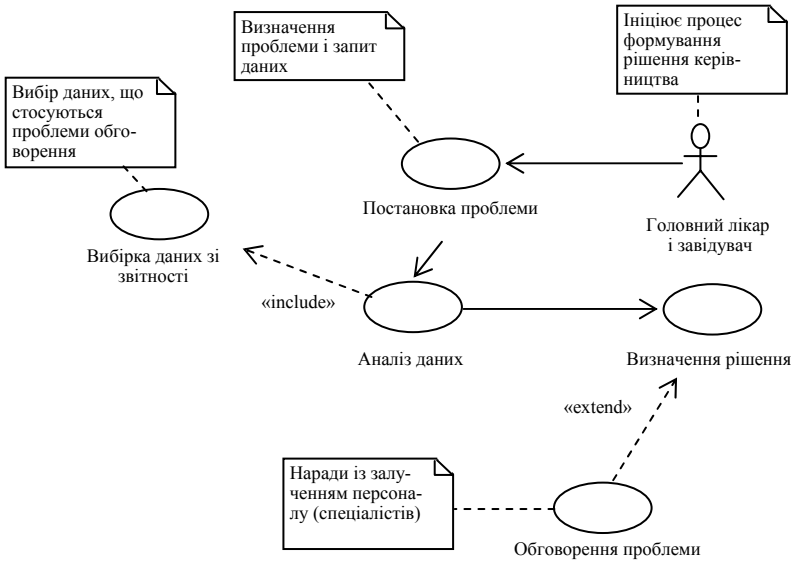


Рис. 1.6. Діаграма формування рішення

Реєстрація — прийом відвідувача, занесення даних у базу даних, у разі потреби — проведення консультацій і видача направлення до лікаря відповідно до запиту пацієнта.

Звернення відвідувача — звернення відвідувача до поліклініки, яке опрацьовує працівник реєстратури.

Реєстрація звернення — внесення причин звернення і дати в базу даних, для подальшого створення звітності.

Внесення даних в аналітичний модуль — формування користувачем запису та внесення його в аналітичний модуль.

Видача направлення — формування на основі скарг направлення до лікаря і передача його відвідувачу.

Ведення картки — передбачає зберігання і видачу картки пацієнту.

Створення картки — якщо картки немає, чи її загублено, то співробітник заведе для пацієнта нову.

Обслуговування — діагностування пацієнта на підставі його скарг та отримання заключення лікаря і спеціалістів, занесення отриманих даних у картку.

Діагностування — етап обслуговування, котрий полягає в проведенні досліджень стану пацієнта і пошуку методів лікування; зазвичай проводиться лікарем або спеціалістами.

Проведення опитування — опитування пацієнта щодо його скарг, симптомів, та ситуацій, у яких вони виявляються.

Огляд пацієнта — комплекс заходів щодо визначення стану пацієнта і необхідних дій з його покращення.

Аналіз лікарняної картки — передбачається вивчення історії хвороби пацієнта за його лікарняною карткою.

Виконання діагностичних процедур — використання спеціального устаткування для виконання процедур, які допоможуть встановити діагноз.

Використання специфічних діагностувальних пристроїв — застосування пристроїв, котрі діагностують суто специфічні хвороби, або можуть мати негативні наслідки для здоров’я.

Проведення аналізів — збирання даних про організм пацієнта за матеріальними зразками.

Формування висновку — створення лікарем клінічної картини хвороби і визначення методів її лікування.

Лікування — етап обслуговування, який полягає в наданні допомоги пацієнту, проведенні лікувальних процедур, наданні медикаментів або видачі рецептів.

Видача медикаментів — надання необхідних медикаментів пацієнту в разі крайньої потреби або через особливість їх дії і необхідність нагляду.

Виконання лікувальних процедур — спеціалізовані процедури, що проводяться за допомогою спеціального обладнання і направлені на покращення стану пацієнта.

Лікування пацієнта — комплекс заходів щодо лікування.

Надання призначень — визначення послідовності і необхідних дій щодо лікування.

Надання рекомендацій — поради лікаря щодо лікування або запобігання хворобам, до яких має прислухатись пацієнт.

Видання рецептів — видача рецептів лікарем на спеціалізовані медикаменти.

Ведення звітності — накопичення і класифікація інформації, яка згодом використовується для формування звітності і надання інформації керівництву про стан справ в установі.

Збирання інформації про обслуговування — збирання даних, що стосуються безпосередньої діяльності поліклініки, тобто діагностування та лікування населення.

Збирання інформації про роботу персоналу — збирання даних щодо скарг на персонал поліклініки, ефективність його роботи і виконання своїх службових обов'язків.

Збирання інформації — комплекс заходів щодо збирання різноманітної інформації, що стосується роботи поліклініки.

Зведення звітності — зведення звітності за зібраними даними до певної конкретно заданої форми, властивої установи.

Перевірка результатів — верифікація форми та змісту звітності контролерами.

Видання вказівок — інструмент керівництва, створення вказівок, які впливають на роботу установи.

Керування поліклінікою — використання повноважень керівництва для забезпечення оптимального виконання поліклінікою свого призначення.

Формування рішення — аналіз даних та погодження методів вирішення проблеми керівництвом із залученням спеціалістів.

Вибірка даних зі звітності — обрання даних, необхідних для розгляду конкретної проблеми.

Постановка проблеми — визначення вирішуваної проблеми.

Аналіз даних — аналіз наявної проблеми, можливостей для її вирішення, визначення даних та способів розв'язання.

Визначення рішення — комплекс обговорень та обрання оптимального вирішення проблеми за наявних ресурсів.

Обговорення проблеми — наради за участю не тільки керівництва, а і причетних осіб з персоналу.

Розглянемо дійові особи на діаграмах.

Головний лікар і завідувачі відділеннями — фактичне керівництво поліклініки, яке повинно забезпечувати роботу даної установи.

Лікар — особа, котра виконує основні медичні функції — діагностування і лікування пацієнтів.

Медичний персонал — загальна кількість працівників поліклініки, котрі виконують медичні функції; до них належать як лікарі, так і молодший медичний персонал та інші спеціалісти.

Співробітники поліклініки — кількість працівників поліклініки, котрі виконують немедичні функції, наприклад, консультацію відвідувачів, ведення звітності тощо.

Пацієнт — відвідувач, котрий звертається до поліклініки, її фактичний клієнт.

Значущість концептуальної моделі в процесі розроблення ПП важко переоцінити. Ця модель безпосередньо поєднує погляд аналітика на систему зі сподіваннями замовника. Чим менше вони різняться, тим більше шансів проекту бути успішним. Діаграми моделі повинні бути належним чином складені та документовані, щоб дати змогу не спеціалісту проаналізувати запропоновану модель системи. Тільки якісне проектування концептуальної моделі здатне забезпечити позитивний ефект від розроблення ІС.

Завдання

Варіант завдання з описом ПрО видається студенту викладачем (див. додаток). Необхідно виконати аналіз ПрО, виділити її основні характерні ознаки та об'єкти і описати результати аналізу базових функцій (сценаріїв) ІС. Особливо слід звернути увагу на ті сценарії, які підлягають автоматизації в межах ІС (можуть стати компонентами ПС), яка буде функціонувати в заданій ПрО, та виконати їх детальний аналіз.

Спочатку слід обрати методи визначення вимог та обґрунтувати й описати ці методи. Потім скласти перелік вимог до ІС, пронумерувати кожну з них і привласнити ім'я (назву). Після цього складається текстовий опис вимоги, що описується таким чином:

а) кожна вимога має містити деяку *функціональну потребу*;

б) на функціональну вимогу з п. а) мають накладатися обмеження, сформовані на основі *нефункціональних вимог* (у загальному випадку розглядають декілька *нефункціональних обмежень* на одну функціональну вимогу).

Для виконання завдання виконати такі дії:

1) провести аналіз ПрО та описати основні бажані на ваш погляд функції створюваної ІС;

2) створити концепцію ІС у вигляді опису призначення системи, об'єктів автоматизації та основних функцій ІС;

3) побудувати діаграми варіантів використання для заданої ПрО і описати базові функції та дійових осіб проекту ІС;

4) виконати детальний опис розроблених діаграм сценаріїв та описати базові функції системи;

5) виявити учасників (акторів), які будуть працювати з ІС, та охарактеризувати їх;

6) проаналізувати методи визначення вимог, які можна застосувати до ПрО, та описати їх;

7) застосувати обрані методи визначення вимог і сформувати перелік вимог до створюваної ІС та їх опис, використовуючи вказані вище рекомендації.

Вимоги до звіту

Для виконання лабораторної роботи потрібно використати CASE-засіб IBM Rational Rose UML (версії 2007 р. або наступні) та систему створення діаграм Microsoft Visio (версії 2010 р. або наступні — продукт Microsoft Office). Звіт про лабораторну роботу оформити за допомогою Microsoft Word. Він має містити: 1) титульний аркуш з відомостями про виконавця; 2) повний опис варіанта ПрО; 3) аналіз ПрО та концепцію ІС у вигляді опису призначення системи, об'єктів автоматизації та основних функцій ІС; 4) діаграму сценаріїв (use case diagram) базових функцій ІС та діаграми декомпозиції тих сценаріїв (функцій), які мають бути автоматизовані в межах ПС (загалом не менше чотирьох діаграм); 5) детальний опис розроблених діаграм сценаріїв; 6) опис застосованих

методів для збирання вимог; 7) перелік та аналіз визначених вимог; 8) загальні висновки про лабораторну роботу.

Взаємозв'язки у циклі лабораторних робіт

За допомогою Rational Rose UML у лабораторній роботі 1.1 слід створити *об'єктно-орієнтований проект ІС* для заданої ПрО, у якому реалізувати *діаграми варіантів використання* (не менше як чотири діаграми). У лабораторній роботі 1.2 проект доповнюється трьома *діаграмами класів* з метою опису логічної моделі ІС.

У лабораторній роботі 1.3 використовуються *логічна модель* системи (*діаграми класів*) та *діаграми сценаріїв* (інформація про акторів) для побудови *діаграм організаційної структури* організації, а також *діаграм носіїв інформації* (документів) організації, діяльність якої автоматизується шляхом створення ІС. Крім того, з лабораторних робіт 1.1 та 1.2 беруться *описання* та *специфікації* вимог і застосовуються у лабораторних роботах 1.3–1.5.

У лабораторній роботі 2.1 буде застосовано *головну діаграму сценаріїв* та інші три декомпозиції прецедентів (*варіантів використання*) з метою побудови двох *діаграм* базових процесів *доданої вартості* і двох *діаграм подієво-керованого процесу,* що описують ланцюжки процесів, які деталізують сценарії з головної діаграми сценаріїв, що описує базові функції ІС.

Запитання для самоперевірки

1. Які методи визначення вимог до ІС вам відомі?
2. Як визначають функціональні та нефункціональні вимоги?
3. Яким чином використовують діаграми Use case для формування специфікації вимог до ІС?

Література: [1–4]; [6–9].

Лабораторна робота 1.2

ВИЗНАЧЕННЯ ЛОГІЧНОЇ МОДЕЛІ ТА ФОРМАЛІЗОВАНИЙ ОПИС ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

***Мета*:** оволодіння основними принципами побудови логічної моделі інформаційної системи, опису специфікації вимог до програмної системи і створення технічного завдання (ТЗ) на розроблення системи.

Основні теоретичні відомості

Створення логічної моделі інформаційної системи

Побудова *діаграм класів* (class diagrams) є центральною ланкою методології об'єктно-орієнтованого аналізу та проектування. Діаграма класів відображає класи та їх взаємозв'язки, тим самим подаючи логічний аспект проекту. Кожна діаграма класів являє собою певний ракурс структури класів. На стадії аналізу діаграми класів використовують для виділення загальних ролей та обов'язків сутностей, які забезпечують необхідну поведінку системи. На стадії проектування діаграми класів застосовують для передачі загальної структури класів, що формують архітектуру ПС.

Формування вимог. У діаграмі класів описуються об'єкти, які мають бути у ПС, відповідно до поставлених до неї вимог, та задається поведінка об'єктів через опис функцій відповідних класів.

Крім того, діаграми класів визначають типи об'єктів системи та різні статичні зв'язки, які існують між ними. Є два основні види статичних зв'язків:

– асоціації (наприклад, менеджер може вести кілька проектів);

– підтипи (працівник є різновидом особистості).

На діаграмах класів зображуються також атрибути класів, операції (функції класів) та обмеження, які накладаються на зв'язки між об'єктами.

Модель класів ІС «Поліклініка» у своїй структурі спирається на основну концепцію системи, тобто на сценарії збирання, оброблення даних і формування регулювальних рішень на основі діяльності установи. Відповідно діаграма класів включає в себе класи, що виконують ці завдання, а у своїй діяльності спираються на класи пакетів *«Реєстрація відвідувачів»* та *«Обслуговування пацієнтів»*. Відповідні пакети під час своєї роботи заповнюють даними базу даних, у якій формується вибірка даних для редактора звітів, що забезпечує інструментарій створення звітів для користувача.

Звіти, отримані під час оброблення інформації про діяльність, надаються керівництву для перегляду, що дозволяє йому формувати власні рішення на базі адекватної актуальної оцінки стану організації.

Реєстрація передбачає отримання даних від пацієнта-відвідувача і внесення даних у вигляді запиту в систему, де аналітичний модуль, спираючись на алгоритми оброблення даних, формує для пацієнта направлення.

Дані, отримані на етапі реєстрації, заносяться в картку пацієнта і слугують вхідними даними для етапу діагностування, а отже, і лікування, де після виконання процедур звіти про результати і супутні документи заносяться в картку, а отже, і в базу даних ІС.

Таким чином, очевидно, що діаграма класів описує ту частину ПрО, що належить до керування поліклінікою і виконує покладене в основу концепції завдання.

Для реалізації подібної системи необхідна достатньо складна ПС, яка здатна реалізовувати потрібні для функціонування корпоративних ІС складні алгоритми.

Користувачі розподілені за ролями, які вони відіграють у системі, тобто до множини користувачів включені і керівники — головний лікар і завідувачі відділень, а також спеціалісти, які виконують обслуговування, або персонал, який виконує адміністративні та облікові функції.

Перед побудовою діаграм класів необхідно визначити потенційні сутності, які можна подати у вигляді класів: пацієнт, поліклініка, база даних, словник, картка пацієнта, головний лікар, лікар, начальник відділу, спеціаліст, звіт, редактор звітів, фінансовий звіт, звіт про діяльність, рахунок, медикаменти, аналітичний модуль, модуль уведення, модуль виведення, направлення, запис про виконання процедури, запис відвідування, рецепт, наказ, процедура, обладнання, кабінет, розклад, вибірка даних.

З огляду на те, що основною концепцією проектованої системи є збирання і подання даних про діяльність, необхідно виокремити з наведених сутностей значущі для виконання наведеної функції, а саме: користувач, керівник, спеціаліст, картка пацієнта, запис відвідування, запис про виконання процедури, рецепт, направлення, аналітичний модуль, модуль обліку, база даних, вибірка даних, редактор звітів, фінансовий звіт, звіт про діяльність.

Після детального аналізу отриманих сутностей можна виокремити для їх певних підмножин абстрактні батьківські класи: *звіт* для фінансового звіту і звіту про основну діяльність, *користувач* для керівника і спеціаліста, *запис* для запису виконання процедури та рецепту.

Отримаємо комплект діаграм (рис. 1.7–1.9), у якому описано структуру класів створюваної системи.

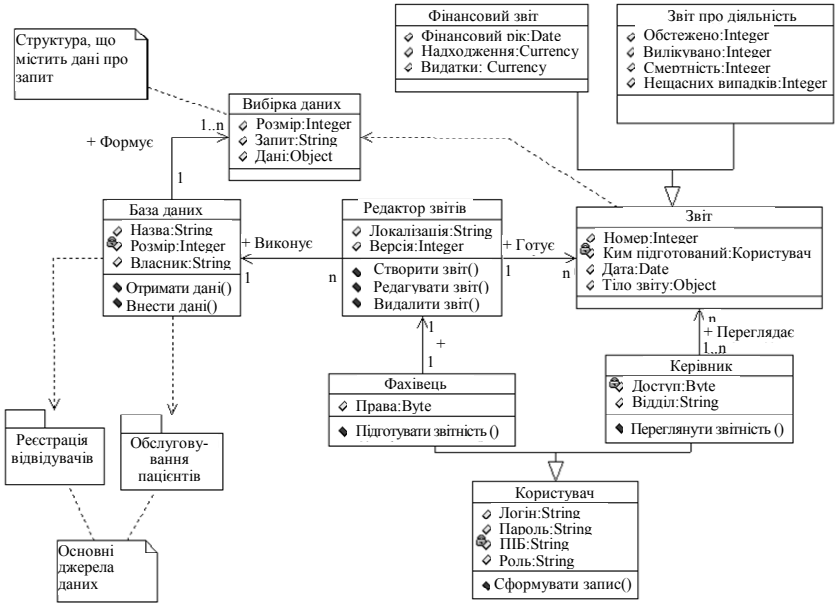


Рис. 1.7. Загальна діаграма класів системи

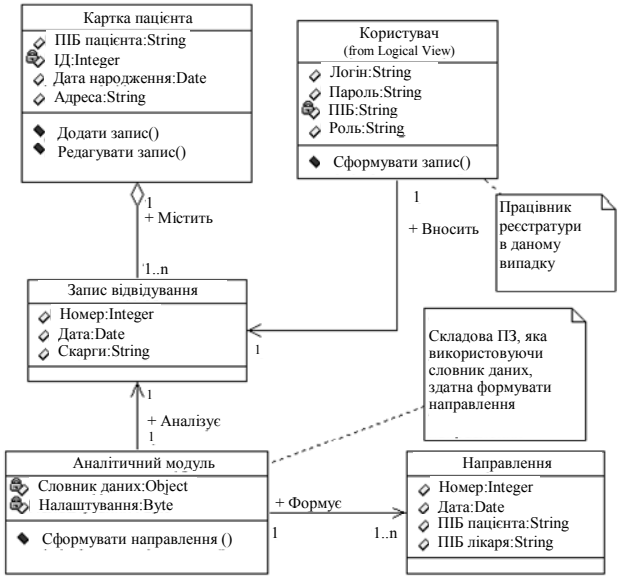


Рис. 1.8. Діаграма класів реєстрації відвідувачів

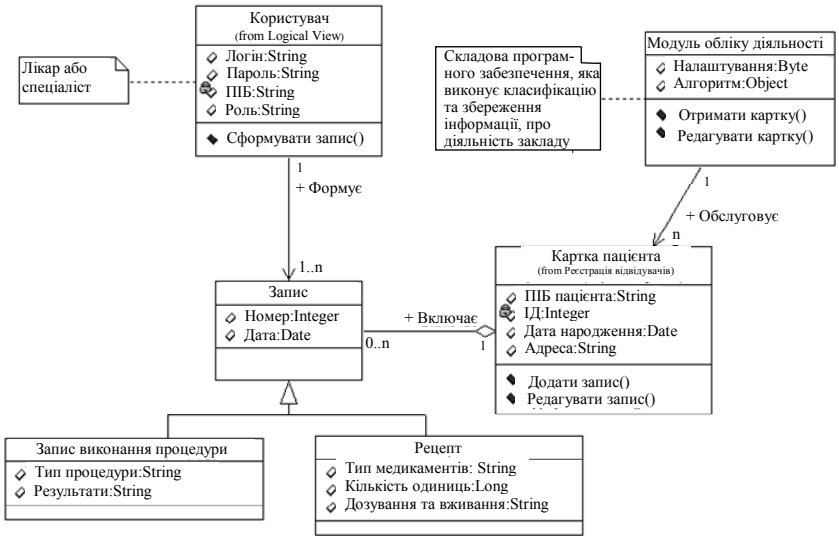


Рис. 1.9. Діаграма класів обслуговування пацієнта

Модель класів є засадничою для розроблення ПЗ згідно з об'єктно-орієнтованим підходом до проектування ІС, оскільки являє собою множину класів та взаємодію між ними в межах системи, що фактично і є сутністю об'єктно-орієнтованого програмування.

Очевидно, що в сучасних моделях розроблення ПЗ кожний наступний етап розроблення спирається на вихідні дані попереднього і відповідно якість виконання поточного етапу залежить від якості проведення попередніх. Тому не можна переоцінити значущість моделі класів у процесі розроблення, оскільки ця модель надає логічну структуру даних для побудови ПЗ. Зрозуміло, що чим точніше ця модель системи відображає реальний стан ПрО, тим більше шансів на створення успішного дієвого проекту ІС.

Керування вимогами до ПЗ ІС

Керування вимогами до ПЗ (Software Requirements Management) — це процес, що включає визначення, ідентифікацію, документування, аналіз, відстеження, позначення пріоритетності вимог, досягнення угоди за вимогами до ІС із замовником і потім керування змінами та повідомлення відповідних зацікавлених осіб. Керування вимогами — безперервний процес протягом усього ЖЦ розроблення ПЗ ІС.

Принципи і прийоми керування вимогами

Базова версія вимог. Щоб домовитися про зміну вимог, спочатку потрібно їх зафіксувати в первинному вигляді. *Базова версія* (baseline) — це набір функціональних та нефункціональних вимог, які розробники зобов'язалися реалізувати у певній версії (ітерації) ПС.

Процедури керування вимогами. Процедури керування вимогами базуються на таких інструментах:

– заходах, прийомах, угодах про керування версіями різних документів вимог і окремих вимог;

– правилах складання базової версії вимог;

– статусах вимог, які будуть використовуватися, і категоріях осіб, які мають право змінювати їх;

– способах, за допомогою яких нові вимоги і зміни існуючих вимог пропонуються, обробляються, обговорюються і передаються зацікавленим особам;

– методах аналізу впливу запропонованої зміни;

– відстеженні зв'язків планів і зобов'язань проекту зі зміною вимог.

Контроль версій. Кожна версія документа вимог повинна містити історію перероблення, де вказуються внесені зміни, дата кожної з них, особа, яка внесла зміну, а також причина. Доцільно додавати номер версії до назви кожної окремої вимоги, який можна послідовно збільшувати під час модифікації вимоги.

Для документування версій використовуються текстові процесори, електронні таблиці. Існують спеціалізовані засоби та інструменти для контролю версій і конфігурацій (у тому числі CASE-засоби).

Атрибути вимог. Із позицій керування кожна з вимог являє собою самостійний об'єкт. Зміни здійснюються в описовій частині об'єкта. Контролювати зміни зручніше за допомогою атрибутів вимог. Набір атрибутів підбирають для кожного проекту індивідуально, виходячи з максимальної результативності для команди проекту. Для першого впровадження засобів керування змінами вимог рекомендується використовувати не більше ніж п'ять атрибутів. Цю кількість можна розширити згодом, коли команда «увійде у смак» процесу, а також якщо додавання атрибутів виправдано практичними міркуваннями.

Як шаблон опису атрибутів вимог до ПЗ К. Вігерс пропонує такий набір даних [2]:

– дата створення вимоги та номер її поточної версії;

– автор вимоги та особа, відповідальна за задоволення вимоги;

– відповідальна особа за вимогу або список зацікавлених осіб (щоб приймати рішення про запропоновані зміни);

– стан вимоги;

– походження або джерело вимоги;

– логічне обґрунтування вимоги;

– підсистема (або підсистеми), для якої призначена вимога;

– номер версії продукту, для якого призначена вимога;

– метод перевірки або критерій тестування прийнятності;

– пріоритет реалізації та стабільність вимоги.

Контроль статусу вимог. В автоматизованих засобах керування проектами для контролю ступеня виконання тієї чи іншої роботи використовується поняття ступеня виконання (progress), яке виражається у відсотках. Такий спосіб рідко застосовують у програмістських розробках, у яких через їх слабку формалізованість важко оцінити роботу у відсотках. Для керування вимогами рекомендується оперувати не відсотком, а статусом. К. Вігерс [2] пропонує шаблон для визначення статусу вимоги (табл. 1.1).

Таблиця 1.1

Статус вимоги

Статус	Опис
Proposed (запропоновано)	Вимогу викликано та задано авторизованим джерелом
Approved (одобрено)	Вимогу проаналізовано, її вплив на проект прораховано, і вона була розміщена в базовій версії певної версії продукту. Ключові зацікавлені в проекті особи погодилися з цією вимогою, а розробники ПЗ зобов’язалися реалізувати її
Implemented (реалізовано)	Код, який реалізує вимога, розроблений, написаний і протестований. Вимогу відстежено до відповідних елементів дизайну та коду
Verified (перевірено)	Коректне функціонування реалізованої вимоги підтверджено у відповідному продукті. Вимогу відстежено до відповідних варіантів тестування. Тепер вимога вважається завершеною

Закінчення табл. 1.1

Статус	Опис
Deleted (вилучено)	Затверджену вимогу видалено з базової версії. Описано причини видалення і названий той, хто прийняв це рішення
Rejected (відхилено)	Вимогу запропоновано, але не заплановано для реалізації в жодній з майбутніх версій. Описано причини відхилення вимоги і названий той, хто прийняв це рішення

Мова специфікації вимог

Використання чіткої і зрозумілої мови (узгоджених термінів) для написання вимог дозволяє істотно полегшити подальше розуміння вимог та їх класифікацію. Як приклад можна навести використання в тексті слова «повинен» («має») як ключового слова, що позначає наявність вимоги. Деякі підходи навіть вказують на використання ключових слів, що відрізняються одне від одного, для характеристики пріоритету вимог; наприклад, «повинен», «має» — must, «рекомендується» — should і «можливо» — may.

Мова, використовувана для написання вимог, значною мірою залежить від «рівня» документа з вимогами, тобто існує принципова відмінність між призначеними для користувача вимогами, які відносяться до ПрО, і системними вимогами, які відносяться до області рішень.

Призначені для користувача вимоги в основному описують можливості (послуги), необхідні користувачам (тобто потреби користувачів), або обмеження, пов'язані з цими можливостями чи потребами. У цьому випадку вимога, що описує таку потребу, повинна описувати тільки одну потребу, необхідну або для одного користувача, або для групи з декількох однотипних користувачів. При цьому в тексті вимоги потрібно вказувати тип користувача. Наведемо типову вимогу, що описує можливість (потребу).

<Тип користувача> *повинен мати можливість* **<опис можливості>**

Якщо існують певні вимоги до продуктивності або обмеження, пов'язані тільки з однією конкретною вимогою, тоді текст вимоги може бути доповнений і набувати такого вигляду:

<Тип користувача> *повинен мати можливість* **<опис можливості>** *з* **<показник продуктивності>** *від* **<момент відліку>** *функціонуючи в* **<умови експлуатації>**

Сформульована загальна вимога в окремому випадку може виглядати так (містити умови продуктивності й обмеження):

<Оператор> *повинен мати можливість* **<зробити постріл>** *протягом* **<3 секунд>** *від* **<моменту виявлення мети радаром>** *функціонуючи у* **<складних морських умовах>**

Набагато рідше трапляється ситуація, коли атрибут з однією і тією самою умовою продуктивності пов'язаний з декількома різними вимогами. Можна уявити, наприклад, ситуацію, коли декілька різних вимог характеризуються одним і тим самим часовим параметром. Проте на практиці, коли існує ієрархія вимог і вимоги нижчого рівня є деталізацією вимоги вищого рівня, це часто означає, що необхідне значення атрибута продуктивності фактично пов'язане з вимогою вищого рівня, а отже, всі вимоги, що є його деталізацією, просто успадковують те саме значення атрибута.

Часто обмеження описуються не в тексті самої вимоги, що описує можливості (потреби), а окремо. Це може бути зумовлено тим, що таке обмеження або накладається цілком на всю систему і немає сенсу багато разів повторювати його в кожній вимозі, або, навпаки, ці обмеження стосуються різних аспектів системи, а тому їх необхідно виділяти для подальшого контролю.

Зазвичай обмеження в призначених для користувача вимогах згадуються або як мінімально прийнятні параметри продуктивності, що заявляються замовником, або є наслідком необхідності взаємодії створюваної системи з оточенням (сюди ж, що характерно, належать правові та соціальні системи). Вимога типу обмеження зазвичай виражається в такій формі:

<Тип користувача> *не повинен підлягати дії* <**відповідне законодавство>**

Приклад з реального життя:

<Водій швидкої допомоги> *не повинен підлягати дії* **<законодавства, що передбачає відповідальність за порушення правил дорожнього руху>**

Оскільки обмеження пов'язані з областю рішень, то мова системних вимог відрізняється від мови вимог, призначених для замовника ПП або користувача. Під час формулювання системних вимог основний акцент робиться на опис системних функцій і побудову обмежень. Конкретне формулювання вимоги залежить також від типу обмеження або показника продуктивності, які

пов’язані з цією вимогою. Наведемо загальний приклад опису функції (системна вимога), який містить потрібне значення показника продуктивності (у наведеному випадку — навантаження):

<Система> *повинна* **<виконувана функція>** *не менше ніж* **<кількість> <об’єкт>** *функціонуючи в* **<умови експлуатації>**

Або в реальній ситуації:

<Телекомунікаційна система> *повинна* **<забезпечувати телефонний зв’язок>** *не менше ніж з* **<10> <абонентами>** *функціонуючи в* **<умовах відсутності зовнішнього джерела електроживлення>**

Наведемо інший приклад, що описує періодичне обмеження:

<Система> *повинна* **<виконувана функція> <об’єкт>** *кожні* **<показник продуктивності> <одиниця вимірювання>**

Мовою інструкції це виглядає так:

<Кава-машина> *повинна* **<виготовляти> <гарячий напій>** *кожні* **<10> <секунд>**

Шаблони вимог

Розпочнемо обговорення теми про мову написання вимог з використанням шаблонів, зокрема про збирання і формулювання вимог типу обмеження.

Використання шаблонів, як це показано на прикладах, є дієвим способом стандартизації мови, вживаної для розроблення вимог. Для того щоб мати можливість писати стандартним способом вимоги різних типів, достатньо лише зібрати певний набір таких шаблонів. У процесі застосування цього набору на практиці його можна уточнювати, розширювати і коригувати з тим, щоб отримати повніший набір шаблонів, який надалі використовуватиметься і в інших проектах.

Таким чином, процес створення вимог за допомогою шаблонів можна поділити на два етапи:

1) вибір найбільш відповідного шаблону із загального набору шаблонів;
2) підставляння конкретних даних для заповнення порожніх полів у шаблоні.

Такий підхід дає змогу відокремити шаблон і дані, що підставляються в нього. Тоді будь-яка вимога міститиме посилання на шаблон, а дані фіксуватимуться окремо — як атрибути цієї вимоги.

«Шаблонний» підхід дозволяє згенерувати текстове подання вимоги в будь-який потрібний момент. Використання виділених шаблонів має такі переваги:

– можливість глобальної зміни стилю: для зміни формулювання певних вимог необхідно внести зміну тільки до одного або декількох конкретних шаблонів, задіяних у цій схемі;

– можливість легшого оброблення інформації: наприклад, виділення всіх полів, що пов'язані з **<умови експлуатації>**, в окремий атрибут вимог дозволяє зручніше фільтрувати і сортувати вимоги, виходячи з конкретних ознак *умови експлуатації*;

– можливість захисту конфіденційної інформації: у тому випадку, коли вимоги містять конфіденційну або секретну інформацію, шаблони можуть бути використані для захисту саме тієї частини тексту вимоги, доступ до якої повинен бути захищений.

Розглянуті шаблони вимог ілюструє рис. 1.10.

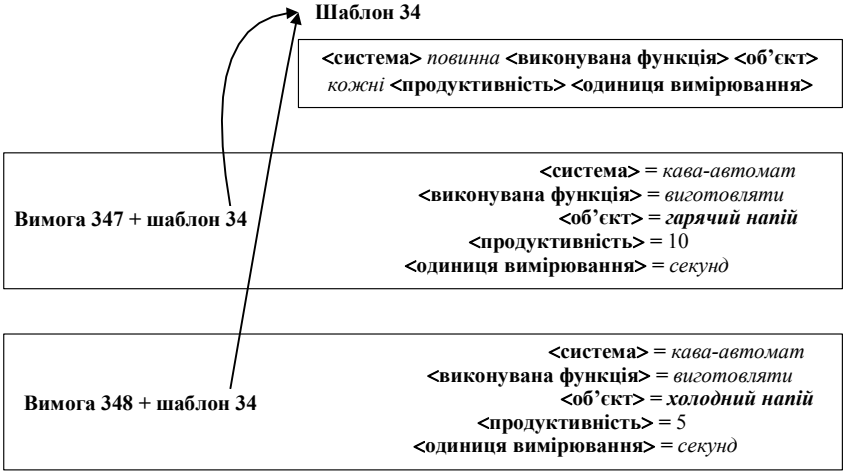


Рис. 1.10. Глобальні шаблони вимог

Останній пункт, поза сумнівом, потребує деякого уточнення. В оборонних і деяких комерційних проектах необхідно обмежувати доступ, але не до всієї інформації, а лише до деякої її частини. Дуже часто текст однієї вимоги містить інформацію про різні рівні секретності. Наприклад, абсолютно очевидно (не таємно), що сучасний військовий корабель буде оснащений ракетами, які він за-

пускатиме, проте подальша інформація про продуктивність може мати закритий характер: можлива кількість запусків за одиницю часу, радіус ураження і т. ін.

Замість того, щоб обмежувати доступ до цілої вимоги тільки через те, що деяка його частина є конфіденційною, такий підхід дає змогу дозволити переглядати всю вимогу, але без доступу до конфіденційної інформації, що міститься в деяких його атрибутах. Насправді, використовуючи цей підхід, різні учасники проекту (з різним рівнем доступу) можуть бачити різні набори атрибутів. Одним з найскладніших з погляду формулювання і одним з найбільш поширених типів вимог є обмеження. Так, саме для обмежень метод шаблонів істотно полегшує завдання формулювання вимог.

Пропонується такий підхід для формулювання обмежень:

1. Спочатку зібрати усі можливі вимоги.

2. Підготувати список обмежень різних типів, які можливі під час роботи над проектом. Якщо цей список ґрунтується на попередньому досвіді виконання аналогічного проекту, тоді вже є повний набір шаблонів з обмеженнями, який можна використовувати для опису обмежень поточного проекту. У противному випадку доведеться розробляти нові шаблони.

3. Стосовно кожної вимоги розглянути повний перелік можливих обмежень зі списку і визначити саме ті з них, які потрібно застосувати (зафіксувати) для цієї вимоги.

Для виконання цієї процедури зручно використовувати таблицю. Стовпчики відповідатимуть різним типам обмежень, а рядки — обмеженням. Якщо для вимоги необхідно додати обмеження певного типу, то у відповідній комірці необхідно сформулювати це обмеження, якщо ж потреби в обмеженні немає, то у відповідній комірці поставити «ні» (або N/A = not available).

4. Для кожного обмеження вибрати найбільш відповідний для нього шаблон і сформулювати вимогу з його допомогою.

5. Процес закінчується після заповнення всіх елементів таблиці.

Використання такого підходу дозволяє відповісти на два питання, що часто задаються:

– як сформулювати вимогу, у якій повинне бути обмеження (відповідь: використовувати шаблони);

– як переконатися в тому, що всі обмеження враховані (відповідь: використовувати таблицю для аналізу покриття вимог з обмеженнями).

Наведемо приклади шаблонів для вимог з обмеженнями. Для одного типу обмежень можуть використовуватися різні шаблони, а обмеження можуть мати складну класифікацію. У наборі прикладів шаблонів безпосередньо до самого *обмеження* належить *тільки той текст*, що виділений жирним курсивом.

Тип обмеження ***Шаблон***

Продуктивність/можливість **<Система>** *повинна* **<виконувана функція> <об'єкт>** ***не менше ніж <продуктивність> разів в <одиниця вимірювання>***

Продуктивність/можливість **<Система>** *повинна* **<виконувана функція> <об'єкт>** ***типу <характеристика> протягом <продуктивність> <одиниця вимірювання>***

Продуктивність/потужність **<Система>** *повинна* **<виконувана функція>** ***не менше ніж <кількість>*** **< об'єкт>**

Продуктивність/своєчасність **<Система>** *повинна* **<виконувана функція> <об'єкт>** ***протягом <продуктивність> <одиниця вимірювання> з моменту <подія>***

Продуктивність/періодичність **<Система>** *повинна* **<виконувана функція>** ***не менше ніж <кількість>*** **<об'єкт>** ***протягом <продуктивність> <одиниця вимірювання>***

Здатність до взаємодії/потужність **<Система>** *повинна* **<виконувана функція> <об'єкт>, *що складається з не менше ніж <продуктивність> <одиниця вимірювання > з <зовнішня сутність>***

Стійкість/періодичність **<Система>** *повинна* **<виконувана функція> <об'єкт>** *з* ***<продуктивність> <одиниця вимірювання> кожні <продуктивність> <одиниця вимірювання>***

Оточення/працездатність **<Система>** *повинна* **<виконувана функція> <об'єкт>** ***функціонуючи в <умови експлуатації>***

Приклад. У лабораторній роботі 1.1 на рис. 1.5 виконано аналіз функцій сценарію «Ведення звітності» у поліклініці. До функції «Зведення звітності» можна поставити таку вимогу:

Стійкість/періодичність **<Автоматизована система поліклініки>** *повинна* **<зводити звітність за зібраними даними до> <форми>** *з продуктивністю* ***<1 форма> <у неділю>*** *кожний* ***<1> <місяць>***

Технічне завдання

Техні́чне завдання́ — вихідний документ для проектування споруди або промислового комплексу, конструювання технічного

пристрою (приладу, машини, системи керування тощо), розроблення автоматизованої системи, створення програмного продукту або виконання науково-дослідних робіт, відповідно до якого проводяться виготовлення, приймання під час уведення в дію та експлуатація відповідного об’єкта.

Згідно з ГОСТ 34.602–89 ТЗ є основним документом, що визначає вимоги і порядок створення (розвитку або модернізації) ІС, відповідно до якого її розробляють і приймають під час уведення в дію. Згідно з чинними стандартами ТЗ має містити такі відомості про об’єкт розроблення:

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ.
    1.1. Повне найменування системи та її умовне позначення.
    1.2. Номер договору (контракту).
    1.3. Найменування організації-замовника та організацій-учасників робіт.
    1.4. Перелік документів, на підставі яких створюється система.
    1.5. Планові терміни початку і закінчення роботи зі створення системи.
    1.6. Джерела і порядок фінансування робіт.
    1.7. Порядок оформлення і подання замовнику результатів робіт зі створення системи.
    1.8. Перелік нормативно-технічних документів, методичних матеріалів, використаних під час розроблення технічного завдання.
    1.9. Визначення, позначення і скорочення.
2. ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ.
    2.1. Призначення системи.
    2.2. Цілі створення системи.
3. ХАРАКТЕРИСТИКА ОБ’ЄКТА АВТОМАТИЗАЦІЇ.
4. ВИМОГИ ДО ПРОГРАМИ.
    4.1. Вимоги до функціональних характеристик.
    4.2. Вимоги до надійності.
    4.3. Умови експлуатації програми.
    4.4. Вимоги до складу і параметрів технічних засобів.
    4.5. Вимоги до інформаційної та програмної сумісності.
    4.6. Вимоги до маркування та упаковки.
    4.7. Вимоги до транспортування та зберігання.
5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ.
6. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ.

7. СТАДІЇ І ЕТАПИ РОЗРОБЛЕННЯ.
8. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ.
8.1. Види випробувань.
8.2. Загальні вимоги до приймання роботи.

Нормативні документи

1. ГОСТ 2.114–95. Единая система конструкторской документации. Технические условия.

2. ГОСТ 19.201–78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.

3. ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

4. *IEEE Std 830–1993*. Recommended Practice for Software Requirements Specification. — http://www.ieee.org.

Більш детальну інформацію за ГОСТ 34.602 та іншими стандартами можна знайти за посиланням: http://www.rugost.com/. Під час розроблення технічного завдання його зміст розробляється відповідно до розроблення проекту ІС у межах заданої ПрО. Технічне завдання можна описувати засобами *MS Word* або іншого редактора, а також відповідними CASE-засобами.

Опис стандарту IEEE Std 830–93

Базовим стандартом, який визначає порядок і умови збирання та поставлення вимог до ПС, є міжнародний стандарт IEEE Std 830 — Recommended Practice for Software Requirements Specifications (SRS). Стандарт складається з таких розділів (нумерація відповідає плану опису SRS, а в дужках указані відповідні розділи і підрозділи власне стандарту IEEE Std 830, де описані рубрики цього плану):

1. Вступ (5.1): 1.1 Мета (5.1.1); 1.2. Область застосування (5.1.2); 1.3. Терміни, скорочення (5.1.3); 1.4. Посилання (5.1.4) ; 1.5. Короткий огляд (5.1.5).

2. Основна частина. Загальний опис (5.2). 2.1. Перспективи програмного продукту (5.2.1): інтерфейси системи (5.2.1.1); інтерфейси користувача (5.2.1.2); інтерфейси апаратних засобів (5.2.1.3); програмні інтерфейси (5.2.1.4); комунікаційні інтерфейси (5.2.1.5); обмеження пам’яті (5.2.1.6); дії, операції (5.2.1.7); вимоги до налаштування робочих місць (5.2.1.8). 2.2. Функції продукту (5.2.2).

2.3. Користувацькі характеристики (5.2.3). 2.4. Обмеження (5.2.4). 2.5. Припущення та залежності (5.2.5). 2.6. Розподіл вимог: C-вимоги — вимоги користувача (Customer requirements); D-вимоги — вимоги розробника (Developer requirements) (5.2.6).

3. Конкретні (детальні) вимоги (5.3). 3.1.Вимоги до зовнішнього, системного, користувацького, програмного, апаратного та комунікаційного інтерфейсів (5.3.1). 3.2. Функції системи (5.3.2). 3.3. Вимоги до виконання (вимоги продуктивності належать до нефункціональних вимог) (5.3.3). 3.4. Вимоги логіки бази даних (5.3.4). 3.5. Обмеження проекту (5.3.5). Угода щодо стандартів (5.3.5.1). 3.6.Характеристики програмного забезпечення системи (5.3.6). Надійність (5.3.6.1). Експлуатаційна готовність (5.3.6.2). Безпека (5.3.6.3). Супроводжуваність (5.3.6.4). Переносність (5.3.6.5). 3.7. Організація детальних вимог (5.3.7). Режим системи (5.3.7.1). Класи користувачів (5.3.7.2). Об'єкти (5.3.7.3) (5.3.7.2 і 5.3.7.3 належать до функціональних вимог через те, що у класах описані функції системи). Особливості (5.3.7.4). Вплив (5.3.7.5). Реакція (5.3.7.6). Функціональні ієрархії (5.3.7.7). 3.8. Додаткові коментарії (5.3.8).

C-вимоги — неформалізовані, а *D-вимоги* — детальні вимоги, які поділяються на функціональні та нефункціональні. Функціональні вимоги пов'язані з функціональними характеристиками системи (можливості, які повинна забезпечувати система), а нефункціональні вимоги пов'язані з якісними властивостями, які безпосередньо не стосуються функціональності ПС (обмеження, що пов'язані з характеристиками функціонування ПС).

D-вимоги складаються за допомогою функціональних специфікацій, що містять повний список усіх властивостей і функцій, які повинна мати ПС. Ці вимоги мають відповідати **C-вимогам**. Специфікація вимоги SRS має бути: несуперечливою; з можливістю контролю; тестованою; погодженою; повною.

Є кілька класів *нефункціональних вимог*, які є значущими для більшості ПС. Це обмеження, які актуальні для більшості ПрО: вимоги до конфіденційності; вимоги до відмовостійкості; вимоги до кількості клієнтів, що одночасно мають доступ до системи; вимоги до безпеки; вимоги до часу очікування відповіді від системи; вимоги до властивостей системи під час виконання її функцій (обмеження на ресурси пам'яті або процесора, швидкість реакції на звернення до ПС тощо).

Завдання

Відповідно до заданого варіанта ПрО визначити та сформулювати вимоги до ІС та її ПЗ. Для цього потрібно виконати такі дії:

1. Створити логічну модель ІС, яка буде складатися щонайменше з трьох діаграм класів. Описати класи, їх атрибути та функції. З огляду на те, що основними ознаками концепції кожної проектованої інформаційної системи є збирання, оброблення, пошук і подання даних про діяльність організації, виділити із сутностей ПрО значущі об'єкти для виконання наведених функцій і побудувати загальну (базову) діаграму класів. У базовій діаграмі класів побудувати щонайменше два пакети класів, які будуть слугувати основними джерелами інформації про людей (персонал, клієнти, замовники тощо), ресурси, матеріали, готову продукцію, інформаційні та грошові потоки, документи організації.

2. Використовуючи методи визначення вимог, визначити вимоги до ПС, що розробляється. При цьому слід використати повний перелік та опис вимог, який був сформований у лабораторній роботі 1.1.

3. Сформулювати вимоги, відповідно ідентифікуючи їх. Навести номер вимоги та ідентифікатор вимоги (наприклад, 1. REQ001, 2.REQ002, …, 12. REQ012 тощо), а також класифікувати вимогу відповідно до рубрик (підрозділів) *Розділу 5* стандарту IEEE Std 830–93 (задати клас вимоги). Слід особливо звернути увагу на **підрозділи 5.2** та **5.3** стандарту і, відповідно до рекомендацій цих підрозділів, сформулювати повну множину вимог до системи, а також виконати опис кожної конкретної вимоги у текстовому вигляді згідно з вимогами цих підрозділів. У п. 5.2 описуються загальні вимоги до ІС в цілому та складові компоненти цих вимог, а у п. 5.3 задаються детальні описи щодо визначених компонентів (наприклад, інтерфейсів, функцій).

4. Визначити статус та пріоритетність кожної вимоги (див. табл. 1.1).

5. Виконати повний опис кожної вимоги, враховуючи повний перелік та опис вимог, який був сформований у лабораторній роботі 1.1, додавши идентифікатор та класифікацію вимоги відповідно до п. 3 та статус вимоги (див. табл. 1.1, п. 4).

6. Побудувати додаткове подання кожної вимоги формалізованою мовою специфікації вимог. Для подання вимоги необхідно описати потрібну **функцію**, а також **нефункціональні обмеження**, які накладаються на неї. Для цього доцільно скористатися додатковими матеріалами про нефункціональні характеристики та підхарактеристики якості зі стандарту ISO/IEC 9126-1, а також розділами та підрозділами стандарту IEEE Std 830–93.

7. Розробіть ТЗ на створення ІС відповідно до заданого варіанта ПрО. Пункти технічного завдання мають відповідати рубрикам стандарту ГОСТ 34.602–89.

Вимоги до звіту

Для виконання лабораторної роботи слід використати CASE-засіб IBM Rational Rose UML (версії 2007 р., або наступні) та систему створення діаграм Microsoft Visio (2010 р., або наступні — продукт Microsoft Office). Звіт про лабораторну роботу оформляти за допомогою Microsoft Word. Він повинен містити: 1) титульний аркуш з відомостями про виконавця; 2) повний опис варіанта ПрО; 3) діаграми класів (class diagram) логічної моделі ІС (щонайменше три діаграми класів з додатковим описом класів, їх атрибутів та функцій класів природною мовою для кожної діаграми); 4) перелік та аналіз визначених вимог; 5) ідентифікацію та повний опис кожної вимоги до ІС та дані щодо їх статусу; 6) додаткове подання кожної вимоги формалізованою *мовою опису (специфікації) вимог*; 7) опис ТЗ на розроблення ІС.

Запитання та завдання для самоперевірки

1. Для чого використовуються діаграми класів на стадії аналізу вимог до ІС?
2. Назвіть атрибути, які використовують для опису вимог.
3. Що таке шаблони вимог до ПС і для чого їх застосовують?

Література: [1–3]; [6–9].

Лабораторна робота 1.3

МОДЕЛЮВАННЯ АРХІТЕКТУРИ ТА БІЗНЕС-ПРОЦЕСІВ КОМПАНІЇ З ВИКОРИСТАННЯМ ДІАГРАМ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ ТА ДІАГРАМ НОСІЇВ ІНФОРМАЦІЇ

Мета: оволодіння основними принципами моделювання архітектури та бізнес-процесів компанії шляхом створення проекту із застосуванням моделей організаційної структури та діаграми носіїв інформації, що відображають організаційну структуру та інформаційні потоки в межах компанії.

Основні теоретичні відомості

Під методологією створення моделі бізнес-процесу розуміють сукупність способів, за допомогою яких процеси і зв'язки між ними подаються у вигляді моделі [5–6; 10–11]. Для графічного зображення моделі розроблено відповідні нотації. У проекті використовуються основні й допоміжні моделі. Основні моделі застосовують аналітики та власники процесів, а допоміжні моделі — керівники процесів. Студенти в цій роботі використовують дві основні моделі — модель організаційної структури компанії (*Organizational chart — OC*) та модель носіїв інформації (*Information carrier diagram — ICD*).

Діаграми організаційної структури

Діаграма організаційної структури призначена для зведення організаційних елементів компанії в єдину структуру. Як основний принцип побудови використовується принцип ієрархічної супідрядності. На цій діаграмі подаються організаційні одиниці (об'єкти керування) і відношення (зв'язки) між ними. Модель будується ієрархічно, від верхнього рівня організаційної структури до нижнього рівня. До моделі верхнього рівня входять підрозділи організації. Усі вони деталізуються на нижчому рівні — рівні структурних підрозділів. Кінцевим рівнем деталізації є рівень конкретних посад. Усі припустимі об'єкти (табл. 1.2) та зв'язки між об'єктами, що використовуються в діаграмах організаційної структури (ОС), наведено в табл. 2.8 і 2.9 посібника [5].

Діаграма організаційної структури може бути двох типів.

1. *Лінійно-функціональна структура* — відображає ієрархічні взаємозв'язки між організаційними елементами компанії.

2. *Матрична структура* — відображає належність кожного зі співробітників підрозділу до конкретного бізнес-процесу або проекту, що відбувається в цьому підрозділі. Для цього, крім ієрархічних взаємозв'язків, відображуються зв'язки з командами бізнес-процесів.

Таблиця 1.2

Типи об’єктів у діаграмах ОС

Організаційна ланка	Департамент або окремий штатний підрозділ
Група	Група співробітників, що працюють разом протягом певного проміжку часу, наприклад проектна група
Посада	Позначення посади
Бізнес-роль	Опис бізнес-ролей співробітників компанії
ПІБ	Прізвище, ім’я та по батькові штатного співробітника компанії
Позаштатний співробітник EXT	Прізвище, ім’я та по батькові позаштатного співробітника компанії

Правила побудови діаграм організаційної структури

1. Для побудови діаграми ОС використовуються об’єкти, зазначені в табл. 2.8, і зв’язки між ними, зазначені в табл. 2.9 посібника [5].

2. Визначаються організаційні одиниці компанії (відділи, підрозділи, цехи тощо).

3. Визначаються та відображуються зв’язки між організаційними одиницями компанії.

Модель будується ієрархічно, від верхнього рівня структури до її нижнього рівня. До моделі верхнього рівня входять підрозділи та дочірні компанії, що належать до структури холдингу. Кожна з них деталізується на нижчі рівні — рівні структурних підрозділів.

4. Визначаються та відображуються керівники організаційних одиниць (компанії, відділу, підрозділу).

5. Визначаються та відображуються посади, що входять до організаційних одиниць.

6. Визначаються та відображуються співробітники, якими керує безпосередньо начальник.

7. Визначаються та відображуються прізвища, імена та по батькові людей, які обіймають відповідні посади.

8. Визначаються та відображуються бізнес-ролі кожної особи (рис. 1.11).

9. Під час побудови матричної ОС, крім ієрархічних взаємозв’язків, відображуються зв’язки з командами бізнес-процесів. Для цього необхідно визначити групи бізнес-процесів, керівників груп та їх склад (рис. 1.12).

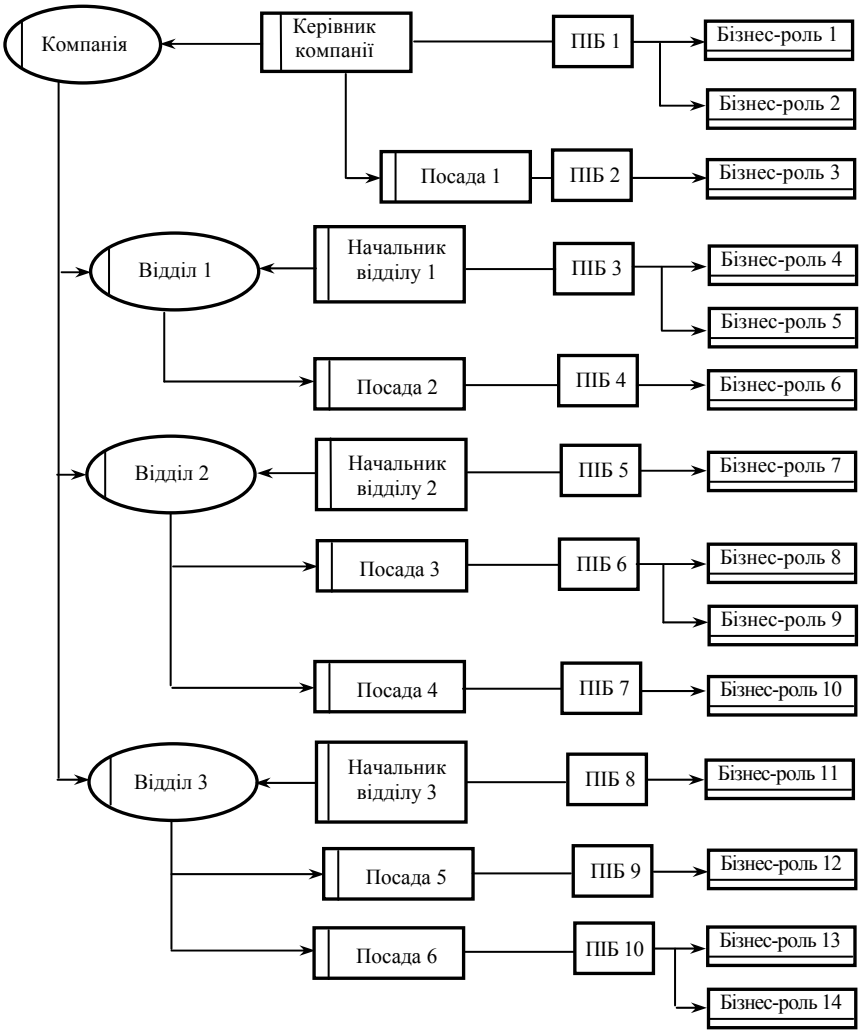


Рис. 1.11. Лінійна організаційна структура компанії

Діаграми носіїв інформації

Діаграма носіїв інформації призначена для відображення структури інформаційних пакетів, розміщених на певних носіях інформації. Інформаційний пакет складається з інформаційних потоків [5; 11–13]. Відображення інформаційних потоків є основним завданням побудови діаграми носіїв інформації.

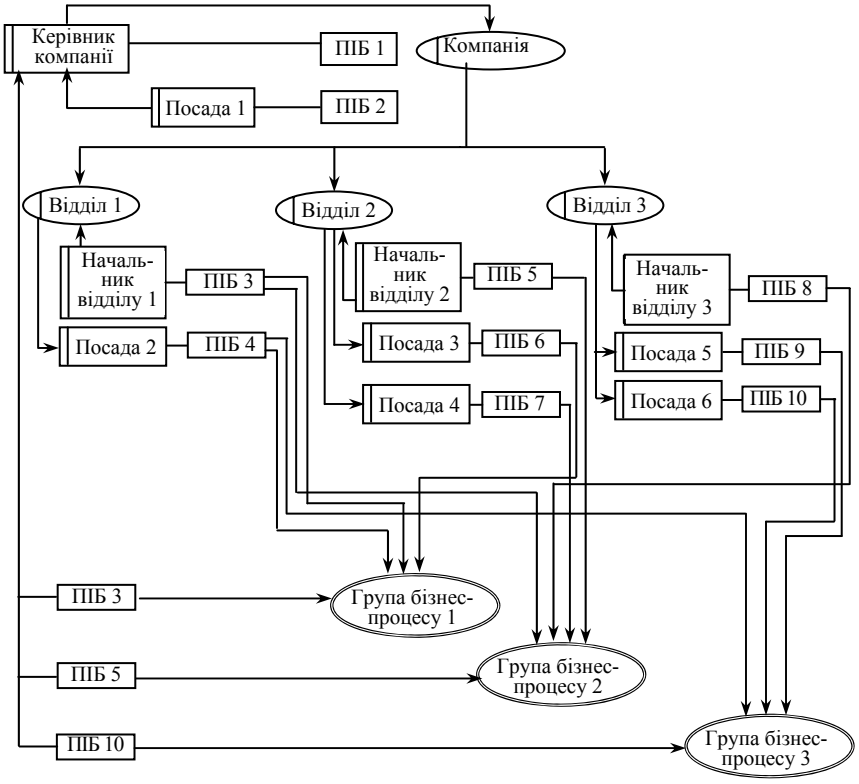


Рис. 1.12. Матрична організаційна структура компанії

Діаграми носіїв інформації (Information Carrier Diagram) використовують для структурованого опису документів компанії.

У діаграмах типу ICD дозволено використовувати об’єкти, типи яких подано в табл. 2.12 посібника [5] і табл. 1.3.

1. Для побудови діаграми носіїв інформації використовуються об’єкти, наведені в табл. 2.12, і зв’язки між ними, наведені в табл. 2.13 посібника [5].

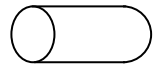
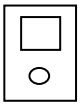
2. Визначаються всі документи, які використовуються в організаційній одиниці (компанії, підрозділі, відділі).

3. Структуруються документи, визначаються групи документів (розбиття за картотеками).

4. У разі потреби виділяються підгрупи документів.

Таблиця 1.3

Допустимі типи об'єктів у діаграмах ICD

 Носій інформації	Позначення носіїв інформації, тип яких не визначено
 Документ	Позначення паперових документів
 Список Р	Позначення списків
 Файл	Позначення документів в електронному вигляді
 Папка	Позначення папки, що містить паперові документи
 Картотека	Позначення картотеки документів

5. У разі потреби відображаються документи у відповідних групах (рис. 2.75 посібника [5]).

6. Деталізація документів до інформаційних потоків здійснюється на окремій ICD діаграмі (рис. 2.76 посібника [5]).

Як носій інформаційного пакета використовують і паперові, і електронні носії. Більшість інформаційних носіїв мають свої реквізити — інформаційні потоки.

Деталізувати кожен інформаційний носій до інформаційних потоків є завданням побудови ICD-діаграм.

На верхньому рівні ICD-діаграми описується структура інформаційних носіїв компанії. Приклад діаграми носіїв інформації (ICD) зображено на рис. 2.77 посібника [5] та на рис. 1.13.

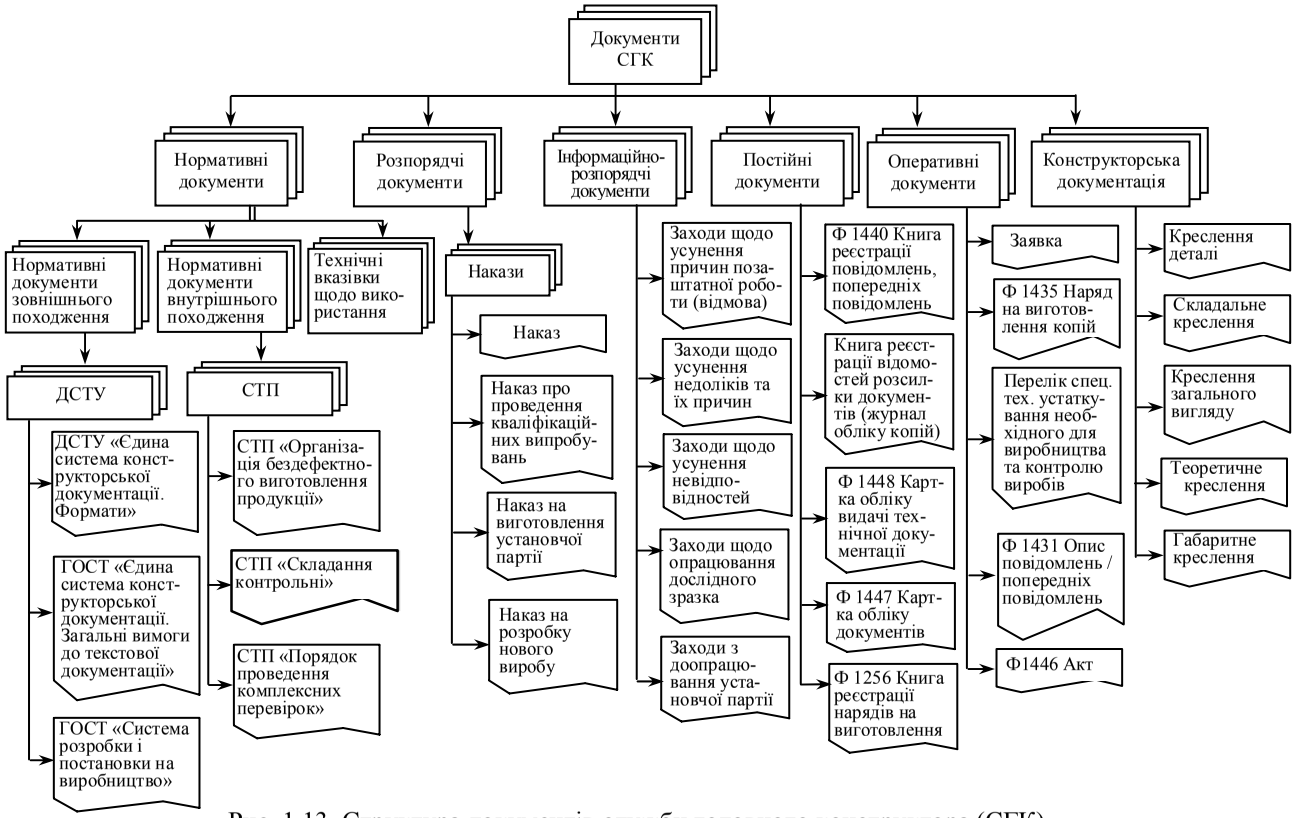


Рис. 1.13. Структура документів служби головного конструктора (СГК)

Завдання

Діаграми моделей створюються за допомогою CASE-засобів **ARIS**. Документація на **ARIS**, указівки з налаштування та інсталяційні комплекти CASE-засобів містяться на сервері кафедри комп'ютерних інформаційних технологій (**\\Balu\student\ Raichev\kurs4**).

1. За допомогою ARIS створити для заданої ПрО діаграми ОС для організації (компанії, установи, фірми, банку, спортивного клубу, будівельної організації тощо), яка вказана у варіанті завдання (описана в межах варіанта ПрО студента).

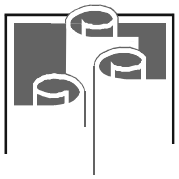
Для цього використати CASE-засіб **ARIS ToolSet** або **ARIS Express** і побудувати діаграму ОС для організації з ПрО студента, що відповідає діаграмі «*Лінійна організаційна структура компанії*» [5]. Побудувати діаграму «*Матрична організаційна структура компанії*», створивши не менше ніж три групи відповідних бізнес-процесів, що функціонують у межах компанії [5]. Бізнес-процеси мають відповідати базовим функціям системи, що розробляється (див. діаграми функцій use case, лабораторна робота 1.1).

2. За допомогою ARIS проаналізувати потоки інформації в межах організації і створити для заданої ПрО діаграми носіїв інформації (ICD): подібні до наведених на рисунках «Групи й підгрупи документів організаційної одиниці» — створити документи для обраних відділів компанії; «Інформаційні потоки документа» — створити потоки для найважливіших документів обраної організаційної ланки компанії; «Структура документів компанії» — створити загальну структуру документів (рис. 1.13).

Запитання та завдання для самоперевірки

1. *Охарактеризуйте поняття системи та підсистеми. Дайте визначення організаційної системи, назвіть її властивості та елементи. Наведіть приклад ОС для типової корпоративної структури.*
2. *Як залежить загальний стан організаційної системи від станів окремих процесів? Наведіть приклад.*

Література: [5–6]; [10–14].



Модуль 2

ГНУЧКІ ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ТЕСТУВАННЯ ТА ДОКУМЕНТУВАННЯ ПРОЦЕСІВ РОЗРОБЛЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Лабораторна робота 2.1

МОДЕЛЮВАННЯ АРХІТЕКТУРИ ТА БІЗНЕС-ПРОЦЕСІВ КОМПАНІЇ З ВИКОРИСТАННЯМ ДІАГРАМ ПРОЦЕСУ ДОДАНОЇ ВАРТОСТІ ТА ДІАГРАМ ПОДІЄВО-КЕРОВАНОГО ПРОЦЕСУ

Мета: оволодіння основними принципами моделювання архітектури та бізнес-процесів компанії шляхом створення моделі процесу доданої вартості з побудовою діаграм **V**alue-**a**dded chain **D**iagram (VAD), які відображають базові процеси, що створюють вартість, а також створення моделі подієво-керованого процесу з побудовою діаграм розширеного ланцюжка подієво-керованих процесів (***e**xtended **E**vent-driven **P**rocess **C**hain — **eEPC***), які відображають базові процеси шляхом опису їх алгоритмів.

Основні теоретичні відомості

Під методологією створення моделі бізнес-процесу розуміють сукупність способів, за допомогою яких процеси і зв'язки між ними подаються у вигляді моделі. У проекті використовуються основні та допоміжні моделі.

Основні моделі застосовують аналітики і власники процесів, а допоміжні — керівники процесів. Студенти в цій роботі використовують одну з основних моделей і будують діаграми процесу доданої вартості VAD.

Діаграми процесу доданої вартості (VAD)

Діаграми процесу доданої вартості застосовуються для опису процесів верхнього рівня компанії через визначення логічного взаємозв'язку між основними напрямами діяльності компанії та відображення цих взаємозв'язків у вигляді структурованих груп бізнес-процесів, що створюють продукт чи послугу.

Діаграми процесу доданої вартості VAD використовуються для концептуального моделювання бізнес-процесів компанії.

Допустимі об'єкти та зв'язки на діаграмі VAD

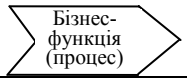
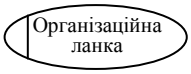
Для побудови VAD-діаграми необхідно виокремити основні бізнес-процеси, за допомогою яких провадиться діяльність компанії. На наступному етапі визначається логічна послідовність цих процесів (вони можуть бути паралельними або послідовними).

На VAD-діаграмі всі бізнес-процеси мають вхідну інформацію (необхідну для виконання процесу), вихідну інформацію (результат виконання процесу), виконавця (відділ, підрозділ, посада) і відповідального за процес. Кожний бізнес-процес на VAD-діаграмі повинен мати посилання або на VAD-діаграму більш низького рівня, або на eEPC-діаграму, що деталізує даний бізнес-процес. Допустимі типи об'єктів на діаграмі доданої вартості наведено в табл. 2.1. Усі допустимі зв'язки між об'єктами, що використовуються в діаграмах типу VAD, наведено в табл. 2.11 посібника [5].

Модель будується ієрархічно, починаючи від верхнього рівня процесів компанії до нижнього.

Таблиця 2.1

Типи об'єктів у діаграмах VAD

	Бізнес-процес верхнього рівня, виконання якого спрямоване на досягнення мети
	Департамент або окремий штатний підрозділ, що виконує бізнес-процес
	Власник бізнес-процесу
	Товар чи послуга, що забезпечує досягнення мети або є результатом виконання бізнес-функції
	Інформаційні потоки, що забезпечують вхідні та вихідні дані процесу
	Якісна або кількісна ситуація (стан), досягнення якої важливе для компанії
	Нормативні документи

Правила побудови діаграм процесу доданої вартості

1. Для побудови діаграм VAD використовуються об'єкти, наведені в табл. 2.1, і зв'язки між ними, наведені в табл. 2.11 посібника [5].

2. Визначаються та відображуються бізнес-процеси організаційної одиниці (компанії, відділу, підрозділу), за допомогою яких провадиться її діяльність.

3. Визначаються та відображуються бізнес-процеси у відповідній логічній послідовності з відображенням власника і команди бізнес-процесу, що виконує бізнес-процес (рис. 2.1).

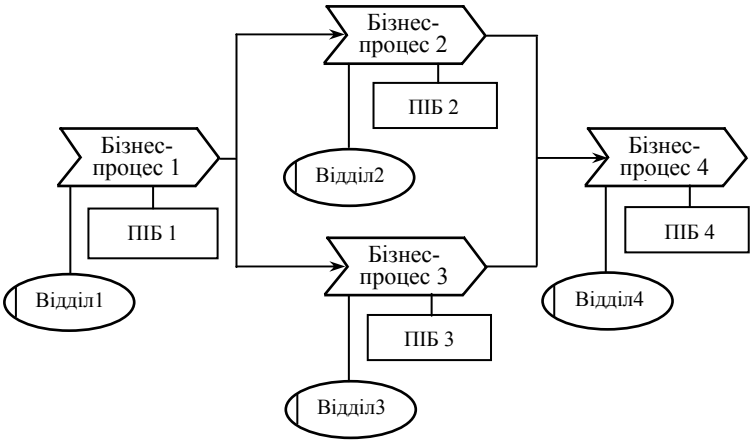


Рис. 2.1. Бізнес-процеси організаційної одиниці із зазначенням бізнес-команди та власників бізнес-процесів

4. Визначаються та відображуються нормативні документи, згідно з якими виконується бізнес-процес.

5. Визначаються та відображуються інформація і ресурси, необхідні для виконання бізнес-процесу, та інформація, ресурси та продукти, які буде отримано в результаті його виконання.

VAD-діаграму бізнес-процесів організаційної одиниці із зазначенням бізнес-команди, власників бізнес-процесів, регламентної документації, а також вхідної і вихідної документації та інформаційних потоків даних і робочих та вихідних продуктів, показано на рис. 2.2.

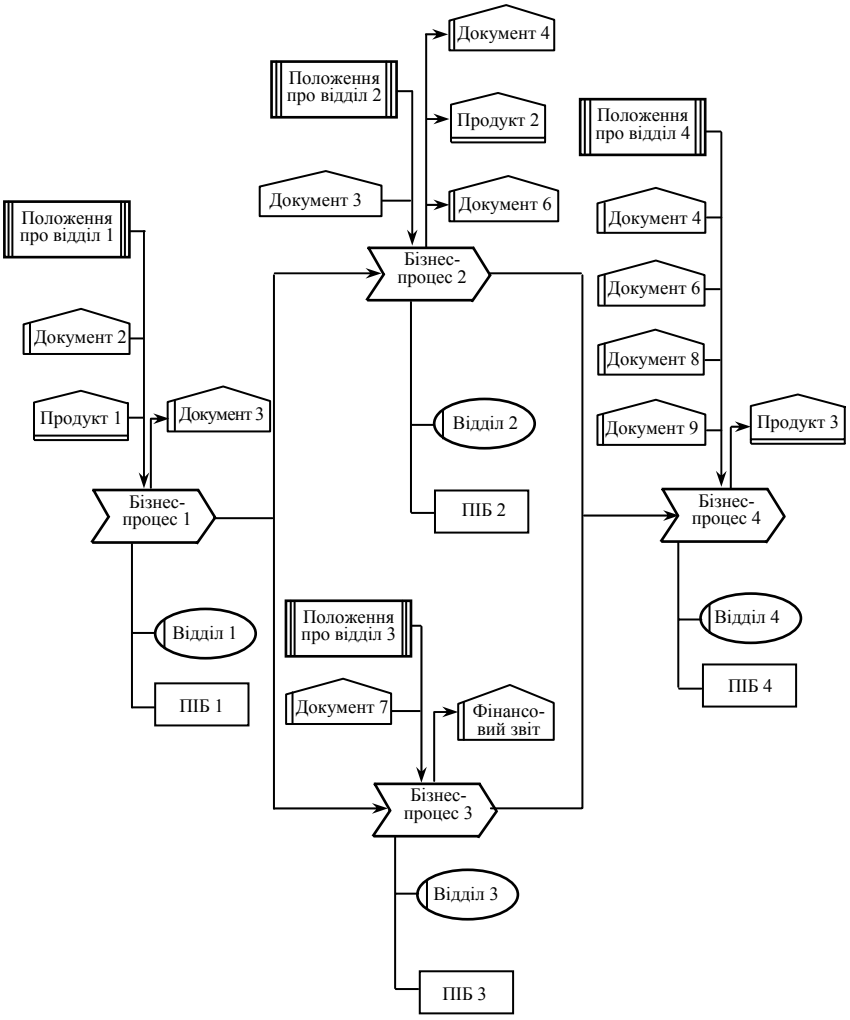


Рис. 2.2. VAD-діаграма бізнес-процесів організаційної структури

Діаграми подієво-керованого процесу (extended Event-driven Process Chain — eEPC)

Під методологією створення моделі бізнес-процесу розуміють сукупність способів, за допомогою яких процеси і зв'язки між ними подаються у вигляді моделі [5; 10]. Для графічного зображення моделі розроблено відповідні нотації.

Студенти в цій роботі будують ще одну з основних моделей — діаграму подієво-керованого процесу (***e****xtended* ***E****vent-driven* ***P****rocess* ***C****hain — **eEPC***).

Ця діаграма призначена для опису алгоритму виконання окремого сценарію або бізнес-процесу у вигляді послідовності процедур, якими керують події. У моделі головна увага приділяється послідовності виконання процедур (дій) як складових цього сценарію. Модель становить собою послідовний набір подій і функцій, що відображають логічне виконання взаємозалежних дій, спрямованих на досягнення заданого результату (виробництво продукції, провадження торгової діяльності, укладання контракту тощо).

Діаграми подієво-керованого процесу (eEPC) призначені для детального моделювання бізнес-процесів діяльності компанії.

У діаграмах типу eEPC дозволено використовувати об'єкти, типи яких наведено в табл. 2.14 посібника [5].

Опишемо всі допустимі взаємозв'язки між об'єктами, що застосовуються в діаграмах типу eEPC за цією угодою. Подія активізує функцію, подія може розгалужуватися і є результатом інтерфейсу або межі процесу. Функція, своєю чергою, створює подію, вихідні документи, бази даних, знання, навички, регламенти, товари та послуги, картотеку, інформаційні потоки та підтримує мету і може бути результатом нормативних документів. Функція має ризики і може надавати інформацію в контрольних точках. Розгалуження веде до події і активізує функцію. Посада й організаційна ланка виконують певні функції. Документи, бази даних, картотеки, інформаційні потоки і товари та послуги забезпечують вхід для функцій. Знання та навички необхідні для виконання функцій. Межа та інтерфейс процесу є входом для події [5; 10].

Правила побудови діаграми подієво-керованого процесу (eEPC)

1. Для побудови діаграми подієво-керованого процесу застосувати об'єкти, подані в табл. 2.14 посібника [5], і взаємозв'язки між ними.

2. Визначити послідовність дій і подій, потрібних для виконання процесу. Кожна eEPC-модель починається щонайменше однією стартовою ініціюючою подією (станом) і завершується щонайменше однією результуючою подією (станом). Події та функції під час виконання процесу мають чергуватися [5; 10].

3. Усі функції подавати в правильній послідовності. Врахувати можливість як послідовного, так і паралельного виконання функцій.

4. Події та функції повинні мати тільки по одному вхідному і одному вихідному відношенню, що показує хід виконання бізнес-процесу. Якщо є розгалуження, то необхідно використовувати оператори розгалуження («AND», «OR», «XOR»), при цьому показувати всі можливі варіанти перебігу процесу та результати виконання функцій. Розгалуження завжди починається після функції.

5. Визначити попередні і наступні процеси з погляду процесу, що деталізується, відобразити їх у вигляді інтерфейсів та створити з них посилання на відповідні бізнес-процеси.

6. Якщо немає попередніх або наступних процесів у межах компанії, то використовується об'єкт «Межа процесу» («Початок процесу» або «Завершення процесу»).

7. Визначити та відобразити всю необхідну інформацію та ресурси, потрібні для виконання функції, а також результати виконання функції. Максимально точно відображати вхідну і вихідну інформацію. Для документів, таких як наказ, службова записка, заява та інші, вказувати їх призначення.

8. Якщо в результаті виконання функції не створюється новий документ, а тільки виконуються деякі дії із вхідним документом, то на виході вказати інформацію про виконані дії, при цьому назву документа не змінювати.

9. Будь-який рух документів фіксувати в інформаційному об'єкті («документ передано», «документ отримано»). Якщо відповідно до стандарту чи положення передбачено реєстрацію руху документів, то на eEPC-діаграмі обов'язково відображається окрема функція «Реєстрація документа».

10. Визначити та відобразити нормативний документ, який регламентує виконання кожної функції, а також виконавця кожної функції.

11. Відобразити прикладні системи, що використовуються під час виконання функції і вказати час її виконання (у разі потреби).

12. Визначити можливість і необхідність декомпозиції функції. Якщо потрібно, можна розписати будь-яку функцію більш детально на наступній eEPC-діаграмі та зробити посилання (або ★) на неї. Аналогічним чином можна виконати декомпозицію базових процесів, у тому числі інтерфейсних.

Діаграму подієво-керованого процесу, на якій зображено функціонально-подієву послідовність бізнес-процесу з інтерфейсами, вхідною та вихідною інформацією, регламентними документами, виконавцями, прикладними системами, продуктами та часом виконання функцій, показано на рис. 2.3.

Детальний приклад розроблення діаграми eEPC від бізнес-процесу збирання інформації до бізнес-процесу формування звітності з використанням операторів розгалуження подано на рис. 2.97 посібника [5].

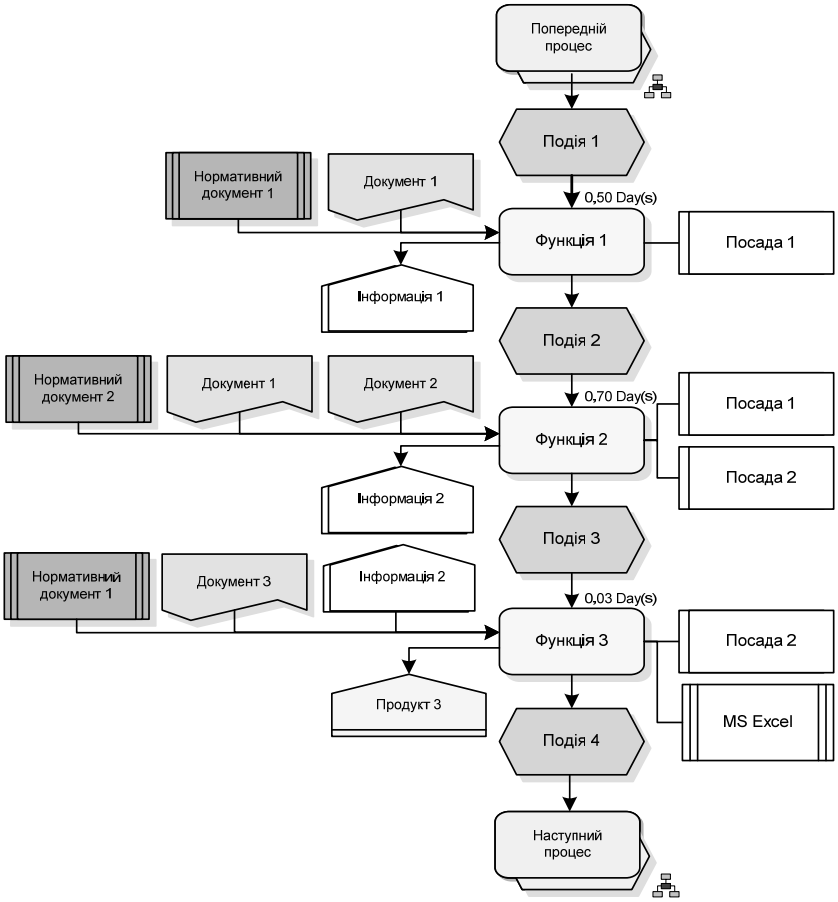


Рис. 2.3. Загальна форма eEPC-діаграми без застосування операторів розгалуження (керований подіями ланцюжок функцій бізнес-процесу)

Завдання

Діаграми моделі створюються за допомогою CASE-засобів **ARIS**. За допомогою ARIS створити для заданої ПрО діаграми процесів доданої вартості (VAD) для організації (компанії, установи, фірми, банку, готелю, спортивного клубу, будівельної організації тощо), яка подана у варіанті завдання (описана в межах варіанта ПрО).

Для цього використати CASE-засіб **ARIS ToolSet** та **ARIS Express** і побудувати діаграму VAD для ***трьох головних (базових) процесів організації***, якими керують групи бізнес-процесів, що беруть участь у діаграмі *матричної організаційної структури компанії* (лабораторна робота 1.3). Для цих базових процесів побудувати діаграму, що відповідає рис. 2.2, а також рис. 2.71 посібника [5]. Проаналізувати подальшу декомпозицію і визначити ті бізнес-процеси, які потрібно декомпозувати, створивши діаграми **VAD**, та інші процеси, які потрібно декомпозувати, створивши діаграми **eEPC**, для відповідних бізнес-процесів компанії, тобто виконати подальшу декомпозицію щонайменше трьох базових процесів діаграми процесів доданої вартості.

Головні (базові) бізнес-процеси компанії мають відповідати базовим функціям розроблюваної системи. За допомогою побудованої ARIS-моделі у звіті письмово проаналізувати потоки інформації, документів, ресурсів та продуктів у межах організації, що зображені на VAD-діаграмах функціонування компанії у середовищі ПрО.

Діаграми моделі **eEPC** створюються також за допомогою CASE-засобів **ARIS**. За допомогою ARIS створити для заданої ПрО діаграми подієво-керованого процесу (eEPC) для організації, указаної у варіанті його завдання.

Для цього використати CASE-засіб **ARIS ToolSet** та **ARIS Express** і побудувати діаграми подієво-керованого процесу *тих базових бізнес-процесів організації*, що визначені в лабораторній роботі для декомпозиції за технікою eEPC. Для кожного з цих бізнес-процесів, що функціонують у межах компанії, побудувати діаграми (*не менше двох*), подібні до зображених на рис. 2.3 або рис. 2.96 та рис. 2.97 посібника [5].

Бізнес-процеси компанії будувати із застосуванням ланцюжка подій і функцій, які мають розкривати відповідні бізнес-процеси з

VAD-діаграм. За допомогою побудованих діаграм у звіті письмово проаналізувати події, функції, інтерфейси, організаційні ланки, потоки інформації, документи, ресурси, товари та продукти в межах організації, використані на побудованих eEPC-діаграмах функціонування компанії у середовищі ПрО.

Зауваження. Для інсталяції **ARIS Express** необхідно зареєструватися на сайті www.ariscommunity.com, указавши e-mail, куди буде надіслано логін та пароль.

Запитання та завдання для самоперевірки

1. *Розгляньте процесний підхід до моделювання систем. Наведіть визначення бізнес-процесу. Як поєднуються в єдину модель керування бізнес-процеси та організаційна структура компанії?*
2. *Дайте визначення процесного підходу в організації (процесна модель). Для чого поєднують бізнес-процеси та інформаційні потоки даних у процесному підході? Назвіть недоліки функціонально-орієнтованого підходу і переваги процесного підходу.*

Література: [5–6]; [10–14].

Лабораторна робота 2.2

МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ WEB-ДОДАТКІВ

Мета: оволодіння засадами та методами тестування web-додатків. Навчитися застосовувати правила написання тест-кейсів і баг-репортів з метою їх використання в процесі тестування web-сайтів для складання тестових випадків та тестових наборів даних, а також для опису виникаючих помилкових ситуацій.

Основні теоретичні відомості

Загальні відомості та основні поняття

Верифікація (Verification) — це процес оцінювання системи і/або її компонентів з метою визначення, чи задовольняють результати поточного етапу розроблення умови, сформовані на початку цього етапу, тобто чи виконуються терміни і завдання розроблення проекту, визначені на початку поточної фази. Крім того здійснюється повна перевірка працездатності компонентів ПС на даному етапі розроблення.

Валідація (Validation) — це визначення відповідності ПС, що розробляється, очікуванням і потребам користувача, тобто визначення відповідності ПС вимогам до неї.

Тестування програмного забезпечення (Software Testing) — це перевірка відповідності між реальною і очікуваною поведінкою програми, яка здійснюється на скінченному наборі тестів, що вибираються певним чином [IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004]. У більш широкому розумінні, **тестування** — це одна з технік ***контролю якості***, що передбачає активність планування робіт з тестування (**Test Management**), проектування тестів (**Test Design**), виконання тестування (**Test Execution**) та аналіз отриманих результатів (**Test Analysis**).

Особливості тестування **web-додатків** пов'язані з трьома основними факторами: моделлю ЖЦ ПЗ, архітектурою web-додатка та тестуванням технічних характеристик якості (*надійність, безпека, зручність використання, продуктивність, ефективність, задоволеність користувача* та ін.).

План тестування (Test Plan) — це документ, що описує обсяг робіт з тестування, починаючи з описів об'єктів, стратегії, розкладу, критеріїв початку і закінчення тестування, необхідного в процесі роботи обладнання, спеціальних знань, а також оцінювання ризиків з варіантами їх розв'язання.

Тест-дизайн (Test Design) — це етап процесу тестування ПЗ, на якому проектуються та створюються тестові випадки (тест-кейси), згідно з визначеними раніше критеріями якості та метою тестування.

Тестовий випадок (Test Case) — це артефакт, який описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується відповідно до заданого сценарію.

Баг/Дефект Репорт (Bug Report) — документ, який описує ситуацію або послідовність дій, що призводить до некоректної роботи об'єкта тестування, з указанням причин і очікуваного результату.

Тестове покриття (Test Coverage) — це одна з метрик оцінювання якості тестування, що являє собою щільність покриття тестами вимог, чи коду, який виконується. **Деталізація тест-кейсів** (**Test Case Specification**) — це рівень деталізації опису тестових кроків і необхідного результату, за якого забезпечується розумне

співвідношення часу проходження до тестового покриття. **Час проходження тест-кейсу** (**Test Case Pass Time**) — це час від початку проходження кроків тестового випадку до отримання результату тесту.

Тестовий випадок (Test Case)

Тестовий випадок (**Test Case**) описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини. Під тест-кейсом розуміють структуру вигляду: **Action** > **Expected Result** > **Test Result**. Приклад тест-кейсу наведено в табл. 2.2.

Таблиця 2.2

Приклад тест-кейсу

Action	Expected Result	Test Result (passed/failed/blocked)
Open page Login	Login page is opened	Passed

Види тестових випадків. Тест-кейси поділяють за очікуваним результатом на **позитивні** або **негативні. Позитивний тест-кейс** використовує тільки коректні дані та перевіряє, що додаток у цьому випадку правильно виконує функцію, що викликається на множині цих даних. **Негативний тест-кейс** оперує як коректними, так і некоректними даними (містить щонайменше один некоректний параметр) і перевіряє виняткові граничні ситуації (спрацьовування валідаторів), а також не виконання викликуваної додатком функції в разі спрацьовування валідатора.

Структура тестових випадків (Test Case Structure). В Інтернеті можна знайти багато інформації про структуру тест-кейсів, рівні їх деталізації та кількість перевірок у них. Однак більшість тестувальників для тестування web-додатків використовують підхід, наведений у табл. 2.3, відповідно до якого будь-який тест-кейс має три частини.

Таблиця 2.3

Частини тест-кейсу

PreConditions	Список дій, які переводять систему у стан, придатний для проведення основної перевірки, або виконується список умов, здійснення яких означає, що система перебуває в придатному стані для проведення основного тесту

Закінчення табл. 2.3

Test Case Description	Список дій, які переводять систему з одного стану в інший, для отримання результату, на підставі якого можна зробити висновок про задоволення реалізованої функції поставленим вимогам
PostConditions	Список дій, що переводять систему в первинний стан (стан до проведення тесту — Initial state)

Примітка. **Post Conditions** не є обов'язковою частиною тест-кейсу. Це передусім правило гарного тону: «насмітив — прибери за собою», що особливо актуально в разі автоматизованого тестування, коли за один прогін можна заповнити базу даних сотнями некоректних документів про результати процесу тестування.

Приклад тест-кейсу:

do A1, verify B1
do A2, verify B2
do A3, verify B3

У прикладі остання перевірка — B3, а це означає, що саме вона є ключовою. Отже, A1 та A2 — це дії, що переводять систему в тестопридатний стан. Своєю чергою, B1 і B2 — умови того, що система перебуває у стані, придатному для тестування. Тоді маємо тест-кейс, наведений у табл. 2.4.

Таблиця 2.4

Тест-кейс із передумовами

Action	Expected Result	Test Result (passed/failed/blocked)
PreConditions		
do A1	Verify B1	
do A2	Verify B2	
Test Case Description		
do A3	Verify B3	
PostConditions		

PostConditions у прикладі не описані, але логічно треба виконати кроки, які б повернули систему в первісний стан (наприклад, видалили б створений запис, або відмінили б зміни, зроблені в документі). Тепер відповімо на *питання*: «Чому таке розбиття зручно використовувати?» *Відповідь:* кінцева перевірка одна, тому

у випадку, якщо тест «провалений» (**test failed**), відразу стане зрозуміло через що. Дійсно, якщо «провалениими» виявляться перевірки B1 і/або B2, тест-кейс взагалі буде заблоковано (**test blocked**), оскільки функції неможливо надати тестопридатного стану. Однак це не означає, що функція не працює або містить помилки. Якщо перевірка B2 «провалена», то виконується опис результатів проходження перевірок умов (табл. 2.5).

Таблиця 2.5

Заблокований тест-кейс

Action	**Expected Result**	**Test Result (passed/failed/blocked)**
PreConditions		
do A1	verify B1	Passed
do A2	verify B2	**Failed**
Test Case Description:		
do A3	verify B3	**Blocked**
PostConditions		

Деталізація опису тест-кейсів (Test Case Specification)

Є багато різних думок про рівень деталізації тест-кейсів, а також про кількість перевірок в одному тест-кейсі. Всі вони по-своєму правильні. У будь-якому випадку рівень деталізації тест-кейсів має бути таким, аби забезпечити розумне співвідношення між часом проходження і тестовим покриттям (доки покриття тестами визначених функцій не змінюється, можна зменшувати деталізацію тест-кейсу, табл. 2.6).

Таблиця 2.6

Приклад тест-кейсу 1

Перевірка відображення сторінки		
Дія	**Очікуваний результат**	**Результат тесту**
Відкрити сторінку «Вхід до системи»	Вікно «Вхід до системи» відкрито. Назва вікна «Вхід до системи». Логотип компанії відображається у правому верхньому куті. На формі 2 поля — Ім’я та Пароль. Кнопка «Вхід» доступна. Посилання «Забув пароль» — доступне	

Приклад тест-кейсу 2.

Назва: Перевірка відображення сторінки.

Дія: Відкрити сторінку «Вхід до системи».

Перевірка: Перевірити, що відображена сторінка відповідає сторінці на рисунку (слід додати правильне зображення сторінки «Вхід до системи» на відповідний рисунок).

У прикладах тест-кейсів 1 і 2 покриття буде однаковим, але час, потрібний для проходження – різний. У цьому випадку тест-кейс 2 є більш наочним. Зазначимо, що рішення про вигляд тест-кейсу та деталізацію його опису приймає особа, яка відповідає за його створення — тест-дизайнер або тест-аналітик. Вона має необхідний досвід застосування на практиці технік тест-дизайну. У деяких компаніях ця роль не виділяється окремо, а покладається на рядових тестерів, тому у випадку їх недостатньої кваліфікації доводиться багато разів переписувати різні тест-кейси.

Для того аби команда (група) тестування працювала злагоджено, у всіх її членів повинен бути ***єдиний шаблон*** або підхід до написання ***тест-кейсів***. Як такий ***єдиний шаблон*** найчастіше використовують ***триступеневу структуру*****: PreConditions**, **Test Case Description** і **PostConditions**.

Баг-Репорт (Bug Report)

Баг або **дефект-репорт** — документ (один з артефактів тестування), що описує ситуацію некоректної роботи об’єкта тестування. Для отримання більш детальної інформації про баг-репорт рекомендується ознайомитись із наступною інформацією про структуру, особливості опису та інших нюансах, потрібних для написання якісних баг-репортів.

Розглянемо три головні аспекти складання баг-репорта: 1) визначення структури баг-репорта; 2) важливість і пріоритет дефекту; 3) написання власне баг-репорта.

Пропонуємо студентам коментар розробника: «*Прочитавши короткий опис бага (**Bug Summary**), я повинен зрозуміти у чому полягає проблема, а прочитавши детальний опис бага (**Bug Description**) я маю точно знати рядок коду, який треба виправити*».

Зміст та значущість висловлення розробника зводяться до того, що всі дії необхідно виконати таким чином, щоб у нього до тестувальника було менше питань по суті описаної в баг-репорті проблеми. Річ у тім, що кожний повернений баг-репорт зі статусом

«Відхилений», «Не відтворюється», «Потрібна інформація» (Rejected, Can't Reproduce, More info) — це втрата часу як тестера, так і розробника. Якщо тестувальники користуватимуться запропонованими нижче рекомендаціями, то якість їх баг-репортів матиме високий рівень, а отже, у процесі роботи виникатиме менше претензій як від менеджерів, так і від розробників.

Структура баг-репорта. Основні поля баг/дефект-репорта

Різні системи менеджменту дефектів пропонують різні поля для заповнення та різні структури опису дефектів.

Сформований на основі отриманого досвіду та рекомендований до використання стандартизований **шаблон баг-репорта**, наведено в табл. 2.7.

Таблиця 2.7

Шаблон баг-репорта

Назва	
Короткий опис (**Summary**)	Короткий опис проблеми, що явно вказує на причину і тип помилкової ситуації
Проект (**Project**)	Назва тестованого проекту
Компонент додатка (**Component**)	Назва частини або функції продукту
Номер версії (**Version**)	Версія, у якій знайдено помилку
Важливість (**Severity**)	Найбільш поширена пятирівнева система градації важливості (серйозності) дефекту: S1 Блокувальний (Blocker) S2 Критичний (Critical) S3 Значний (Major) S4 Незначний (Minor) S5 Тривіальний (Trivial)
Пріоритет (**Priority**)	Пріоритет дефекту: P1 Високий (*High*) P2 Середній (*Medium*) P3 Низький (*Low*)
Статус (**Status**)	Статус бага. Залежить від використовуваної процедури та життєвого циклу бага (**bug workflow and life cycle**)
Автор (**Author**)	Автор баг-репорта
Призначений на (**Assigned To**)	Ім’я співробітника, призначеного для вирішення проблеми

Закінчення табл. 2.7

Оточення	
ОС Сервіс-пак/ім'я браузера + версія	Інформація про оточення, у якому знайдено баг: операційна система; сервіс-пак (SP– Service Pack); для web-тестування – ім'я та версія браузера тощо
Опис	
Кроки відтворення (**Steps to Reproduce**)	Кроки, по яких можна легко відтворити ситуацію, що призвела до помилки
Фактичний результат (**Result**)	Результат, отриманий після проходження кроків відтворення
Очікуваний результат (**Expected Result**)	Очікуваний правильний результат
Доповнення	
Прикріплений файл (**Attachment**)	Файл з логами, скріншот або будь-який інший документ, який може допомогти прояснити причину помилки, чи вказати на спосіб вирішення проблеми

Важливість і пріоритет дефекту

Різні системи баг-трекінгу (**bug tracking**) пропонують різні способи опису важливості та пріоритету баг-репорта. Виходячи із сукупного досвіду, слід розрізняти ці поняття, точніше — застосовувати обидва поля **Severity** й **Priority**, оскільки зміст цих полів істотно відрізняється.

Важливість (**Severity**) — атрибут, що характеризує вплив дефекту на працездатність додатка, тоді як **Пріоритет** (**Priority**) — це атрибут, що вказує на черговість виконання задачі або дій з усунення дефекту. Передусім це інструмент менеджера з планування робіт, причому, чим вищий пріоритет, тим швидше необхідно виправити дефект.

Градація важливості дефекту (Severity)

S1 Блокувальний (Blocker). *Помилка, що блокує* та призводить додаток у непрацюючий стан (відмова), у результаті чого подальша робота із системою або її ключовими функціями, що тестуються, стає неможливою. Вирішення проблеми необхідне для подальшого функціонування системи.

S2 Критичний (Critical). *Критична помилка* — неправильно працююча ключова бізнес-логіка; «дірка» у системі безпеки; проблема, що призвела до тимчасового «падіння» сервера, або помилка, що призвела в непрацюючий стан деяку частину системи (зупинка),

без можливості вирішення проблеми, використовуючи інші вхідні точки. Вирішення проблеми необхідне для подальшої роботи з ключовими функціями тестованої системи.

S3 Значний (Major). *Значна помилка* — коли частина основної бізнес-логіки працює некоректно. Помилка не критична, або є можливість для роботи з тестованою функцією, використовуючи інші вхідні точки.

S4 Незначний (Minor). *Незначна помилка*, що не порушує бізнес-логіку тестованої частини додатка, очевидна проблема користувацького інтерфейсу.

S5 Тривіальний (Trivial). *Тривіальна помилка* — помилка, яка не стосується бізнес-логіки додатка; погано відтворювана проблема, малопомітна засобами користувацького інтерфейсу; проблема сторонніх бібліотек чи сервісів; проблема, що не впливає на загальну якість програмного продукту.

Градація пріоритету дефекту (Priority)

P1 Високий (High). Помилка має бути виправлена якнайшвидше, бо її наявність є критичною для проекту.

P2 Середній (Medium). Помилка повинна бути виправлена, її наявність не є вкрай критичною, але потребує обов'язкового усунення.

P3 Низький (Low). Помилка повинна бути виправлена, однак її наявність не є критичною, та не потребує термінового усунення.

Порядок виправлення помилок відповідно до їх пріоритетів: **High -> -> Medium -> Low.**

Вимоги до кількості відкритих багів

Пропонується такий підхід для визначення вимог до кількості відкритих багів: наявність відкритих дефектів P1, P2 і S1, S2 вважається неприйнятною для проекту. Подібні ситуації потребують негайного вирішення і контролюються менеджерами проекту. Наявність суворо обмеженої кількості відкритих помилок P3 і S3, S4, S5 не є критичною для проекту і дозволяється у видаваному додатку. Загальна кількість відкритих помилок залежить від розміру проекту та встановлених критеріїв якості. Вимоги до відкритих помилок обумовлюються та документуються на етапі прийняття рішення про рівень якості розроблюваного ПП. Приклад документування подібних вимог вказується в пункті «Критерії завершення тестування» у плані тестування.

Написання баг-репорта. Вимоги до обов'язкових полів

Обов'язкові поля баг-репорта: **короткий опис** (*Bug Summary*), **важливість** (*Severity*), **кроки (до) відтворення** (*Steps to reproduce*), **результат** (*Actual Result*), **очікуваний результат** (*Expected Result*). Нижче наведено вимоги та приклади заповнення цих полів.

Короткий опис. В одному реченні необхідно умістити зміст усього баг-репорта, а саме: стисло і зрозуміло, використовуючи правильну термінологію, вказати, що і де не працює. Наприклад: *«Додаток зависає під час спроби збереження текстового файлу розміром понад 50 мб»*, або *«Дані на формі «Профайл» не зберігаються після натиснення кнопки «Зберегти».*

Важливість дефекту. Якщо виявлена проблема полягає у ключовій функціональності додатка і після її виникнення додаток стає повністю недоступним, а подальша робота з ним неможливою, то ця проблема (помилка) є *блокувальною*. Зазвичай усі блокувальні проблеми виявляються під час первинної перевірки нової версії продукту (*Build Verification Test, Smoke Test*), бо їх наявність не дозволяє повноцінно виконувати тестування. Якщо ж тестування можна продовжити, то градація важливості цього дефекту буде **критичною.** В разі *значних*, *незначних* та *тривіальних* помилок, це питання не потребує пояснень.

Кроки до відтворення (Steps to reproduce) / Результат (Actual Result) / Очікуваний результат (Expected Result). Дуже важливо чітко описати всі кроки з указанням усіх введених даних (імені користувача, даних заповнення форми і т. ін.) та проміжних результатів. Наприклад:

Кроки до відтворення

1. Увійдіть у систему: Користувач **Тестер1**, пароль xxxXXX **-->** Вхід у систему здійснений.
2. Натисніть лінк **Профайл -->** Сторінка **Профайл** відкрилася.
3. Уведіть нове ім'я користувача: **Тестер2**.
4. Нажміть кнопку **Зберегти**.

Результат

На екрані з'явилася помилка. Нове ім'я користувача не було збережено.

Очікуваний результат

Сторінка **Профайл** перезавантажилась. Нове значення імені користувача збережено.

Основні помилки під час написання баг-репортів

Недостатність наданих даних. Не завжди одна й та сама проблема виникає для усіх значень, що вводяться, та для будь-якого користувача, що ввійшов у систему, а тому настійно пропонується вносити всі необхідні дані у баг-репорт.

Визначення важливості. Дуже часто відбувається або завищення, або заниження *важливості* дефекту, що приводить до неправильної черговості під час вирішення проблеми.

Мова опису. Часто для опису проблеми використовують неправильну термінологію або складні мовні обороти, які можуть ввести в оману особу, відповідальну за вирішення проблеми.

Відсутність очікуваного результату. У випадках, якщо не вказано, яка буде потрібна поведінка системи, тестувальник витрачає час розробника на пошук цієї інформації, тим самим гальмуючи виправлення дефекту. Необхідно вказувати: пункт у вимогах, написаний тест-кейс, або ж власну думку, у разі якщо ця ситуація не була задокументована.

Заповнення полів баг-репорту

У табл. 2.8 подано основні поля баг-репорта та роль працівника, відповідального за заповнення даного поля. Обов'язкові для заповнення поля позначені курсивом.

Таблиця 2.8

Роль працівника, відповідального за заповнення полів баг-репорта

Поле	Відповідальний за заповнення поля
Короткий опис (Summary)	Автор баг-репорта (зазвичай це тестувальник)
Проект (Project)	Автор баг-репорта (зазвичай це тестувальник)
Компонент додатка (Component)	Автор баг-репорта (зазвичай це тестувальник)
Номер версії (Version)	Автор баг-репорта (зазвичай це тестувальник)
Важливість (Severity)	Автор баг-репорта (зазвичай це тестувальник), однак цей атрибут може бути змінений відповідальним менеджером
Пріоритет (Priority)	Менеджер проекту або менеджер, відповідальний за розроблення компонента, для якого написано баг-репорт
Статус (Status)	Автор баг-репорта (зазвичай це тестувальник), але багато систем баг-трекінгу виставляють статус за замовчуванням

Закінчення табл. 2.8

Поле	Відповідальний за заповнення поля
Важливість (Severity)	Автор баг-репорта (зазвичай це тестувальник), однак цей атрибут може бути змінений відповідальним менеджером
Автор (Author)	Установлюється за замовчуванням; якщо ні, то вказується ім'я автора баг-репорта
Назначений (на) (Assigned to)	Менеджер проекту або менеджер, відповідальний за розроблення компонента, для якого написано баг-репорт
ОС / Сервіс-пак / вид браузера та версія / ...	Автор баг-репорта (зазвичай це тестувальник)
Кроки відтворення (Steps to Reproduce)	Автор баг-репорта (зазвичай це тестувальник)
Фактичний результат (Result)	Автор баг-репорта (зазвичай це тестувальник)
Очікуваний результат (Expected Result)	Автор баг-репорта (зазвичай це тестувальник)
Прикріплений файл (Attachment)	Автор баг-репорта (зазвичай це тестувальник), а також будь-хто з членів команди (групи), який вважає, що прикріплені дані допоможуть у виправленні багу

Завдання

Оформити тест-кейс і баг-репорт. Наведемо приклад виконання завдання лабораторної роботи (табл. 2.9, 2.10).

Варіант 3

3	http://www.ameno.ru/	Mozilla Firefox	Сонник	Без реєстрації

Завдання 1. Оформлення тест-кейсу

Таблиця 2.9

Тест-кейс ID TSC-0001 Автор: Сидоров Микола

Назва Title	Пошук у соннику на головній сторінці з використанням російських слів		
Опис Description	При переході на «СОННИК» у процесі пошуку потрібного сну, указаного російськими словами, опис відображається неправильно		
Середовище Environment	Windows 7 (x64), Mozila Firefox 13.0.1		
Дія		Очікуваний результат	**Результат тесту:** пройдений (pass) **провалений (fail)** заблокований (blocker)

Закінчення табл. 2.9

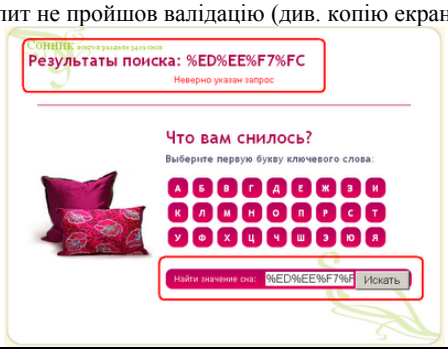
Передумова		
Відкриваємо сайт http://www.ameno.ru/	**Сайт www.ameno.ru відткритий та доступний**	
Кроки тесту:		
1. Відкрийте головну сторінку сайту http://www.ameno.ru 2. Опустіться, використавши скрол, донизу сторінки 3. Зліва зверху переходимо лівим кліком мишки на розділ: «СОННИК» (див. скріншот Sonnik.jpg — виділено рамкою) 4. Уведіть пошукове слово, наприклад, «ночь» 5. Натисніть кнопку «Знайти»	**Пошук надійшов вдало, опис потрібного сну показано правильно**	
Прикріплений файл: Sonnik.jpg **Attachment: Sonnik.jpg**		

Таблиця 2.10

Завдання 2. Оформлення баг-репорта ID TSR-0001

Короткий опис *(що не працює, де…, коли...?)*	Пошук у соннику на головній сторінці з використанням російських букв, працює неправильно
Опис **Description**	На сайті **http://www.ameno.ru** При переході на «СОННИК» для пошуку потрібного сну, вказаного російською мовою, описання відображається неправильно
Середовище **Environment**	Windows 7 (x64), Mozilla Firefox13.0.1
Номер версії	0.001
Важливість: S1 що Блокує (Blocker) S2 Критична (Critical) S3 Значна (Major) S4 Незначна (Minor) S5 Тривіальна (Trivial)	**S3 Значна (Major)**
Пріоритет: P1 Високий (High) P2 Середній (Medium) P3 Низький (Low)	**P1 Високий (High)** Іноді заповнюється менеджером

Закінчення табл. 2.10

Статус	Відкритий (Open)
Автор	Віталій Сокіл
Назначений на	Ім’я розробника
Передумови (якщо потрібні)	—
Кроки відтворення	1. Відкриваємо головну сторінку сайту **http://www.ameno.ru** 2. Опускаємося за допомогою скролу вниз сторінки 3. Ліворуч угорі переходимо лівим кліком мишки на розділ: «СОННИК» (див.скріншот Sonnik.jpg-виділено рамкою)  4. Уведіть пошукове слово, наприклад, «ночь» 5. Натисніть кнопку «Знайти»
Фактичний результат	Запит не пройшов валідацію (див. копію екрана) 
Очікуваний результат	Пошук виконано вдало, опис потрібного сну показано правильно

Запитання для самоперевірки

1. Яка структура та основні поля баг/дефект-репорта?
2. Які правила побудови ефективного тест-кейсу?

Література: [6]; [10–15].

ДОДАТОК

Номер варіанта	Назва предметної області
1	Діяльність туристичної фірми
2	Робота Інтернет-магазину «Просто-Сад»
3	Автоматизація роботи біржі праці
4	Діяльність рекрутингової агеції
5	Автоматизація роботи бібліотеки
6	Брокерська система продажу квартир
7	Автоматизована система роботи перукарні
8	Система обміну валют у пункті обміну
9	Система нарахування стипендії студентам
10	Система проходження практики в ІТ-фірмах
11	Організація оренди нерухомості
12	Робота працівника відділу кадрів з роботи зі студентами
13	Система роботи ювелірного магазину
14	Система поселення студентів у гуртожиток
15	Система вступу до магістратури
16	Робота фотостудії
17	Робота хімчистки
18	Діяльність відділу реклами
19	Робота Інтернет-магазину з продажу побутової техніки
20	Автоматизоване робоче місце юриста
21	Робота кадрового агентства
22	Автоматизоване робоче місце працівника кінотеатру
23	Система резервування і доставки замовлень у ресторані
24	Діяльність підприємства «Будинок музики»
25	Аудиторська перевірка організації
26	Робота інтернет-провайдера з обслуговування клієнтів
27	Обслуговування клієнтів готелю
28	Діяльність букмекерської контори
29	Робота метеорологічної станції
30	Робота підприємства з розроблення програмних продуктів

СПИСОК ЛІТЕРАТУРИ

1. *Лэффенгуэлл Д.* Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Лэффенгуэлл, Д. Уидриг; пер. с англ. — М. : Вильямс, 2002. — 448 с.
2. *Вигерс К.* Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти; пер. с англ. — М. : Издательско-торговый дом «Русская Редакция», СПб. : БХВ Петербург, 2014. —736 с.
3. *Халл Э.* Разработка и управление требованиями. Практическое руководство пользователя / Э. Халл, К.Джексон, Дж. Дик; пер. с англ. — М. : Изд-во «Telelogic», 2005. — 230 с.
4. *Андон Ф. И.* Основы инженерии качества программных систем / Ф. И. Андон, Г. И. Коваль, Т. М. Коротун [и др.]. — К. : Академпериодика, 2007. — 672 с.
5. *Райчев І. Е.* Принципи проектування відкритих розподілених систем : навч. посібник / І. Е. Райчев, О. Г. Харченко, В. В. Замковий. — К. : Вид-во нац. авіац. ун-ту «НАУ-друк», 2010. — 240 с.
6. *Зіатдінов Ю. К.* Стандартизація та сертифікація інформаційних управляючих систем : навч. посібник / Ю. К. Зіатдінов, І. Е. Райчев, О. Г. Харченко. — К. : НАУ, 2016. — 184 с.
7. *Райчев І. Е.* Принципи проектування відкритих розподілених систем: Об'єктно-орієнтоване проектування інформаційних систем : лабораторний практикум / І. Е.Райчев, О. Г.Харченко. — К. : НАУ, 2007. — 64 с.
8. *Арлоу Д.* UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, А. Нейштадт; пер. с англ. — СПб. : Символ-Плюс, 2007. — 624 с.
9. *Буч Г.* UML. Классика компьютерных технологий / Г. Буч, А. Якобсон, Дж. Рамбо; пер. с англ. — СПб. : Питер, 2006. — 736 с.
10. *Ильин В. В.* Моделирование бизнес-процессов. Практический опыт разработчика / В. В.Ильин. — М. : Вильямс, 2006. — 176 с.
11. *Брауде Э.* Технология разработки программного обеспечения / Э.Брауде; пер. с англ. — СПб. : Питер, 2004. — 655 с.
12. *Соммервилл И.* Инженерия программного обеспечения / И. Соммервилл; пер. с англ. — М. : Вильямс, 2002. — 624 с.
13. *Грекул В. И.* Проектирование информационных систем / В. И. Грекул, Г. Н. Денисенко, Н. Л. Коровкина. — М. : БИНОМ, 2008. — 300 с.
14. *Лавріщева К. М.* Програмна інженерія : підручник / К. М. Лавріщева. — К. : Академперіодика, 2008. — 320 с.
15. *Канер С.* Тестирование программного обеспечения / С. Канер, Дж. Фолк., Нгуен Енг Кен; пер. с англ. — К. : Диасофт, 2001. — 544 с.

ЗМІСТ

Навчальне видання

ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Лабораторний практикум
для студентів спеціальності 122
«Комп'ютерні науки»

Укладач
РАЙЧЕВ Ігор Едуардович

Редактор *Р. М. Шульженко*
Технічний редактор *А. І. Лавринович*
Коректор *О. О. Крусь*
Комп'ютерна верстка *Л. Т. Колодіної*

Підп. до друку 20.08.2018. Формат 60×84/16. Папір офс.
Офс. друк. Ум. друк. арк. 3,95. Обл.-вид. арк. 4,25.
Тираж 100 пр. Замовлення № 103-1.

Видавець і виготівник
Національний авіаційний університет
03680. Київ-58, проспект Космонавта Комарова, 1
Свідоцтво про внесення до Державного реєстру ДК № 977 від 05.07.2002