МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КПІ ім. І. СІКОРСЬКОГО»

Управління ІТ-проектами

МЕТОДИЧНІ ВКАЗІВКИ
до виконання комп'ютерних практикумів

Київ – 2017

Міністерство освіти і науки України
Національний технічний університет України
«КПІ ім. І. Сікорського»

Управління ІТ-проектами

Методичні вказівки

до виконання комп’ютерних практикумів
для студентів напряму підготовки 6.050101 – "Комп'ютерні науки"

Рекомендовано вченою радою
факультету біомедичної інженерії НТУУ «КПІ ім. І. Сікорського»

Київ
НТУУ «КПІ ім. І. Сікорського»
2017

Основи програмування: методичні вказівки до виконання комп’ютерних практикумів з дисципліни «Управління ІТ-проектами». Управління ІТ-проектами. / Уклад.: А. В. Яковенко, О. О. Коновал. – К.: НТУУ «КПІ ім. І. Сікорського», 2017. – 47 с.

Рекомендовано Вченою радою ФБМІ НТУУ «КПІ ім. І. Сікорського»
(Протокол № _11_ від __24.05.2017_ р.)

Навчально-методичне видання

Управління ІТ-проектами

Методичні вказівки

до виконання комп’ютерних практикумів

для студентів напряму підготовки 6.050101 – "Комп'ютерні науки"

Укладачі: *А. В. Яковенко, к.т.н.*
О. О. Коновал

Відповідальний редактор: *Носовець Олена Костянтинівна, к.т.н.*

Рецензент: *Юрчишин Василь Якович, с.н.с, к.т.н., доцент*

ЗМІСТ

ВСТУП

Предмет навчальної дисципліни «Управління ІТ-проектами» – процес навчання і підготовки зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» за спеціалізацією «Інформаційні технології в біології та медицині» першого (бакалаврського) рівня вищої освіти ступеня бакалавра, який дозволить використовувати базові поняття та методи управління проектами з використанням сучасних технологій і засобів, які їх підтримують, що відіграє важливу роль в біомедичних застосуваннях.

В структурно-логічній схемі програми підготовки фахівця:

– дисципліну **забезпечують** наступні дисципліни та кредитні модулі: "Технології створення програмних продуктів", "Організація баз даних та знань", "Об'єктно-орієнтоване програмування", "Алгоритмізація та програмування", "Системний аналіз", "Теорія алгоритмів";

– дисципліна **забезпечує** наступні навчальні дисципліни та кредитні модулі: дисципліна – завершальна в циклі та є основою для підготовки дипломних робіт за спеціальністю та в подальшій практичній роботі за фахом..

Метою навчальної дисципліни є формування у студентів **здатностей**:

- здатність використання методів оцінки ІТ-проектів, розробки плану створення програмних продуктів та систем;
- здатність використання принципів планування потоків робіт;
- здатність використання методів оцінки ризиків ІТ-проектів;
- здатність використання методів контролю за перебігом виконання проекту.

Основні завдання навчальної дисципліни.

Згідно з вимогами освітньо-професійної програми студенти після засвоєння навчальної дисципліни мають продемонструвати такі результати навчання:

знання:

- задач менеджера проекту на всіх фазах циклу розробки програмного забезпечення;
- методики застосування PERT-аналізу для розрахунку термінів виконання та бюджету проекту;
- методів визначення ризиків ІТ-проектів, їх класифікацію, стратегію управління ризиками;
- методів контролю за виконання проекту;
- принципів побудови, складу та призначення апаратних та програмних комплексів, що забезпечують управління проектами;

вміння:

- вибирати стратегії для планування життєвого циклу системи (1.ПФ.Д.02.01, рівень сформованості Р);
- визначати організаційну, економічну, технічну та операційну здійсненність проекту (1.ПФ.Д.02.02, рівень сформованості Р);

- аналізувати організаційне оточення, існуючі системи, синтезувати вимоги до системи (1.ПФ.Д.02.03, рівень сформованості Р);
- проектувати та моделювати бізнес-процеси системи (1.ПФ.Е.03.01, рівень сформованості Р);
- розробляти проектну та робочу документації системи (3.ПФ.Е.03.01, рівень сформованості Р);
- розробляти технічні інструкції (3.ПФ.Е.03.02, рівень сформованості Р);

досвід в системі типових завдань діяльності:

- застосовувати набуті знання в професійній діяльності під час планування та управління проектом;
- розробляти проектну документацію та програми на базі оптимального використання сучасних технологій;
- працювати в команді.

При виконанні комп'ютерних практикумів, студенти мають здобути навички розробки технічного завдання, оцінки проекту, розробки плану робіт, моделювання бізнес-процесів програмних систем біомедичного призначення. Аналізувати можливі ризики та розробляти анти ризикові заходи. Виконувати проміжний контроль за ходом виконання проекту. Методичні вказівки розроблені для проведення занять за допомогою частково-пошукового методу, який сприяє активному пошуку розв'язання поставлених задач та продуктивному аналізу одержаних результатів, для студентів зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» за спеціалізацією «Інформаційні технології в біології та медицині» першого (бакалаврського) рівня вищої освіти ступеня бакалавра факультету біомедичної інженерії.

В методичних вказівках практикум побудований таким чином, що студент ознайомлюється з теоретичними відомостями відповідно до зазначеної теми, може проглянути приклад виконання типових задач для розв'язання поставленої та проаналізувати одержані результати. Для самоконтролю студенти даного курсу дисципліни можуть скористуватись питаннями для самоконтролю, що забезпечить повторення та закріплення пройденого матеріалу.

Комп'ютерний практикум № 1. Аналіз предметної галузі

Мета роботи: відповідно до варіанта завдання, знайти приклади 3-4 аналогічних програмних систем та проаналізувати принципи їх функціонування. На основі отриманих даних створити проект технічного завдання для розробки власної системи.

Звіт повинен складатися з двох частин.

Перша частина містить аналіз аналогічних систем (не менше 3, можна іноземні аналоги). Для кожного аналогу ПЗ визначити наступні характеристики:

− назва;
− розробник (дистриб'ютор);
− архітектура (desktop application, client-server, 3tier web application);
− мова реалізації;
− перелік функцій, характеристик (не менше 5);
− аналіз переваг та недоліків даного ПЗ;
− джерело інформації (веб-сайт);

Друга частина визначає перелік характеристик системи, що проектується та розробляється студентом. Для програмної системи, що проектується визначити:

− призначення ПЗ;
− основні функції (не менше 5);
− користувачі системи (2-3 користувачі);
− архітектура системи;
− опис структури (схема);
− засоби апаратної та програмної реалізації (платформа, ОС та мова програмування);
− output (вихідні дані): таблиці, звіти, графіки (не менше 3).

ПЕРЕЛІК ВАРІАНТІВ ЗАВДАННЯ

1. Електронна система продажу ліків в аптеці.
2. Медична інформаційна система.
3. Лабораторна інформаційна система.
4. Інтернет-магазин аптеки.
5. Редактор медичних зображень.
6. Система документообігу в закладі охорони здоров'я.
7. Система для дослідження головного мозку.
8. Система для дослідження органів дихання.
9. Система для ультразвукових досліджень.
10. Система обліку медичних товарів на складі.
11. Система підтримки складання розкладу.
12. Електронна система для лабораторних досліджень.
13. Автоматизована аптечна система.
14. GPS навігатор для машини швидкої допомоги.
15. Інтернет-магазин медичного обладнання.
16. Інтернет-сайт закладу охорони здоров'я.
17. Он-лайн система довідкової інформації з лікарських засобів та даних про наявність медикаментів і ціни на них в аптеках.

Комп'ютерний практикум № 2. Методологія структурного аналізу та проектування SADT. Методологія функціонального моделювання IDEF0

Мета роботи: виконати опис моделі, визначити мету, точку зору, межі, створити контекстну діаграму А-0 і діаграму її декомпозиції А0. Виконати подальшу декомпозицію функцій, використовуючи засоби навігації по моделі.

ЗАВДАННЯ

У середовищі BPwin розробити функціональну модель для обраного варіанта комп'ютерної системи.

Модель повинна містити контекстну діаграму.

Визначити мету, точку зору моделі. Описати властивості у відповідних закладках діалогу Model Properties.

Задати входи, виходи, механізми і управління.

Створити декомпозицію контекстної діаграми, що складається з 3-4 блоків.

Встановити зв'язки між блоками. Задати імена дуг.

Модель повинна включати діаграму дерева вузлів.

Теоретичні відомості

Алгоритм аналізу системи

Збір інформації про систему. Вибір мети і точки зору

Це основоположні параметри моделі. Мета вибирається на основі переліку питань, на які модель повинна відповісти. Точка зору представляє позицію, з якої описується система.

Створення діаграми А-0 і діаграми декомпозиції А0

1. Складання списку даних. Це список об'єктів, що мають значення на даному рівні декомпозиції. Функціональна декомпозиція більш ефективна, якщо починається зі складання списку даних.

2. Складання списку функцій. Це список функцій, які оперують з даними з попереднього списку. Кілька функцій можуть мати одні й ті ж дані, в той же час одна функція може використовувати кілька різних типів даних.

3. Побудова діаграми:

- Розташувати блоки на сторінці (з урахуванням домінування),
- Намалювати основні дуги, що представляють обмеження,
- Намалювати зовнішні інтерфейсні дуги,
- Намалювати всі що залишилися дуги.

Створення діаграми А-0

Узагальнення діаграми А0 призводить до отримання верхньої діаграми моделі А-0. Ця операція дуже важлива, так як дозволяє перевірити чи адекватно назва моделі того, що робить система, переконатися в повноті зовнішніх інтерфейсів системи (дуг), виконати остаточне затвердження цілі і точки зору моделі. Створення діаграми А-0 полягає в зображенні блоку А0 і записи мети і точки зору під цим блоком.

Подальша декомпозиція (декомпозиція обмежених об'єктів)

Діаграма першого рівня декомпозиції А0, а також всі наступні діаграми декомпозиції, надають інтерфейсні обмеження (контекст) для дочірніх діаграм. Крім того, модель вже має на меті і точкою зору. Це робить процес подальшого

проектування більш формалізованим і необхідний ступінь деталізації досягається виконанням наступного рекурсивного процесу:

1. Вибір блоку діаграми. Декомпозицію рекомендується починати з самого змістовного блоку з точки зору домінування, функціональної складності і впливу на декомпозицію інших блоків цієї діаграми. Кращим для початку декомпозиції не обов'язково буде найскладніший для розуміння блок.

2. Розгляд об'єкта, визначеного цим блоком.

3. Створення нової діаграми (за алгоритмом, подібного побудови діаграми А0).

4. Виявлення недоліків нової діаграми.

5. Створення альтернативних декомпозицій.

6. Коригування нової діаграми.

7. Коригування всіх пов'язаних з нею діаграм.

Коли зупиняти декомпозицію

Мета моделювання містить список питань, на які повинна відповідати модель. Коли ми можемо по діаграмах, що становить модель, знайти відповіді на ці питання, мета моделювання вважається досягнутою загальні витрати на виготовлення моделі можна припиняти.

Якщо ж мета моделювання ще не досягнута, то необхідно виконувати декомпозицію тих функцій, розуміння роботи яких дасть відповіді на необхідні питання. При цьому необхідно строго дотримуватися кордону моделі, закладені в контекстної діаграмі, і діаграми декомпозиції першого рівня А0. При необхідності опис функцій нижнього рівня можна проводити з використанням інших методологій: IDEF3 або DFD, призначених для детального опису процесів.

SADT - одна з найвідоміших і широко використовуваних систем проектування. SADT - Structured Analysis and Design Technique (Технологія структурного аналізу і проектування) - це графічні позначення і підхід до опису систем.

Розроблено кілька графічних мов моделювання, які отримали назви:

- Нотація IDEF0 - для документування процесів виробництва і відображення інформації про використання ресурсів на кожному з етапів проектування систем. Функціональне моделювання.

- Нотація IDEF1 - для документування інформації про виробниче оточення системи.

- Нотація IDEF2 - для документування поведінки системи в часі.

- Нотація IDEF3 - спеціально для моделювання бізнес-процесів.

Найбільш зручною мовою моделювання бізнес-процесів є IDEF0, де система представляється як сукупність взаємодіючих робіт або функцій. Така чисто функціональна орієнтація є принциповою. Функції системи аналізуються незалежно від об'єктів, якими вони оперують. Це дозволяє більш чітко змоделювати логіку і взаємодію процесів організації.

Основу методології IDEF0 складає графічна мова опису бізнес-процесів. Модель в нотації IDEF0 являє собою сукупність ієрархічно впорядкованих і взаємопов'язаних діаграм. Кожна діаграма є одиницею опису системи і розташовується на окремому аркуші.

Модель може містити чотири типи діаграм:

• контекстну діаграму (у кожній моделі може бути тільки одна контекстна

діаграма);

• діаграми декомпозиції;

• діаграми дерева вузлів;

• діаграми тільки для експозиції (FEO).

Методологія IDEF-SADT являє собою сукупність методів, правил і процедур, призначених для побудови функціональної моделі системи будь-якої предметної області. Функціональна модель SADT відображає структуру процесів функціонування системи і її окремих підсистем, тобто виконувані ними дії і зв'язки між цими діями. Для цієї мети будуються спеціальні моделі, які дозволяють у наочній формі представити послідовність певних дій. Вихідними функціональними блоками будь-якої моделі IDEF0 процесу є діяльність (activity) і стрілки (arrows).

Діяльність є деяка дія або набір дій, які мають фіксовану мету і призводять до деякого кінцевого результату. Іноді діяльність називають просто процесом. Моделі IDEF0 відстежують різні види діяльності системи, їх опис та взаємодія з іншими процесами. На діаграмах діяльність або процес зображується прямокутником, який називається блоком. Стрілка служить для позначення деякого носія або впливу, які забезпечують перенесення даних або об'єктів від однієї діяльності до іншої. Стрілки також необхідні для опису того, що саме виробляє діяльність і які ресурси вона споживає. Це так звані ролі стрілок - ICOM - скорочення перших букв від назв відповідних стрілок IDEF0. При цьому розрізняють стрілки чотирьох видів (Рис. 2.1):

- I (Input) - вхід, тобто все, що надходить в процес або споживається процесом.
- C (Control) - управління або обмеження на виконання операцій процесу.
- O (Output) - вихід або результат процесу.
- M (Mechanism) - механізм, який використовується для виконання процесу.

Методологія IDEF0 однозначно визначає, яким чином зображуються на діаграмах стрілки кожного виду ICOM. Стрілка Вхід (Input) виходить з лівого боку рамки робочого поля і входить зліва в прямокутник процесу. Стрілка Управління (Control) входить і виходить зверху. Стрілка Вихід (Output) виходить з правого боку процесу і входить в праву сторону рамки. Стрілка Механізм (Mechanism) входить в прямокутник процесу знизу.

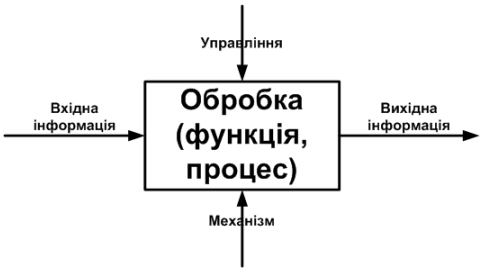


Рис. 2.1. Позначення процесу і стрілок ICOM на діаграмах IDEF0

Однією з найбільш важливих особливостей методології IDEF-SADT є поступове введення все більш детальних уявлень моделі системи у міру розробки окремих діаграм. Побудова моделі IDEF-SADT починається з отримання всієї системи у вигляді найпростішої діаграми, що складається з одного блоку процесу і стрілок ICOM, службовців для зображення основних видів взаємодії з об'єктами поза системою. Оскільки вихідний процес представляє всю систему як єдине ціле, дане подання є найбільш загальним і підлягає подальшій декомпозиції.

Декомпозиція є умовним прийомом, що дозволяє уявити систему у вигляді, зручному для сприйняття, і оцінити її складність. В результаті декомпозиції підсистеми за певними ознаками виділяються окремі структурні елементи та зв'язки між ними. Декомпозиція служить засобом, що дозволяє уникнути труднощів у розумінні системи. Глибина декомпозиції визначається складністю і розмірністю системи, а також цілями моделювання (Рис. 2.2).

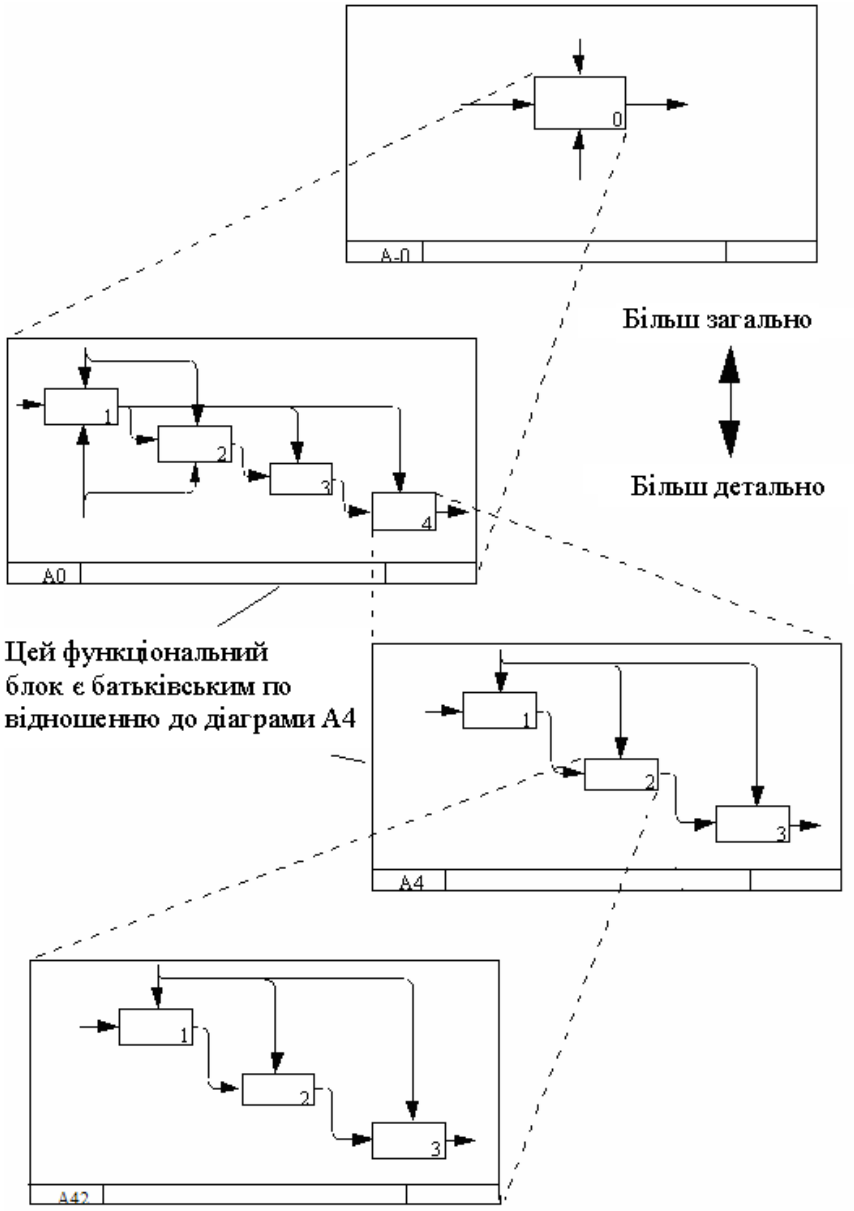


Рис. 2.2. Декомпозиція функціональних блоків

Зазвичай IDEF0-моделі несуть в собі складну і концентровану інформацію, і для того, щоб обмежити їх перевантаженість і зробити легким для читання, у відповідному стандарті прийняті відповідні обмеження складності:

- Обмеження кількості функціональних блоків на діаграмі трьома-шістьма. Верхня межа (шість) змушує розробника використовувати ієрархії при описі складних предметів, а нижня межа (три) гарантує, що на відповідній діаграмі досить деталей, щоб виправдати її створення;

- Обмеження кількості відповідних до одного функціонального блоку (що виходять з одного функціонального блоку) інтерфейсних дуг чотирма.

Типи зв'язків між функціями

Одним з важливих моментів при проектуванні ІС за допомогою методології SADT є точна узгодженість типів зв'язків між функціями. Розрізняють принаймні сім типів зв'язування (табл. 2.1.).

Таблиця 2.1. Типи зв'язування

Тип зв'язку	Відносна значимість
Випадковий	0
Логічний	1
Тимчасовий	2
Процедурний	3
Комунікаційний	4
Послідовний	5
Функціональний	6

(0) Тип випадкової зв'язності

Цей тип найменш бажаний. Випадкова зв'язність виникає, коли конкретний зв'язок між функціями малий або повністю відсутній. Це відноситься до ситуації, коли імена даних на SADT-дугах в одній діаграмі мають малий зв'язок один з одним. Крайній варіант цього випадку показаний на рис. 2.3.

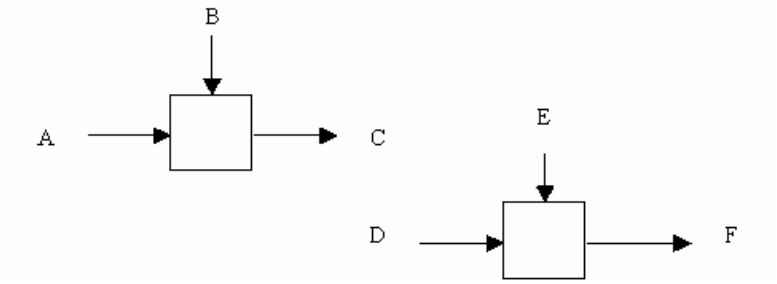


Рис. 2.3. Випадкова зв'язність

(1) Тип логічної зв'язності

Логічне зв'язування відбувається тоді, коли дані і функції збираються разом внаслідок того, що вони потрапляють у загальний клас або набір елементів, але необхідних функціональних відносин між ними не виявляється.

(2) Тип тимчасової зв'язності

Пов'язані з часом елементи виникають внаслідок того, що вони представляють функції, що пов'язані в часі, коли дані використовуються одночасно або функції вмикаються паралельно (рис. 2.4), а не послідовно.

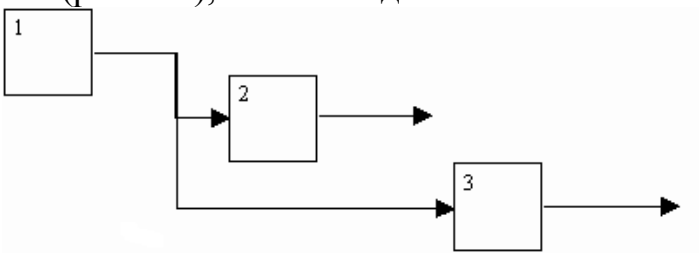


Рис. 4. Тимчасова зв'язність

(3) Тип процедурної зв'язності

Процедурно-пов'язані елементи з'являються згрупованими разом внаслідок

того, що вони виконуються протягом однієї і тієї ж частини циклу або процесу. Приклад процедурно-зв'язаної діаграми приведений на рис. 2.5.

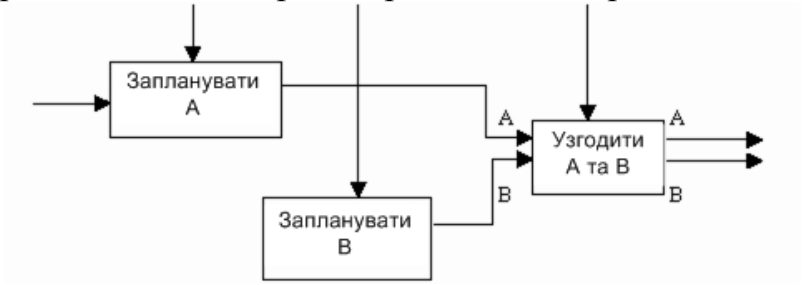


Рис. 2.5. Процедурна зв'язність

(4) Тип комунікаційної зв'язності

Діаграми демонструють комунікаційні зв'язки, коли блоки групуються внаслідок того, що вони використовують одні й ті ж вхідні дані і / або виробляють одні й ті ж вихідні дані (рис. 2.6).

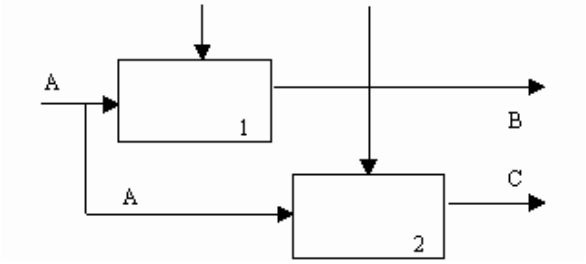


Рис. 2.6. Комунікаційна зв'язність

(5) Тип послідовної зв'язності

На діаграмах, що мають послідовні зв'язки, вихід однієї функції служить вхідними даними для наступної функції. Зв'язок між елементами на діаграмі є більш тісним, ніж на розглянутих вище рівнях зв'язок, оскільки моделюються причинно-наслідкові залежності (рис. 2.7).

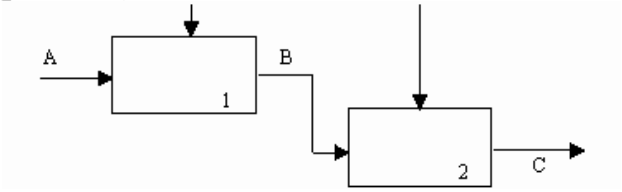


Рис. 2.7. Послідовна зв'язність

(6) Тип функціональної зв'язності

Діаграма відображає повну функціональну зв'язність, за наявності повної залежності однієї функції від іншої. Діаграма, яка є чисто функціональною, не містить чужорідних елементів, що відносяться до послідовного або слабшого типу зв'язності. Одним із способів визначення функціонально-пов'язаних діаграм є розгляд двох блоків, пов'язаних через управляючі дуги, як показано на рисунку 2.8.

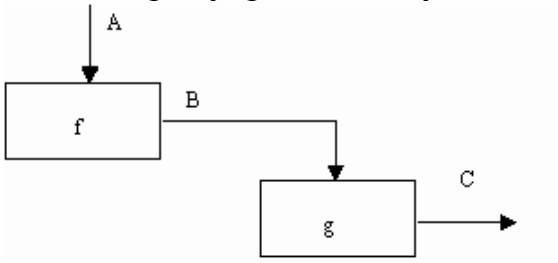


Рис.2.8. Функціональна зв'язність

Отже, модель IDEF-SADT являє собою серію взаємопов'язаних діаграм із супровідною документацією, яка розбиває вихідне уявлення складної системи на окремі складові частини. Деталі кожного з основних процесів представляються у вигляді більш детальних процесів на інших діаграмах. У цьому випадку кожна діаграма нижнього рівня є декомпозицією деякого процесу з більш загальної діаграми. Тому па кожному кроці декомпозиції більш загальна діаграма конкретизується на ряд більш детальних діаграм.

Стандарт IDEF0 містить набір процедур, що дозволяють розробляти і погоджувати модель великою групою людей, що належать до різних областей діяльності, що моделюється. Зазвичай процес розробки є ітеративним і складається з наступних умовних етапів:

- Створення моделі групою фахівців, що відносяться до різних сфер діяльності. Ця група в термінах IDEF0 називається авторами (Authors). Побудова первинної моделі є динамічним процесом, протягом якого автори опитують компетентних осіб про структуру різних процесів. На основі наявних положень, документів і результатів опитувань створюється чернетка (Model Draft) моделі.

- Поширення чернетки для розгляду, погоджень і коментарів. На цій стадії відбувається обговорення чернетки моделі з широким спектром компетентних осіб (в термінах IDEF0 - читачів). При цьому кожна з діаграм чорнової моделі письмово критикується і коментується, а потім передається автору. Автор, в свою чергу, також письмово погоджується з критикою або відкидає її з викладом логіки прийняття рішення і знову повертає відкоригований чернетку для подальшого розгляду. Цей цикл триває доти, поки автори і читачі не прийдуть до єдиної думки.

- Офіційне затвердження моделі. Затвердження узгодженої моделі відбувається керівником робочої групи в тому випадку, якщо у авторів моделі і читачів відсутні розбіжності з приводу її адекватності. Остаточна модель являє собою узгоджене уявлення про підприємство (системі) з заданої точки зору і для заданої мети.

Наочність графічного мови IDEF0 робить модель цілком читається і для осіб, які не брали участі в проекті її створення, а також ефективної для проведення показів і презентацій. Надалі, на базі побудованої моделі можуть бути організовані нові проекти, націлені на виробництво змін в системі.

КОНТРОЛЬНІ ЗАПИТАННЯ:

Що визначає контекстна діаграма?

Які основні поняття використовуються при створенні функціональної діаграми IDEF0?

З якою метою будуються діаграми для експозиції (FEO).

Що означає поява "тунелів" на діаграмі?

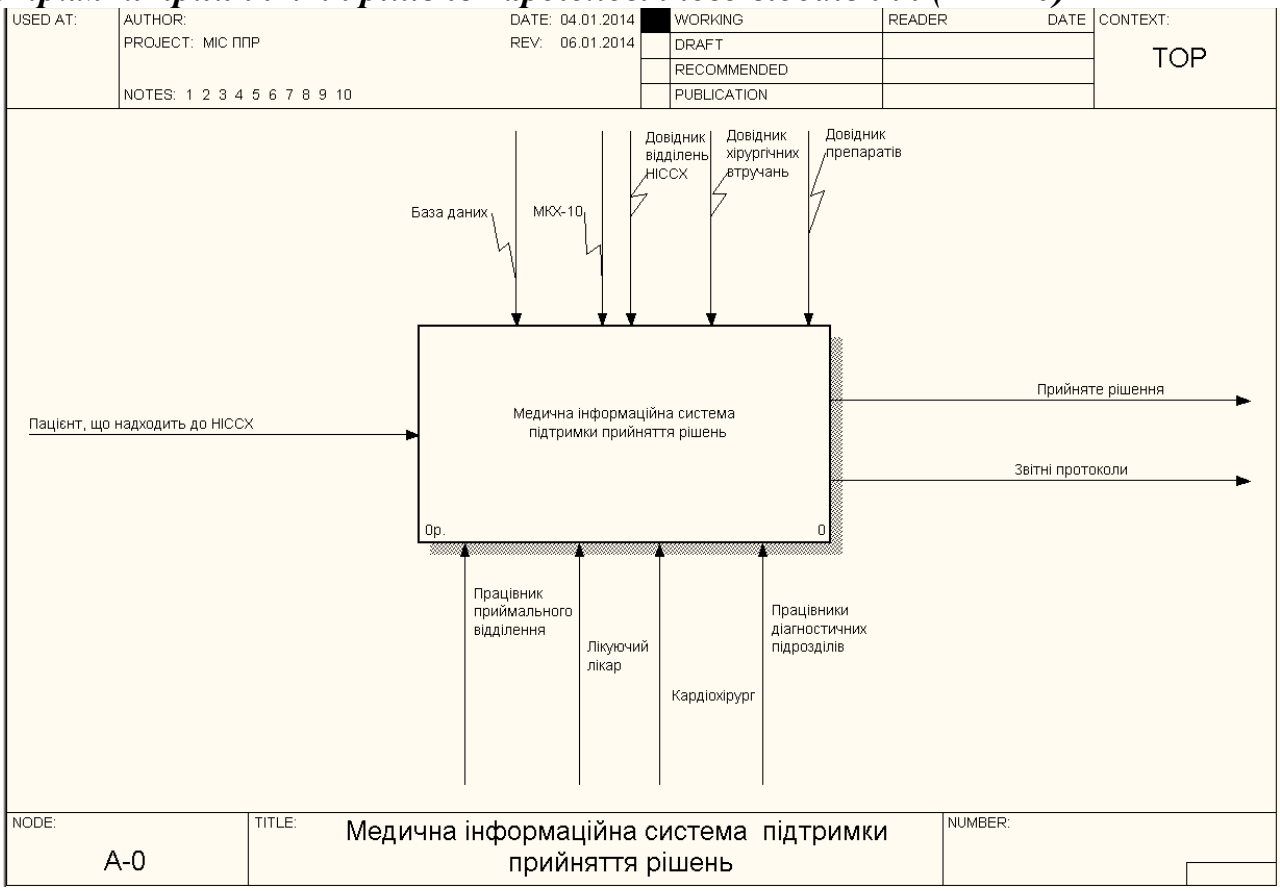
Що показує діаграма дерева вузлів.

Які стрілки називаються стрілками механізму (Mechanism)?

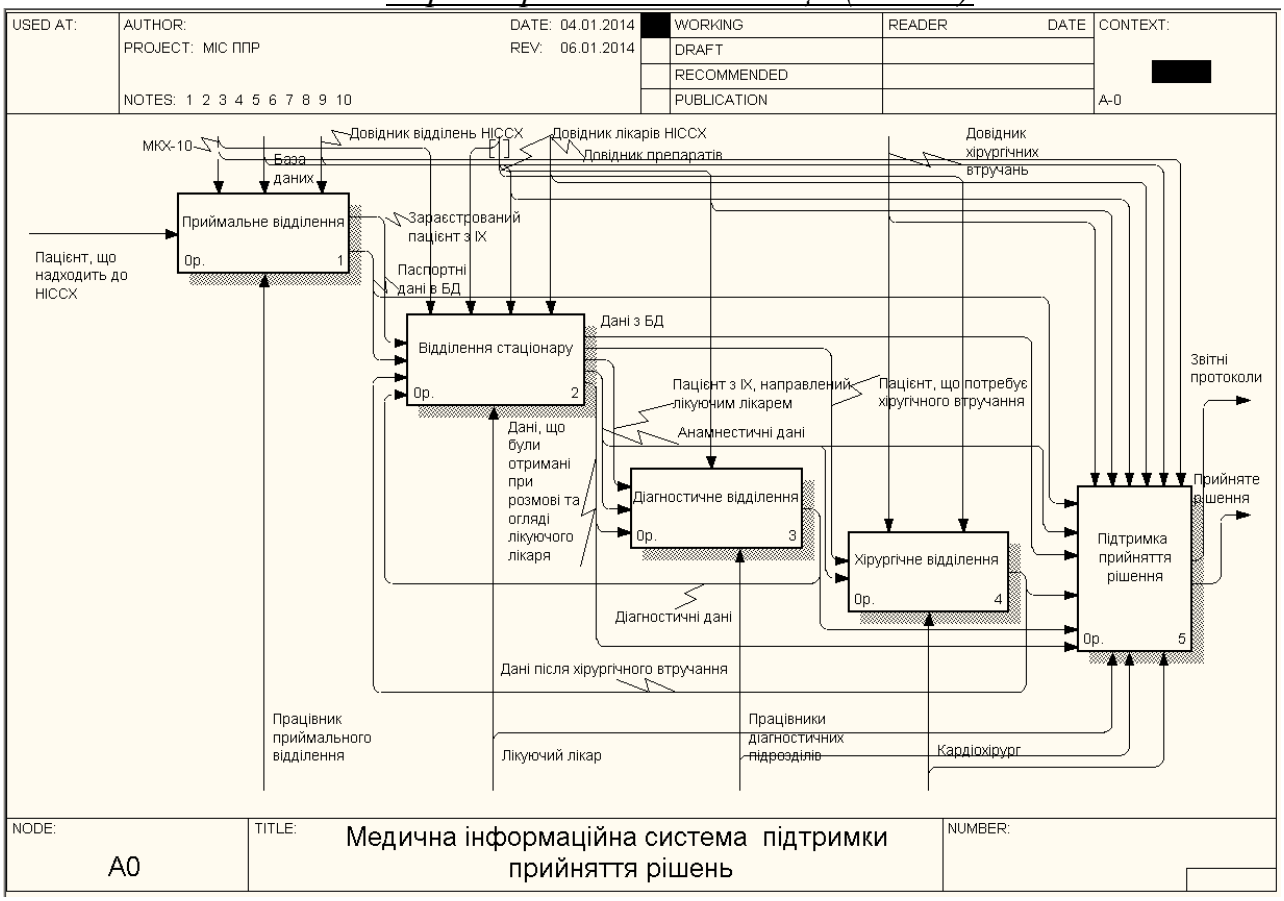
Що входить в визначення контексту моделі.

Які основні поняття використовуються при створенні функціональної діаграми IDEF0?

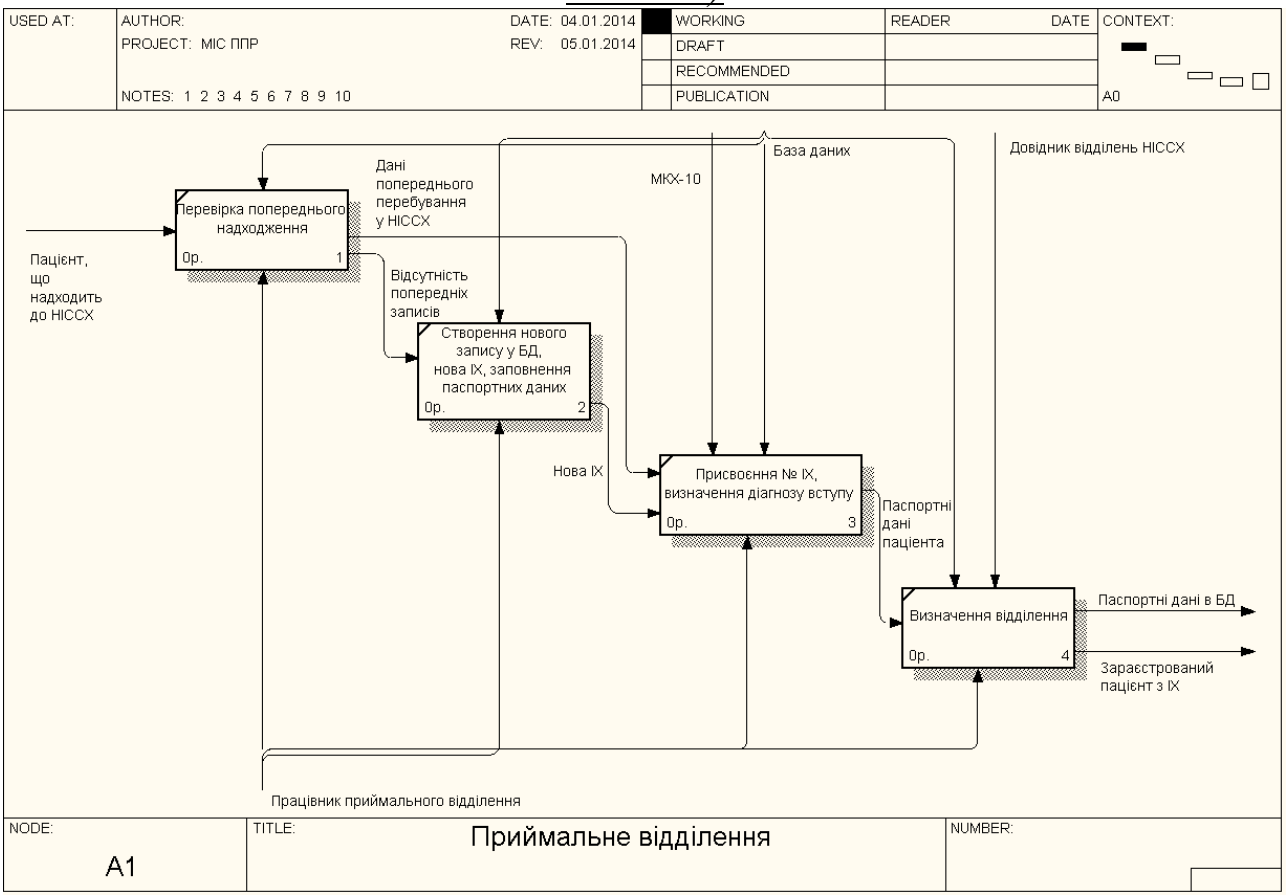
ПРИКЛАД. Модель бізнес-процесів медичної інформаційної системи підтримки прийняття рішень кардіологічного відділення (IDEF0)

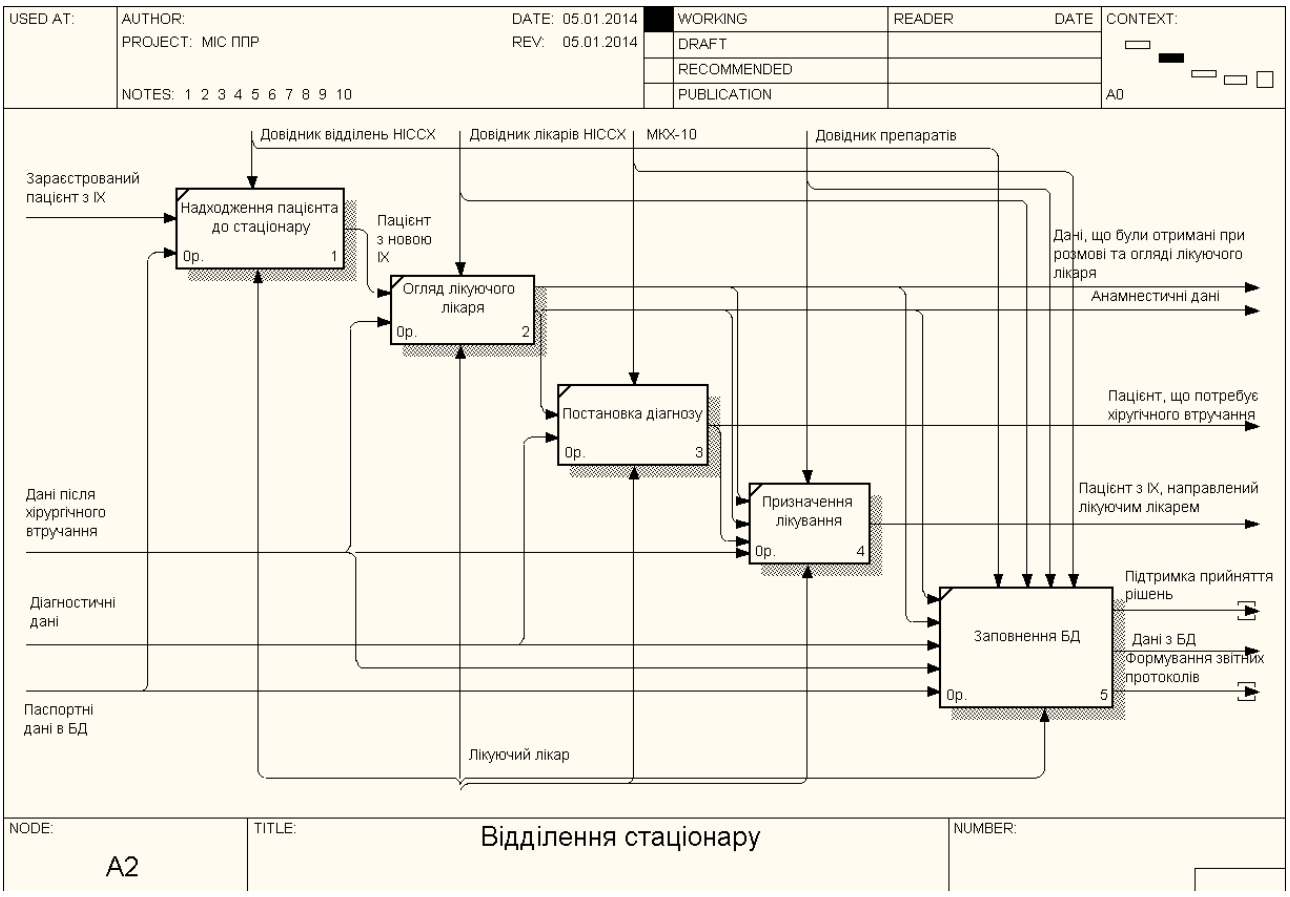


Перший рівень декомпозиції (IDEF0)



Діаграма декомпозиції IDEF0 (аналіз бізнес-процесів при впровадженні МІСППР)





Комп'ютерний практикум № 3. Методологія структурного аналізу та проектування SADT. Методологія функціонального моделювання IDEF3. Моделювання процесів в нотації DFD

Мета роботи: виконати декомпозицію функцій в стандарті IDEF3. Побудувати діаграму потоків даних в нотації DFD.

ЗАВДАННЯ

У середовищі BPwin на основі методології IDEF3 виконати декомпозицію нижніх рівнів ієрархічної структури робіт проекту. Для побудови моделі необхідно визначити функціональні можливості проекту, вхідні і вихідні дані, задіяні ресурси і т.д. Виконати моделювання процесів в нотації DFD.

Теоретичні відомості

Моделювання процесів в нотації IDEF3

Для опису логіки взаємодії інформаційних потоків більше підходить IDEF3, звана також workflow diagramming, - методологія моделювання, що використовує графічний опис інформаційних потоків, взаємин між процесами обробки інформації та об'єктів, що є частиною цих процесів. Діаграми Workflow можуть бути використані в моделюванні бізнес-процесів для аналізу завершеності процедур обробки інформації. З їх допомогою можна описувати сценарії дій співробітників організації, наприклад послідовність обробки замовлення або події, які необхідно обробити за кінцевий час. Кожен сценарій супроводжується описом процесу і може бути використаний для документування кожної функції.

IDEF3 - це метод, який має основною метою дати можливість аналітикам описати ситуацію, коли процеси виконуються в певній послідовності, а також описати об'єкти, які беруть участь спільно в одному процесі.

Техніка опису набору даних IDEF3 є частиною структурного аналізу. На відміну від деяких методик описів процесів IDEF3 не обмежує аналітика надмірно жорсткими рамками синтаксису, що може привести до створення неповних або суперечливих моделей.

IDEF3 може бути також використаний як метод створення процесів. IDEF3 доповнює IDEF0 і містить все необхідне для побудови моделей, які в подальшому можуть бути використані для імітаційного аналізу.

Кожна робота в IDEF3 описує будь-який сценарій бізнес-процесу і може бути складовою іншої роботи. Оскільки сценарій описує мету і рамки моделі, важливо, щоб роботи іменувалися віддієслівним іменником, що позначає процес дії, або фразою, що містить такий іменник.

Точка зору на модель повинна бути документально підтверджена. Зазвичай це точка зору людини, відповідального за роботу в цілому. Також необхідно документувати мету моделі - ті питання, на які покликана відповісти модель.

Діаграма є основною одиницею опису в IDEF3. Важливо правильно побудувати діаграми, оскільки вони призначені для читання іншими людьми.

Одиниці роботи - Unit of Work (UOW) - також звані роботами (activity), є центральними компонентами моделі. У IDEF3 роботи зображуються прямокутниками з прямими кутами і мають ім'я, виражене віддієслівним іменником, що позначає процес дії, одиночним або в складі фрази, і номер (ідентифікатор); інше

іменник в складі тієї ж фрази зазвичай відображає основний вихід (результат) роботи.

Зв'язки показують взаємини робіт. Все зв'язку в IDEF3 односпрямовані і можуть бути спрямовані куди завгодно, але зазвичай зв'язку намагаються направити зліва направо. У IDEF3 розрізняють три типи стрілок, що зображують зв'язку:

• Старша (Precedence) - суцільна лінія, що зв'язує одиниці робіт (UOW). Вимальовується зліва направо або зверху вниз. Показує, що робота-джерело повинна закінчитися раніше, ніж робота-мета почнеться.

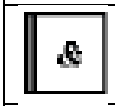
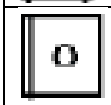
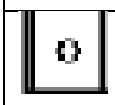
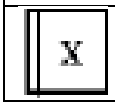
• Стрілка відношення (Relational Link) - пунктирна лінія, що використовується для зображення зв'язків між одиницями робіт (UOW) а також між одиницями робіт і об'єктами посилань.

• Стрілка потоку об'єктів (Object Flow) - стрілка з двома наконечниками, застосовується для опису того факту, що об'єкт використовується в двох або більше одиницях роботи, наприклад, коли об'єкт породжується в одній роботі і використовується в іншій.

Для відображення логіки взаємодії стрілок при злитті і розгалуженні або для відображення безлічі подій, які можуть або повинні бути завершені перед початком наступної роботи, використовуються перехрестя (Junction). Розрізняють перехрестя для злиття (Fan-in Junction) і розгалуження стрілок (Fan-out Junction). Перехрестя не може використовуватися одночасно для злиття і для розгалуження. Сенс кожного типу наведено в таблиці 3.1.

Об'єкт посилання в IDEF3 виражає якусь ідею, концепцію або дані, які не можна пов'язати зі стрілкою, перехрестям або роботою. Як ім'я об'єкта можна використовувати ім'я будь-якої стрілки з інших діаграм або ім'я сутності з моделі даних. Об'єкти посилання повинні бути пов'язані з одиницями робіт або перехрестями пунктирними лініями. Офіційна специфікація IDEF3 розрізняє три стилі об'єктів посилань - безумовні (unconditional), синхронні (synchronous) і асинхронні (asynchronous).

Таблиця 3.1. Типи перехресть

Позначення	Найменування	В разі злиття стрілок (Fan-in Junction)	В разі розгалуження стрілок (Fan-out Junction)
	Asynchronous AND	Всі попередні процеси повинні бути завершені	Всі наступні процеси повинні бути запущені
	Synchronous AND	Всі попередні процеси завершені одночасно	Всі наступні процеси запускаються одночасно
	Asynchronous OR	Один або кілька попередніх процесів повинні бути завершені	Один або кілька наступних процесів повинні бути запущені
	Synchronous OR	Один або кілька попередніх процесів завершені одночасно	Один або кілька наступних процесів запускаються одночасно
	XOR (Exclusive OR)	Тільки один попередній процес завершений	Тільки один наступний процес запускається

При внесенні об'єктів посилань крім імені слід вказувати тип об'єкта посилання.

Типи об'єктів посилань наведені в таблиці 3.2.

Таблиця 3.2. Типи об'єктів посилань

Тип об'єкта посилання	Мета опису
OBJECT	Описує участь важливого об'єкта в роботі
GOTO	Інструмент циклічного переходу (в повторюваної послідовності робіт), можливо на поточній діаграмі, 36но не обов'язково. Якщо всі роботи циклу присутні на поточній діаграмі, цикл може також зображуватися стрілкою, що повертається на стартову роботу. GOTO може посилатися на перехрестя
UOB (Unit of behaviour)	Застосовується, коли необхідно підкреслити множинне використання будь-якої роботи, але без циклу. Наприклад, робота «Контроль якості» може бути використана в процесі «Виготовлення вироби» кілька разів, після кожної одиничної операції. Зазвичай цей тип посилання не використовується для моделювання автоматично запускаються робіт
NOTE	Використовується для документування важливої інформації, що відноситься до будь-яких графічних об'єктів на діаграмі. NOTE є альтернативою внесенню текстового об'єкта в діаграму
ELAB (Elaboration)	Використовується для удосконалення графіків або їх більш детального опису. Зазвичай вживається для детального опису розгалуження і злиття стрілок на перехрестях

У IDEF3 декомпозиція використовується для деталізації робіт. Методологія IDEF3 дозволяє виконати декомпозицію роботи багаторазово, тобто робота може мати безліч дочірніх робіт.

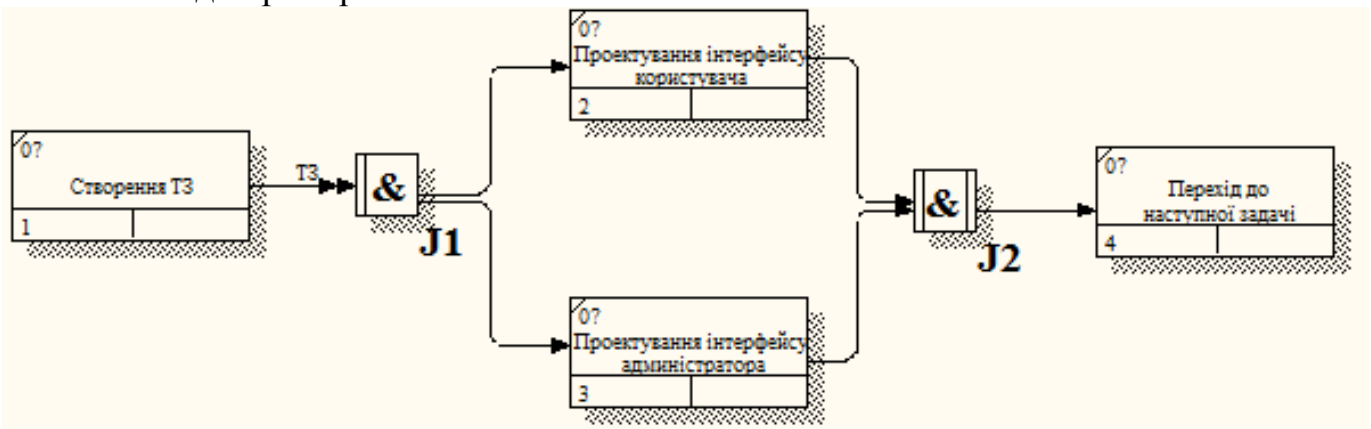


Рис. 3.1. Модель процесу в нотації IDEF3

Моделювання процесів в нотації DFD

Найважливішим способом опису процесу є діаграми потоків даних (інформації) DFD (Data Flow Diagram). Діаграми цього типу містять, як правило, два типи графічних об'єктів: чотирикутники і стрілки. Перші описують функції (роботи, процеси), другі – потоки даних між ними.

На діаграмі DFD функції зазвичай розташовуються зліва направо в порядку, відповідному послідовності їх виконання в часі, хоча це не обов'язково. Якщо дотримуватися зазначеної вимоги, то отримана схема – це опис процесу, схожим з

його описом в нотації IDEF3.

Опис процесу в DFD можна ускладнити, використовуючи поняття «сховище даних». Під ним мається на увазі будь-який носій інформації (наприклад, паперовий документ, електронний файл, промислова база даних на сервері організації). При побудові моделі процесу з використанням сховищ даних необхідно пам'ятати, що дані (інформація) не можуть переміщатися між функціями процесу самі по собі. Вони можуть бути передані тільки за допомогою певних посередників – носіїв інформації (тобто сховищ даних). На рис. 3.1 представлена модель процесу в нотації DFD, побудована з використанням поняття «сховище даних».

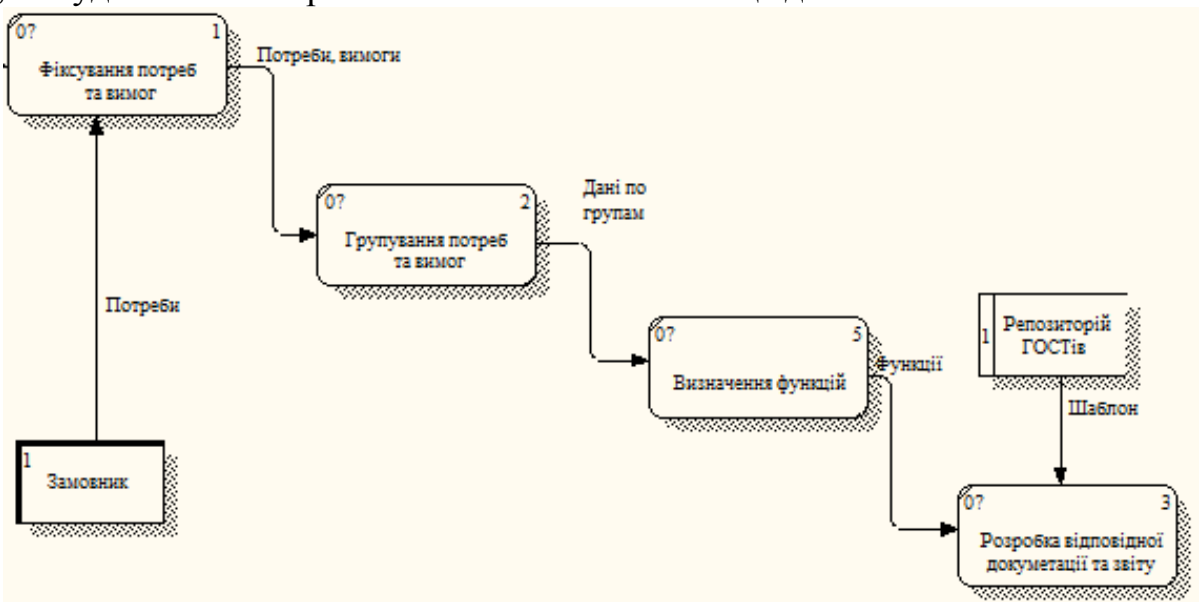


Рис. 3.1. Модель процесу в нотації DFD

Нотація DFD потрібна для опису реально існуючих в організації потоків даних. Описи можуть створюватися як за процесною, так і за функціональною ознакою. У першому випадку ми отримуємо моделі бізнес-процесів в форматі DFD, у другому – схему обміну даними між підрозділами. Створені моделі потоків даних організації можуть бути використані при вирішенні таких завдань, як:

- Визначення існуючих сховищ даних (текстових документів, файлів, СУБД);
- Визначення і аналіз даних, необхідних для виконання кожної функції процесу;
- Підготовка до створення моделі структури даних організації (так званої ERD-моделі);
- Виділення основних і допоміжних бізнес-процесів організації.

КОНТРОЛЬНІ ЗАПИТАННЯ:

Для чого необхідні IDEF3-моделі?

Які основні поняття використовуються при створенні функціональної діаграми IDEF3?

Для чого в діаграмах IDEF3 використовуються перехрестя?

Перерахуйте основні об'єкти IDEF3, їх опис і призначення.

Чим відрізняються синхронні перехрестя від асинхронних?

Що таке посилання?

Чому перехрестя «Виключаюче Або" не може бути синхронним?

При виконанні яких проектів краще всього використовувати DFD-методологію?

Комп'ютерний практикум № 4. Написання UseCases (варіантів використання системи)

Мета роботи: розробити варіанти використання системи.

ЗАВДАННЯ

Відповідно до обраної теми (варіанта) на основі проведеного аналізу прикладної галузі (Комп'ютерний практикум № 1) розробити три різні usecases (по одному в короткій, поверхневій та повній формах відповідно) для свого проекту. Повна форма опису має містити всі пункти наведені в таблиці 1. Головний успішний сценарій повинен мати не менше 10 кроків. Передбачити не менше 5 альтернативних сценаріїв.

Теоретичні відомості

Usecase – це текстовий опис сукупності сценаріїв, що виконуються користувачем при роботі з системою для досягнення певної мети.

Сценарій – послідовність дій при взаємодії користувача із системою для виконання певної операції.

Наприклад, купівля товарів у супермаркеті з використанням кредитної картки і невдала спроба купівлі через перевищення кредитного ліміту – це сценарії, а купівля товарів у супермаркеті – usecase.

Приклад 1. Зняття готівки в банкоматі.

Користувач підходить до банкомату та вставляє свою картку. Система перевіряє картку та просить ввести пін-код. Після перевірки пін-коду система виводить головне меню, де користувач обирає опцію видачі готівки. Він визначає суму грошей та підтверджує виконання операції. Банкомат видає готівку та чек. Система видає запит про здійснення наступної операції. Користувач обирає опцію «не виконувати», забирає картку та йде.

Приклад 2. Замовлення літератури в бібліотеці.

Головний сценарій (успішний):

Користувач підходить до бібліотекаря і надає йому замовлення (перелік видань). Бібліотекар заносить номер читацького квитка до системи і перевіряє облікову картку користувача. Бібліотекар шукає кожне видання в базі і визначає кількість екземплярів вільних у даний момент часу. Система видає шифри (коди), за якими бібліотекар знаходить видання у сховищі, заносить номери книг до картки користувача і видає йому літературу. При цьому читацький квиток лишається у бібліотекаря.

Альтернативні сценарії:

1. Користувач бібліотеки є боржником, бібліотекар не може видати йому літературу.

2. Бібліотекар не може видати користувачу літературу, оскільки в даний момент часу немає вільних екземплярів у сховищі.

3. Читацький квиток користувача є недійсним. Бібліотекар вилучає його.

4. Технічний збій роботи системи. Видається повідомлення «немає зв'язку з сервером баз даних». Бібліотекар викликає адміністратора системи.

Написання usecases дозволяє чітко визначити хто є користувачем системи, які її

сценарії роботи, що є метою використання системи. Usecases – це функціональні та поведінкові вимоги до системи, які показують, що саме вона має робити.

Існують три форми написання usecases:

1. Коротка – короткий опис в один абзац одного зі сценаріїв (зазвичай успішного) роботи системи (приклад 1). Виконуються під час початкового аналізу вимог до системи.

2. Поверхнева – поверхневий опис у вільній формі усіх сценаріїв (головного і альтернативних) одного з usecases (приклад 2). Виконуються під час початкового аналізу вимог до системи.

3. Повна – всі кроки і дії детально описані, включаючи перед- та постумови виконання usecase. Виконуються на стадії відбору з повного переліку usecases у короткій та поверхневій формах невеликої частини важливих (критичних для роботи системи) usecases.

Повна форма опису usecase має перелік розділів, коротко описаних у таблиці 4.1. Нижче наведено приклад складання usecase у повній формі для системи продажу товарів у супермаркеті.

Таблиця 4.1 Параметри опису usecase

Usecase section	Comment
Use Case Name	Назва usecase (починається з дієслова)
Scope	System or Business
Level	User-goal or Sub-function
Primary Actor	Головний актор
Stakeholders and interests	Перелік осіб, які зацікавлені в виконанні даного usecase та мета, яку вони при цьому переслідують
Preconditions	Список передумов, які повинні виконуватись для початку виконання usecase
Success guarantee	Список умов, при виконанні яких можна говорити про успішне закінчення виконання usecase
Main Success Scenario	Головний успішний сценарій usecase. Найчастіше є безумовним
Extensions	Альтернативні сценарії успішного чи неуспішного закінчення usecase
Special Requirements	Спеціальні (нефункціональні) вимоги
Technology and Data Variations List	Поради для реалізації певних кроків usecase
Frequency of Occurrence	Частота виконання usecase при користуванні системою (у відсотках)
Miscellaneous	Додаткові вимоги чи відкриті питання

Приклад 3. Купівля товарів у супермаркеті.
Scope: система продажу товарів в супермаркеті (System)
Level: User-goal
Primary Actor: касир
Stakeholders and interests:
1. Касир: зацікавлений у точному швидкому вводі інформації про товари та

відсутності помилок вводу, які призведуть до штрафу (зменшення зарплатні касира)

2. Покупець: зацікавлений у швидкому придбанні товарів (отриманні послуг) та зручному відображенні розрахунків суми покупки.

3. Компанія (продавець): зацікавлена в точній обробці транзакцій при покупці товарів та задоволенні інтересів користувача.

4. Менеджер: зацікавлений у швидкому розв'язанні проблем під час повернення товарів та легкій перевірці операцій, що здійснюють касири.

5. Державна податкова адміністрація: зацікавлена в отриманні податків від кожного продажу товарів.

Preconditions: касир виконав вхід до системи (авторизація)

Main Success Scenario:

1. Покупець підходить до каси з сукупністю товарів (послуг), які він хоче придбати.

2. Касир стартує нову продаж.

3. Касир вводить послідовно всі товари.

4. Система опрацьовує код кожного товару та виводить його назву, кількість та суму покупки для кожного товару.

5. Система розраховує загальну суму покупки та суму нарахованих податків.

6. Касир озвучує суму покупцеві та просить розрахуватися.

7. Покупець розраховується і система опрацьовує оплату.

8. Система запам'ятовує суму покупки до бази даних.

9. Система друкує чек.

10. Користувач відходить від каси з чеком і товарами.

Extensions:

a) у будь-який час менеджер виконує специфічну операцію:

1) менеджер вводить свій код авторизації;

2) менеджер виконує специфічну операцію (наприклад, перевіряє баланс даної каси);

3) менеджер виходить з системи, і вона повертається до режиму роботи з касиром.

b) у будь-який час виникає фатальна помилка в системі:

1) касир перевантажує систему, входить до неї та вводить запит про повернення до попереднього стану;

2) система відновлює попередній стан.

2a) система не може відновити попередній стан:

− система видає помилку на екран та зберігає її у лог;

− касир починає новий продаж.

1a) покупець чи менеджер хоче повернутися до відкладеного продажу:

1) касир виконує операцію повернення та вводить код продажу;

2) система відображає відтворений продаж.

2a) продаж не було знайдено:

− система виводить помилку на екран;

− касир розпочинає новий продаж і вводить дані про всі товари з початку.

2-4a) покупець говорить касиру, що у нього є картка для отримання знижки:

1) касир вводить номер картки до системи;

2) система запам’ятовує код картки та вид знижки для розрахунку загальної суми продажу.

3a) код товару не знайдено в базі даних:

1) система виводить помилку на екран;

2) касир опрацьовує помилку:

2a) додатковий код товару знаходиться на ньому і касир може його прочитати:

− касир вводить код вручну;

− система опрацьовує код та виводить назву товару та ціну.

2b) касир виконує usecase «Знайти код невідомого товару» для ідентифікації товару.

3b) у покупця є декілька однакових товарів:

1) касир може ввести код товару та його кількість вручну.

3-6a) користувач просить видалити один з товарів із поточного продажу:

1) касир вводить номер товару для видалення;

2) система видаляє строку товару з продажу і перераховує загальну суму.

3-6b) покупець просить касира відмінити продаж у цілому:

1) касир відміняє продаж.

3-6c) касир відкладає продаж:

1) система запам’ятовує продаж, яка може бути відновлена в будь-який час;

2) система видає спеціальний чек із переліком товарів та кодом продажу, який дозволить продовжити продаж.

4b) Покупець вважає, що товар є зіпсованим і пропонує купити його за нижчою ціною:

1) касир робить запит у менеджера;

2) менеджер дозволяє виконати продаж за нижчою ціною;

3) касир вводить вручну ціну, що є нижчою за попередню;

4) система відображає нову ціну для даного товару в поточному продажі.

5a) користувач говорить, що він повинен отримати знижку:

1) касир робить запит щодо знижки;

2) касир вводить ідентифікаційний код покупця;

3) система розраховує сумарну знижку на продаж.

6a) покупець говорить касиру, що він хотів розплатитися готівкою, але не має достатньо грошей:

1) касир просить обрати інший метод сплати за товари:

1a) покупець просить відмінити продаж.

7a) оплата готівкою:

1) касир вводить суму грошей, яку дав покупець;

2) система розраховує суму решти і відкриває касу;

3) касир видає решту покупцю;

4) система вносить випадок оплати до бази даних.

7b) оплата кредитною карткою:

1) покупець вводить інформацію про кредитну картку;

2) система показує суму оплати;

3) касир підтверджує суму оплати;

4) система відсилає авторизаційний запит до зовнішньої Системи Оплати Товарів та послуг та робить запит на підтвердження оплати;

5) система отримує підтвердження оплати, виводить інформацію для касира та відкриває касу;

6) система зберігає даний вид оплати за товари у базі даних;

7) касир просить покупця зробити підпис на чеку для підтвердження оплати. Покупець ставить підпис;

8) касир складає чек до каси та замикає її.

7c) касир робить відміну оплати:

1) система повертається до режиму вводу товарів.

9a) покупець просить видати подарунковий чек (без суми продажу):

1) касир робить запит про подарунковий чек. Система друкує подарунковий чек.

9b) в принтері закінчився папір:

1) система дає сигнал про закінчення паперу;

2) касир замінює папір;

3) касир робить запит про повторення друку чеку.

Special Requirements:

1) великий плоский монітор, що має функції сенсорного екрану. Текст повинно бути гарно видно з відстані в 1 м;

2) система авторизації запитів про оплату кредитною карткою повинна видавати результат за 30 секунд у 90 відсотків випадків;

3) система перекладу тексту на декілька мов;

4) додаткові бізнес-правила можуть бути додані до пунктів 3-7.

Technology and Data Variations List:

1) менеджер виконує авторизацію під час скасування певної операції за допомогою зчитування номера картки менеджера кард-рідером чи введенням коду менеджера з клавіатури;

2) код товару вводиться сканером штрих кодів товарів чи з клавіатури;

3) система може використовувати декілька схем кодування товарів (UTC, EAN, JAN, SKU);

4) інформація про кредитну картку вводиться за допомогою кард-рідера чи з клавіатури;

5) підпис покупця для підтвердження кредитної операції робиться на товарному чеці.

Frequency of Occurrence: 95 %.

Miscellaneous (Open Issues):

1) провести аналіз різних варіантів сплати податків;

2) вивчити можливість відновлення роботи системи після збою;

3) яка додаткова функціональність потрібна для різних прикладних галузей?

4) чи повинен касир забирати гроші з каси, коли він виходить із системи (закінчує зміну)?

5) чи може покупець використовувати сканер для ідентифікації товарів під час покупки, чи це повинен робити касир?

КОНТРОЛЬНІ ЗАПИТАННЯ:

Дайте визначення варіанта використання (usecase).

Чим відрізняються варіант використання та сценарій (scenario)?

Які форми опису варіантів використання ви знаєте?

Дайте визначення та наведіть приклад короткої форми опису usecase.

Дайте визначення та наведіть приклад поверхневої форми опису usecase.

Що таке головний успішний сценарій?

Дайте визначення альтернативних сценаріїв? Чи можуть вони бути успішними/неуспішними?

Дайте визначення повної форми опису usecase.

Хто може виступати в ролі актора для варіанта використання?

Хто може виступати в ролі зацікавлених осіб (stakeholders) для варіанта використання?

Яким чином визначаються альтернативні сценарії і як вони пов’язані з головним успішним?

Яким чином визначається параметр frequency of occurrence для повної форми опису варіанта використання?

Комп'ютерний практикум № 5. Побудова діаграм UseCases

Мета роботи: виконати побудову діаграми варіантів використання.

ЗАВДАННЯ

У середовищі Rational Rose створити діаграму варіантів використання для обраного варіанта комп'ютерної системи. Діаграма повинна містити усіх акторів (користувачів системи) та по три варіанти використання для кожного актора. Пов'язати варіанти використання та акторів, при цьому використати усі види зв'язків (unidirectional association, generalization, extend relationship, include relationship).

Теоретичні відомості

Діаграми варіантів використання (usecase diagrams) використовуються для відображення сценаріїв використання системи (usecases) та користувачів системи (actors), які використовують її функції.

Актори на діаграмі варіантів використання позначаються символом людини, а варіанти використання – еліпсом. Актори та варіанти використання поєднуються напрямленою асоціацією (unidirectional association) – стрілкою, що спрямована від актора до варіанта використання. Також актори можуть поєднуватися з використанням зв'язків узагальнення. На рис. 5.1 показано фрагмент діаграми варіантів використання системи аналізу діагностичних ознак електрокардіограми. Актор «Користувач» при цьому може виконати сценарій «Запуск розрахунку».

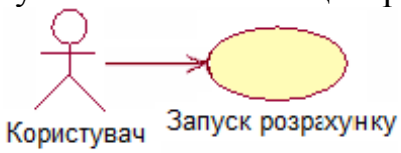


Рис. 5.1. Фрагмент діаграми варіантів використання в Rational Rose

Варіанти використання можуть бути пов'язані між собою трьома видами зв'язків: узагальненням (generalization), розширенням (extend relationship) та включенням (include relationship).

Відношення узагальнення (generalization) показують відношення між загальним і частковим. Також дане відношення може використовуватися для поєднання акторів. Актор «Administrator» може виконувати всі функції актора «Customer», тобто виступає частковим випадком покупця, але може виконувати і специфічні операції (варіант використання «Check db info»).

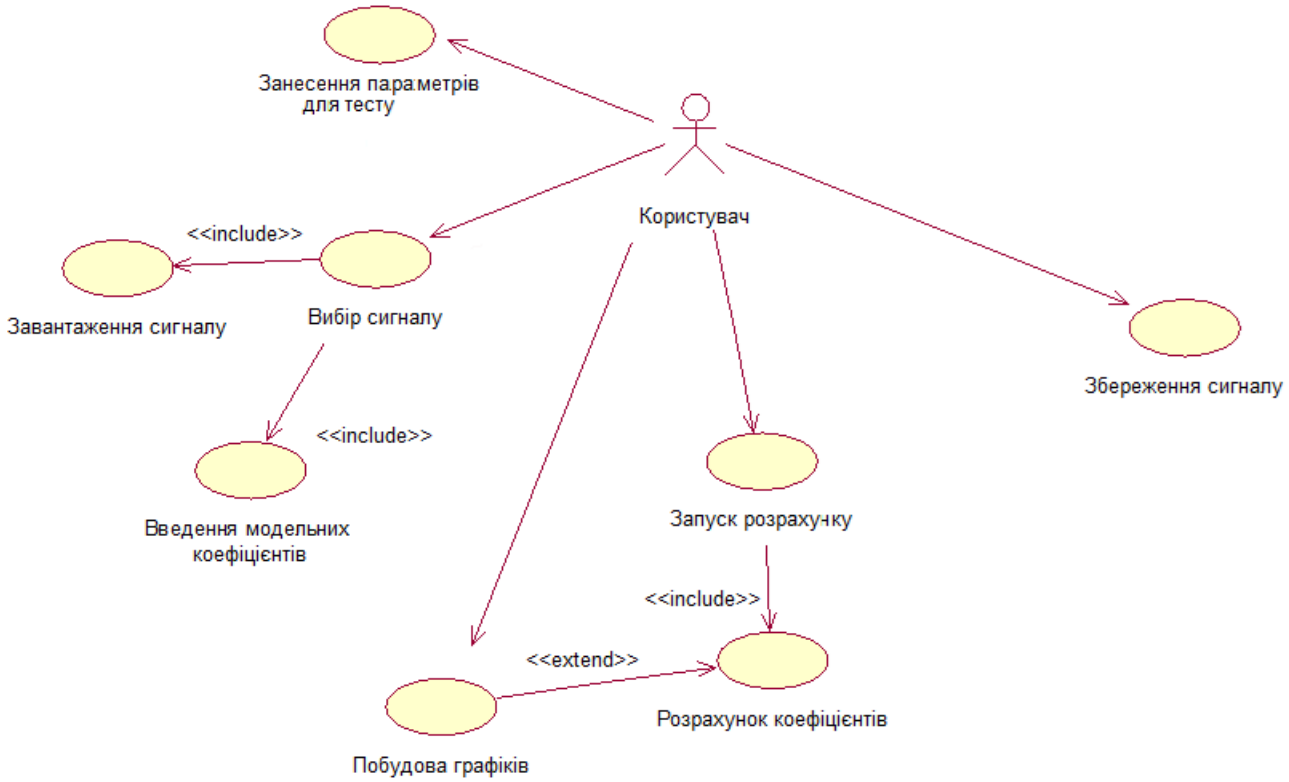


Рис. 5.2. Діаграма варіантів використання системи аналізу діагностичних ознак електрокардіограми

Відношення включення (include) відображає зв’язок «ціле – частина», тобто один варіант завжди в певний момент виконання повністю включає інший. Для прикладу частиною варіанта використання «Вибір сигналу» є сценарії «Завантаження сигналу» та «Введення модельних коефіцієнтів», оскільки для того, щоб вибрати сигнал користувач завжди має завантажити його та обрати ввести коефіцієнти.

Відношення розширення (extend) визначає такий тип відношення, коли один варіант за певних умов повністю використовує інший (розширює його). Так, наприклад, користувач може побудувати графік («Побудова графіків»), знаючи коефіцієнти, які були визначені.

КОНТРОЛЬНІ ЗАПИТАННЯ:

Для чого призначена діаграма варіантів використання (usecase diagram)?

Дайте визначення актора (actor). Хто може виступати в ролі акторів?

Дайте визначення варіанта використання (usecase). Яким чином варіант використання визначається на діаграмі?

Перелічіть відношення, що можуть використовуватися на діаграмі.

Дайте визначення відношення узагальнення (generalization). Наведіть приклад.

У чому відмінність між відношенням узагальнення між акторами та між варіантами використання?

Дайте визначення відношення асоціації (association). Наведіть приклад.

Дайте визначення відношення включення (include). Наведіть приклад.

Дайте визначення відношення розширення (extend). Наведіть приклад.

Які відношення можуть використовуватися для поєднання тільки акторів, акторів та варіантів використання, тільки варіантів використання?

Комп'ютерний практикум № 6. Побудова діаграм взаємодії (Interaction Diagrams)

Мета роботи: виконати побудову діаграми взаємодії.

ЗАВДАННЯ

Для кожного варіанта використання на Usecase Diagram створити Sequence або Collaboration Diagram (тобто у проекті повинно бути не менше шести діаграм кооперації та послідовності). На кожній діаграмі взаємодії повинен бути головний актор (при наявності) та не менше 5 об'єктів. Кожна діаграма взаємодії повинна містити не менше 10 повідомлень, якими обмінюються об'єкти в процесі виконання сценарію. Загальна сума різних об'єктів у проекті повинна налічувати 12-15 об'єктів. Об'єкти та повідомлення на діаграмах повинні мати зрозумілі назви. При побудові діаграм використовувати прямі, рефлексивні та зворотні типи повідомлень, а також символи знищення об'єктів.

Теоретичні відомості

У Rational Rose діаграми взаємодії (Interaction Diagrams) відображають взаємодію логічних елементів системи між собою у процесі виконання актором певного варіанта використання. Розрізняють два типи діаграм взаємодії: діаграми послідовності (Sequence Diagrams) та кооперації (Collaboration Diagrams), які є різними представленнями одного і того ж процесу.

Головними елементами діаграм послідовності є об'єкти, які є логічними сутностями, що представляють окремі елементи системи та повідомлення, якими вони обмінюються. В якості об'єктів можуть виступати також актори. Повідомлення можуть бути не тільки абстрактними діями, що виконуються, але і методи класів, створених на діаграмі класів. Повідомлення на діаграмі послідовності пронумеровані, тобто мають чітку послідовність.

Для прикладу представлено систему електронного документообігу на підприємстві. Для побудови діаграми послідовності обрано варіант використання «Побудова графіків», який виконує актор «Користувач» (рис. 6.1).

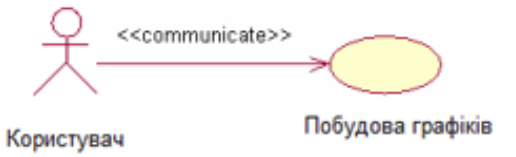


Рис. 6.1. Фрагмент діаграми варіантів використання для системи аналізу діагностичних ознак електрокардіограми

Опис головного успішного сценарію даного варіанта використання у короткій формі:

Користувач має на екрані форму введення коефіцієнтів чи завантаження сигналу, після його дій алгоритм опрацьовує вхідний сигнал, розраховує коефіцієнти, виводить їх на екран та будує зображення графіків.

На рис. 6.2 наведено діаграму послідовності, створену в середовищі Rational Rose для даного прикладу. У верхній частині діаграми наведено перелік об'єктів (логічні сутності), які взаємодіють між собою у процесі виконання сценарію. Часова шкала на даній діаграмі направлена згори донизу, крім того повідомлення

пронумеровані відповідно до черги їх пересилання між об’єктами. Нижче наведено короткий опис подій, що відбуваються при виконанні даного варіанта використання.

При запуску інтерфейсу (1) користувач (2) вибирає тип сигналу (3), інтерфейс передає програмі запит на сигнал (4), алгоритм розраховує коефіцієнти (5) та передає їх інтерфейсу (6), інтерфейс відображає коефіцієнти (7), зображує графіки (8) та передає управління користувачеві (9) (рис. 6.2).

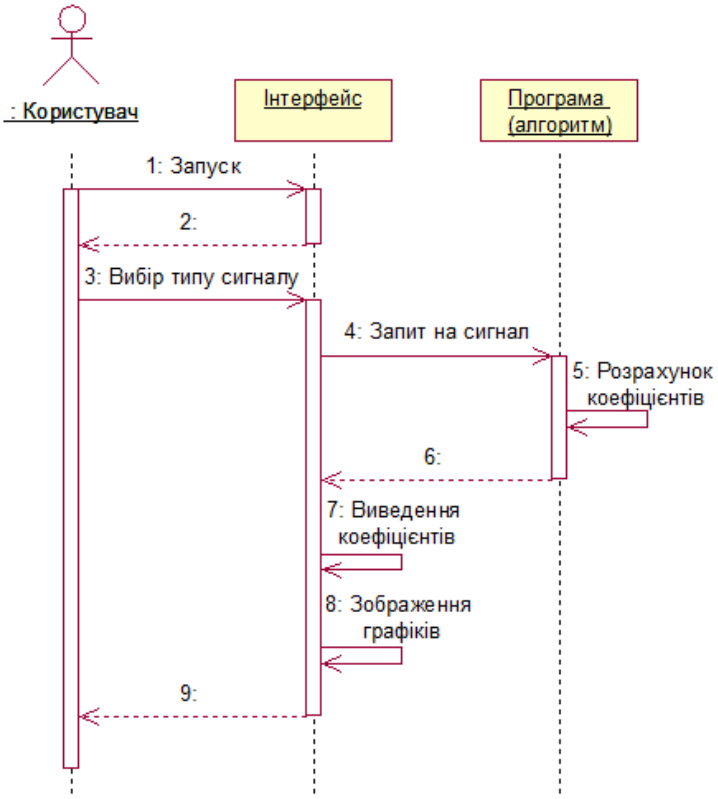


Рис. 6.2. Діаграма послідовності для варіанта використання «Побудова графіків»

Додатковими елементами діаграми послідовності є лінії життя об’єктів (довгі вертикальні пунктирні лінії), які відображають час життя об’єкта від його створення до знищення; фокус керування (прямокутники на лініях життя об’єктів) відображає, який об’єкт виконує операції в певний момент часу; та символи знищення об’єктів (хрест на лінії життя об’єкта), що відображає процес знищення об’єкта.

Основні типи повідомлень на діаграмі послідовності:

− пряме – повідомлення, яке об’єкт-ініціатор надсилає об’єкту-приймачу (1, 3-4);

− рефлексивне, яке об’єкт надсилає сам собі (5, 7-8);

− зворотнє, при якому управління повертається об’єкту ініціатору (2, 6, 9), часто повідомлення даного типу не мають назви.

Діаграма кооперації має те ж саме призначення, що і діаграма послідовності, але відображає процес виконання варіанта використання в іншій нотації (рис. 6.3). Її головними компонентами є об’єкти, назви повідомлень та їх напрям. Характеристика даних компонентів була надана у процесі опису діаграми послідовності.

Примітка: діаграму послідовності можна автоматично перетворити в діаграму кооперації (або навпаки), натиснувши клавішу F5.

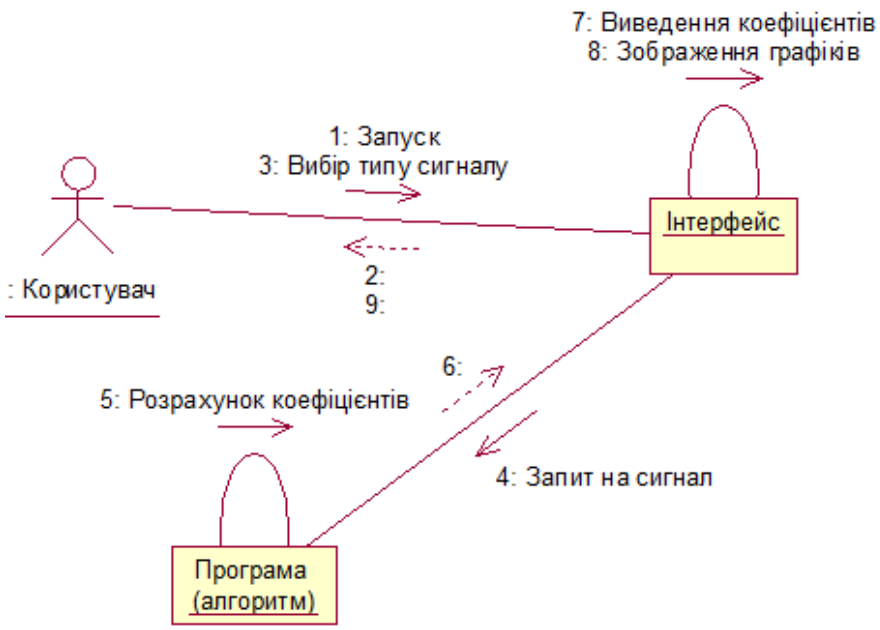


Рис. 6.3. Діаграма кооперації для варіанта використання «Побудова графіків»

КОНТРОЛЬНІ ЗАПИТАННЯ

Які види діаграм взаємодії (interaction diagram) ви знаєте?

Для чого призначена діаграма послідовності (sequence diagram)? Наведіть її основні елементи.

Дайте визначення об’єктів діаграми послідовності, наведіть їх основні характеристики.

Дайте визначення повідомлень діаграми послідовності, перелічіть основні типи повідомлень.

Для чого використовуються прямі, зворотні та рефлексивні повідомлення?

Яким чином на діаграмі послідовності використовуються актори?

Що таке лінія життя об’єкта? Чим визначається початковий і кінцевий момент періоду життя кожного об’єкта?

Для чого призначена діаграма кооперації (collaboration diagram)? Наведіть її основні елементи.

В яких випадках зручніше використовувати діаграму послідовності?

В яких випадках зручніше використовувати діаграму кооперації?

Як діаграму послідовності можна перетворити в діаграму кооперації?

Комп'ютерний практикум № 7. Побудова діаграм класів (Class Diagrams)

Мета роботи: виконати побудову діаграми класів.

ЗАВДАННЯ

1. Для всіх об'єктів на діаграмах взаємодії призначити (створити) певний клас; для кожного повідомлення призначити (створити) відповідний метод (операцію) для класу об'єкта-приймача.
2. Розташувати створені класи з переліком операцій на діаграмі класів.
3. Для кожної операції визначити атрибути, які вона використовує та при необхідності додати їх до списку атрибутів класу.
4. Для кожного атрибуту задати логічний тип даних, для кожної операції логічний тип даних для return value та для переліку аргументів, якщо вони присутні.
5. Пов'язати класи на діаграмі класів, використовуючи різні типи відношень (асоціацію, агрегацію, композицію, наслідування, інстанціювання).

Вимоги

1. Діаграма класів повинна містити не менше 10 класів.
2. Для кожного класу визначити не менше 5 атрибутів та 5 операцій.
3. По можливості використати всі типи відношень між класами.

Теоретичні відомості

Діаграми класів (Class diagrams) – головний тип діаграм UML, які відображають логічну структуру програмної системи та суттєво впливають на процес генерації програмного коду. Основними елементами діаграми класів у Rational Rose є безпосередньо класи (classes) та відношення між ними (relations).

Клас – сукупність логічних об'єктів, які мають схожі характеристики і відрізняються однаковою поведінкою. В ООП характеристики об'єктів певного класу представлені сукупністю атрибутів, а поведінка – сукупністю операцій.

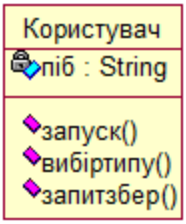


Рис. 7.1. Графічне представлення класу на діаграмі класів у Rational Rose

У Rational Rose кожен клас графічно представлений прямокутником, що має три секції: ім'я класу, перелік атрибутів та перелік операцій (рис. 7.1). Для кожного атрибуту задається тип даних, для кожної операції тип даних для значення, що повертається, та перелік параметрів. Атрибути та операції можуть бути визначені для кожного об'єкта класу, чи для класу в цілому (static attributes and operations). Також для всіх атрибутів та операцій можна визначити тип видимості (public, protected, private).

Розрізняють декілька типів класів:

1. Конкретний клас – для даного класу можна створити об'єкт.
2. Абстрактний клас – клас, для якого не можна створити об'єкт.

Даний клас виступає чистою абстракцією, від нього можна наслідувати конкретні класи.

3. Параметризований клас – клас, для визначення якого використовується список формальних параметрів, які впливають на атрибути та операції (аналог шаблона в C++). Найчастіше такі класи використовуються для створення абстрактних типів даних (списки, вектори, стеки, черги і т. п.).

4. Інтерфейс – клас, що містить тільки визначення набору операцій (без реалізації). У мові C++ аналогом даного класу є клас, всі операції якого є чистими віртуальними функціями. В мові Java класи можуть реалізовувати декілька інтерфейсів (інтерфейси використовуються для реалізації концепції множинного наслідування).

Створення нових класів та операцій класів:

У Rational Rose для кожного об’єкта на діаграмах взаємодії існує можливість призначити певний клас (чи створити новий). При цьому в прямокутнику об’єкта дані відображаються в наступному форматі <ім’я об’єкта>:<ім’я класу>. Також кожне повідомлення на діаграмі взаємодії представляє певну операцію класу-приймача. Для кожного повідомлення можна обрати існуючий метод класу-приймача чи створити новий. На рис. 7.2 наведено фрагмент модифікованої діаграми послідовності для варіанта використання «Побудова графіків».

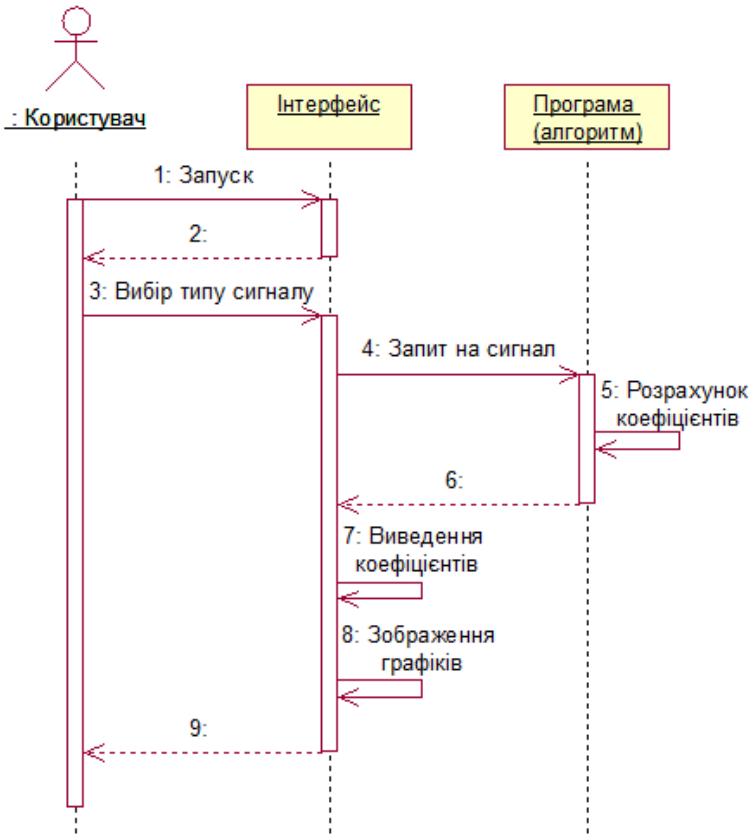


Рис. 7.2. Фрагмент діаграми послідовності для варіанта використання «Побудова графіків»

Після створення набору класів з операціями можна відобразити їх на діаграмі класів. При цьому кожен клас буде мати ім’я та перелік операцій, визначених користувачем. На рис. 7.3 показано відображення класу Алгоритм з операціями, створеними для повідомлення 2 попередньої діаграми.

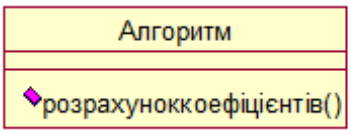


Рис. 7.3. Відображення класу Алгоритм на діаграмі класів

Також нові класи та операції класів можна створювати безпосередньо з діаграми послідовності.

Додавання атрибутів класів:

Після створення переліку класів із набором операцій необхідно для кожного класу створити набір атрибутів – даних, якими оперує програма у процесі виконання операцій класу. Наприклад, виходячи з діаграми послідовності, зображеної на рис. 7.2 можна визначити для класу Алгоритм атрибути для коефіцієнтів, які визначаються при роботі алгоритму та виводяться на інтерфейсі користувача.

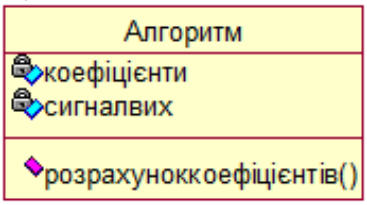


Рис. 7.4. Клас Алгоритм з повним набором атрибутів та операцій

Повний формат відображення атрибутів на діаграмі класів:

<attribute_name>:<attribute_type>

Повний формат відображення операцій на діаграмі класів:

<operation_name>(<list_of_arguments>):<type_of_return_value>

Відношення між класами:

Rational Rose для нотації Unified можна задати 5 основних типів відношень між класами:

1. Відношення асоціації (association) – визначає абстрактний зв’язок між класами, може представляти аналог відношення «m до n».

Приклад: відношення між класами Алгоритм та Вихідний сигнал (лише один даний алгоритм може генерувати від одного до декількох вихідних сигналів).

2. Відношення агрегації (aggregation) – відношення типу «частина – ціле», при якому час життя класа частини не співпадає з часом життя класа цілого.

Приклад: відношення між класами Алгоритм та Коефіцієнти (не обов’язково під час роботи будуть отримані коефіцієнти).

3. Відношення композиції (composition) – відношення типу «частина – ціле», при якому час життя класа частини співпадає з часом життя класа цілого.

Приклад: відношення між класами Алгоритм та ПП (програмний продукт у будь-який момент часу характеризується алгоритмом, який він опрацьовує).

4. Відношення наслідування (generalization) – відношення типу «загальне – часткове».

5. Відношення інстанціювання (instantiation) – визначає інстанціювання нового класу з параметризованого класу шляхом підставлення фактичних параметрів у формальні параметри шаблона.

На рис. 7.5 наведено приклад діаграми класів для системи аналізу діагностичних ознак електрокардіограми, яка містить перелік класів, пов’язаних

різними типами відношень.

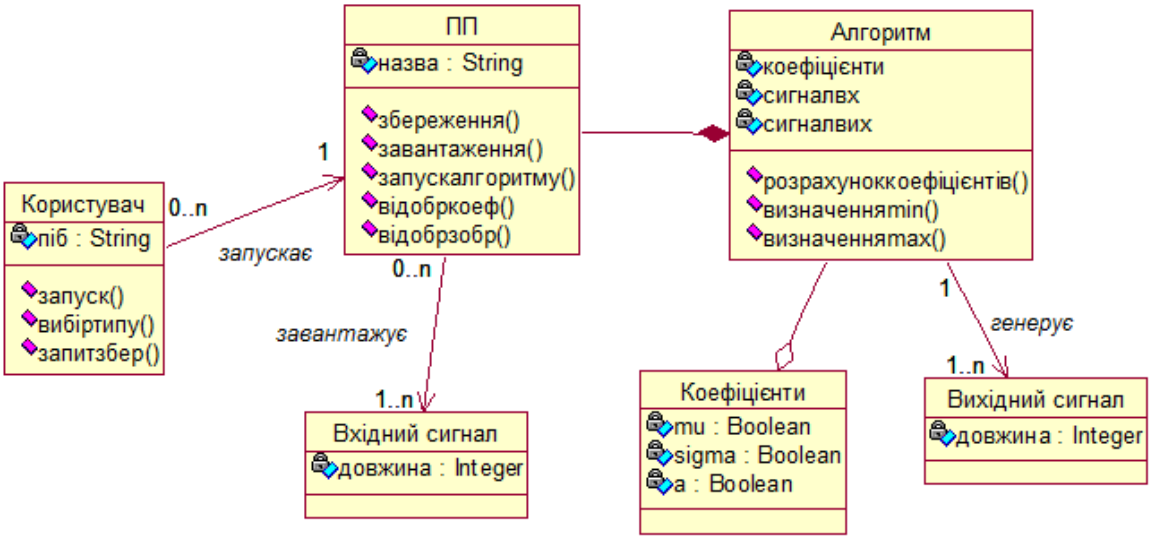


Рис. 7.5. Діаграма класів для системи аналізу діагностичних ознак електрокардіаграми

КОНТРОЛЬНІ ЗАПИТАННЯ:

Для чого призначена діаграма класів (class diagram)? Назвіть її основні елементи.

Дайте визначення класу. Які типи класів ви знаєте?

Чим відрізняється абстрактний клас від конкретного?

Чим відрізняється абстрактний клас від інтерфейсу?

Що таке параметризований клас? Дайте визначення процесу інстанціювання. Наведіть приклад.

Назвіть основні характеристики класу.

Що таке атрибути класу? Які типи атрибутів ви знаєте?

Що таке операції класу? Які типи операцій ви знаєте?

Дайте визначення статичних атрибутів та операцій класу.

Дайте визначення наслідування. Назвіть основні типи наслідування.

Яким чином реалізується інкапсуляція для атрибутів та операцій класу? Назвіть типи модифікаторів доступу до елементів класу.

Яким чином реалізується поліморфізм для класів, що знаходяться в єдиній ієрархії? Наведіть приклад.

Що таке віртуальні функції, для чого вони призначені?

Яким чином можна створити класи та їх операції з діаграм взаємодії?

В якому форматі відображають атрибути та операції на діаграмі класів?

Перелічіть можливі відношення між класами на діаграмі класів.

Які характеристики можна визначити для відношення асоціації?

Як відношення асоціації можна перетворити в агрегацію/композицію?

У чому відмінність між відношеннями агрегації та композиції?

Який клас є базовим, а який похідним при визначенні відношення наслідування?

Які типи класів поєднує відношення інстанціювання?

Комп'ютерний практикум № 8. Побудова діаграми станів та переходів (Statechart Diagrams)

Мета роботи: виконати побудову діаграми станів та переходів (statechart diagrams).

ЗАВДАННЯ

Створити одну діаграму станів для опису процесу функціонування обраної системи в цілому і дві діаграми для конкретних елементів системи. Використовувати діаграму станів для авторизації користувачів забороняється.

Вимоги

1. Кожна діаграма повинна містити не менше 6 станів.
2. По можливості використати обидва типи переходів (звичайний і рефлексивний).
3. Для кожного переходу визначити хоча б одну з характеристик (тригер, гранична умова, дія).

Теоретичні відомості

Діаграми станів та переходів (statechart diagrams) разом із діаграмами діяльності та взаємодії, відображають певний сценарій, що виконується у процесі функціонування системи в цілому, або певної її частини.

Діаграма станів відображає скінчений автомат у вигляді графу, вершинами якого є стани об'єкта, поведінка якого моделюється, а переходами – події, які переводять об'єкт, який розглядається, з одного стану в інший. При цьому вважається, що час перебування об'єкта в певному стані набагато більший за час, необхідний для переходу з одного стану в інший, тобто переходи між станами здійснюються миттєво.

Стан (state) – це логічна сутність, що використовується для моделювання певної ситуації, дії, процесу. Кожен стан має ім'я та список внутрішніх дій. В якості імені стану найчастіше використовуються дієслова, наприклад: «Введення паролю», «Очікування», «Перевірка параметрів». Список внутрішніх дій містить перелік дій, які виконуються у процесі знаходження системи чи об'єкта в даному стані. Кожна дія відображається у форматі:

<період виконання>/<назва дії>,

де поле <період виконання> може набувати наступних значень:

OnEntry – дія виконується під час того, як система входить у даний стан;

OnExit – дія виконується при виході з даного стану;

Do – дія виконується під час знаходження в даному стані;

OnEvent – дія виконується при настанні певної (зовнішньої) події.

Графічне представлення стану «Авторизація» з трьома внутрішніми діями наведено на рис. 8.1.

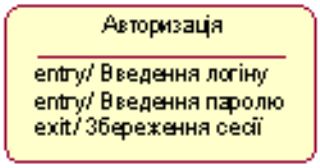


Рис. 8.1. Графічне представлення стану в середовищі Rational Rose

Перехід у даний стан ініціюється при введенні користувачем логіну та паролю. При виході із стану відбувається збереження сесії.

У Rational Rose існує можливість створити також один із специфічних станів:

1. Вхідний стан – стан, в якому знаходиться система (об’єкт) у початковий момент часу.

2. Вихідний стан – стан, в якому знаходиться система (об’єкт) в момент закінчення виконання певної послідовності дій.

3. Стан історії – стан, який запам’ятовує дані, що використовувалися при попередньому входженні системи в даний стан.

Рис. 8.2. Представлення початкового та кіцевого станів на statechart diagrams

Переходи (transitions) на statechart diagrams представлені стрілкою, що виходить з попереднього стану і входить у наступний. Кожен перехід має наступну специфікацію:

<тригер>(<параметри>)[<гранична умова>]/<дія>,

де <тригер> – подія, що ініціює можливість переходу; (<параметри>) – параметри події;

[<гранична умова>] – умова, необхідна для здійснення переходу;

<дія> – дія, що виконується у процесі переходу.

На statechart diagram можна задати два типи переходів:

1. Звичайний – перехід з одного стану в інший.

2. Рефлексивний – перехід із даного стану в цей же стан (зображається у вигляді петлі на графі).

Приклади діаграм:

На рис. 8.3 зображено приклад діаграми станів та переходів для конкретного об’єкта (зміни паролю).

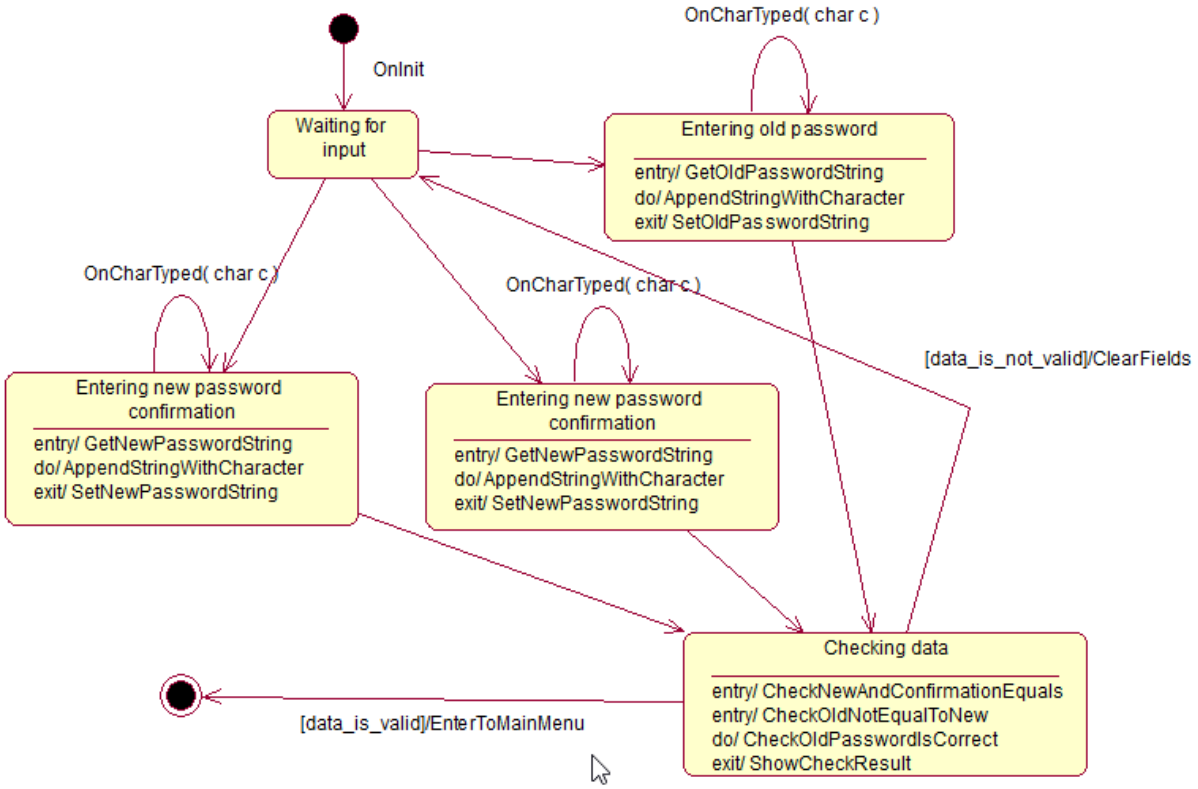


Рис. 8.3. Приклад діаграми станів для зміни паролю

Контрольні запитання

Для чого призначена діаграма станів та переходів (statechart diagram)?
Назвіть основні елементи statechart diagram.
Що таке стан? Перелічіть список тригерів для внутрішніх дій стану.
Перелічіть специфічні стани та наведіть їх призначення.
Для чого призначені переходи на statechart diagram?
Які характеристики властиві для кожного переходу? Чи всі вони обов’язкові?
Які типи переходів існують на діаграмі? Чим вони відрізняються?

Комп'ютерний практикум № 9. Побудова діаграми діяльності (Activity Diagrams)

Мета роботи: виконати побудову діаграми діяльності (activity diagrams).

ЗАВДАННЯ

Побудувати 3 діаграми діяльності для окремих варіантів використання системи.

Вимоги

1. Кожна діаграма повинна містити не менше 6 діяльностей.
2. При побудові кожної діаграми використовувати стани прийняття рішення та синхронізації.

Теоретичні відомості

Діаграми діяльності (activity diagrams) відображають послідовність дій, що виконується в процесі реалізації певного варіанта використання або функціонування системи в цілому. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму. Вони, як і діаграми станів та переходів, відображаються у вигляді орієнтованого графу, вершинами якого є дії, а ребрами – переходи між діями.

Діяльність (activity) є частковим випадком стану (state) без назви, який має одну вхідну подію (OnEntry action). Тому для кожної діяльності назва складається з дієслова та декількох пояснюючих слів, наприклад «Розрахувати заробітну платню» чи «Перевірити результати запиту».

Графічне зображення діяльності «Перекласти слово» подано на рис. 9.1.

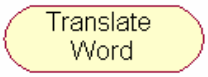


Рис. 9.1. Графічне представлення елементу Activity на діаграмі діяльності

Події (events) на переходах діаграми діяльності не задаються, оскільки вважається, що перехід від однієї дії до іншої здійснюється безумовно.

Гранична умова (guard condition) використовується лише для визначення дії, до якої переходить керування у випадку неоднозначності (рис. 9.2).

Тобто, якщо з даної вершини на діаграмі діяльності можна перейти до декількох інших вершин для всіх переходів необхідно визначити граничну умову. Характеристика дії (action) для переходу також не має сенсу, оскільки всі дії на цій діаграмі представлені вершинами графу.

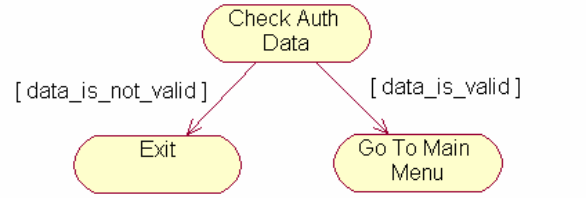


Рис. 9.2. Фрагмент діаграми діяльності для процесу авторизації

Для діаграми діяльності характерними є наступні спеціальні стани:

1. Початковий стан – аналогічний до діаграми станів та переходів.
2. Кінцевий стан – аналогічний до діаграми станів та переходів.
3. Стан прийняття рішення – стан, в якому здійснюється прийняття рішення про перенаправлення потоку управління до одного зі станів, пов'язаних із даним станом.

4. Стан синхронізації – стан, в якому здійснюється розділення загального потоку управління на декілька гілок (чи навпаки, декілька гілок поєднуються в єдиний потік).

Спеціальні стани прийняття рішення та синхронізації представлені на рис. 9.3а та 9.3б відповідно.

Рис. 9.3. Графічне представлення спеціальних станів діаграми діяльності

Розглянемо застосування спеціальних станів на конкретному прикладі (варіант використання «Перекласти слово» для електронного словника).

На рис. 9.4 показано повний вигляд діаграми діяльності.

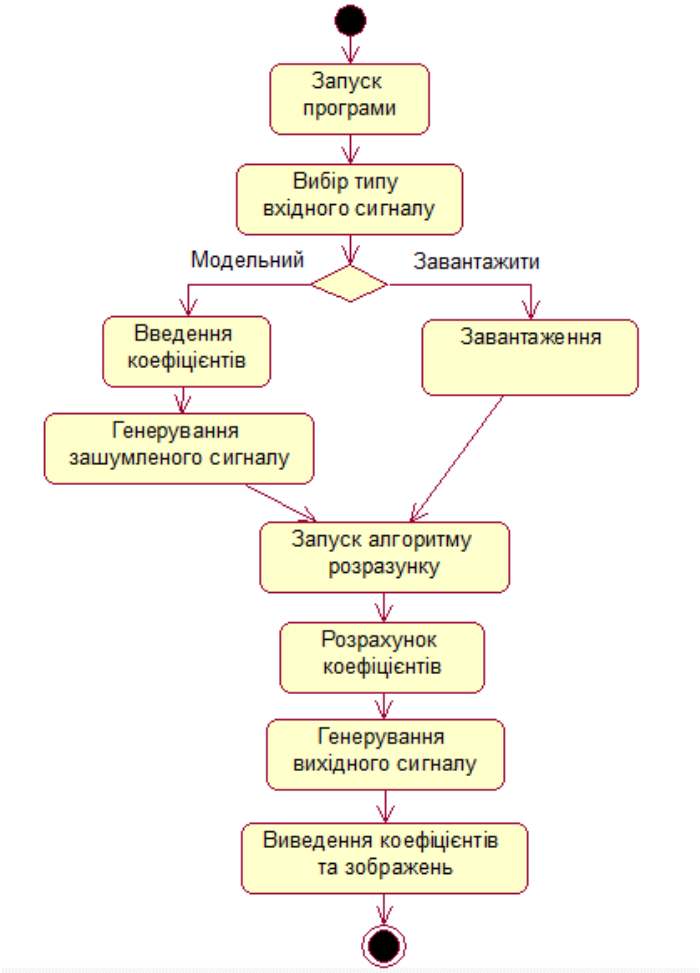


Рис. 9.4. Діаграма діяльності роботи програмного продукту

Після запуску програми користувач обирає використовувати вхідним змодельований чи завантажений сигнал. У разі використання модельного користувач має ввести коефіцієнти, після чого сигнал генерується програмою. Якщо ж вибрали завантажений, то програма завантажує сигнал. Після закінчення виконання операцій гілки поєднуються в один потік, в якому програма запускає алгоритм розрахунку, алгоритм розраховує коефіцієнти, генерує вихідний сигнал. Програма виводить коефіцієнти та зображення.

Також діаграми діяльності використовуються для відображення послідовності дій при моделюванні бізнес-процесів. При цьому використовується додатковий

елемент діаграми, який має назву swimlane.

Swimlane в дослівному перекладі означає доріжка плавального басейну (за аналогією з графічним відображенням). В якості swimlanes на діаграмі можуть виступати фізичні особи, групи осіб, відділи підприємства, чи навіть окремі організації. Розглянемо діаграму діяльності зі swimlanes для спрощеного варіанта бізнес-процесу «Розробка програмного забезпечення» (рис. 9.5).

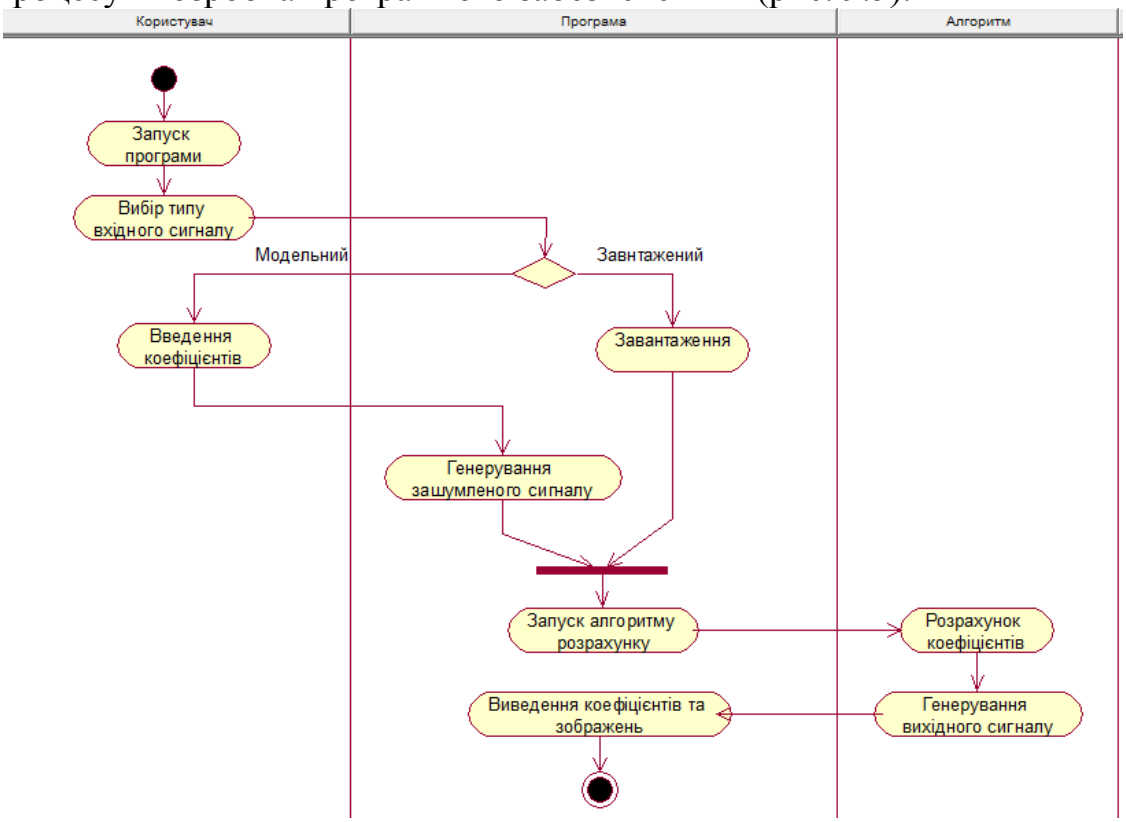


Рис. 9.5. Використання діаграми діяльності для відображення бізнес-процесів

В якості swimlanes в даній діаграмі виступають наступні особи (групи осіб):

1) користувач, який обирає сигнал;

2) програма, яка генерує чи завантажує сигнал та запускає алгоритм розрахунку;

3) алгоритм, який розраховує коефіцієнти та генерує вихідний сигнал.

КОНТРОЛЬНІ ЗАПИТАННЯ:

Для чого призначена діаграма діяльності (activity diagram)? Перелічіть основні елементи.

У чому відмінність між діаграмою діяльності і діаграмою станів та переходів?

Що таке діяльність? Чим діяльність відрізняється від стану?

Що таке стан прийняття рішення?

Що таке стан синхронізації?

Для чого використовується елемент swimlane?

Комп'ютерний практикум № 10. Побудова діаграм компонентів (Component Diagrams)

Мета роботи: виконати побудову діаграм компонентів (component diagrams).

ЗАВДАННЯ

Побудувати діаграму компонентів для обраної програмної системи. Діаграма повинна містити не менше трьох компонентів. Розподілити всі класи між компонентами.

Теоретичні відомості

Діаграми компонентів (component diagrams) відображають фізичне представлення програмної системи у вигляді сукупності елементів, які називають компонентами (components). Кожен компонент має ім'я, мову реалізації та перелік призначених класів. Фізично кожен компонент може бути представлений у вигляді окремого файлу, директорії чи програмного продукту. Графічний вигляд компоненту наведено на рис. 10.1.

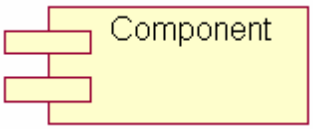


Рис. 10.1. Графічне представлення компоненту

У залежності від обраного стереотипу компонент може належати до одного з наступних типів: Applet, Application, Database, DLL, EXE, MainProgram, SubProgram. Тип компонента є частиною його графічного представлення і не впливає на генерацію програмного коду.

Одним з елементів мови UML є також пакет (package). Головною відмінністю компонентів від пакетів є те, що пакет є логічним об'єднанням класів (аналог namespace в C++) за їх функціональністю, а компонент є окремим фізичним елементом системи, і може виступати в якості прикладної програми, підсистеми, бібліотеки тощо.

Компоненти поєднуються між собою за допомогою відношення залежності (dependency) відповідно до відношень між класами, що належать до цих компонентів.

Розглянемо побудову діаграми компонентів, виходячи з діаграми класів абстрактної системи. На рис. 10.2 відображено діаграму класів програмної системи.

Перший шар складається з сукупності класів, які формують інтерфейс користувача.

Другий шар відповідає за бізнес-логіку (перелік алгоритмів, що оперують даними для виконання функцій системи).

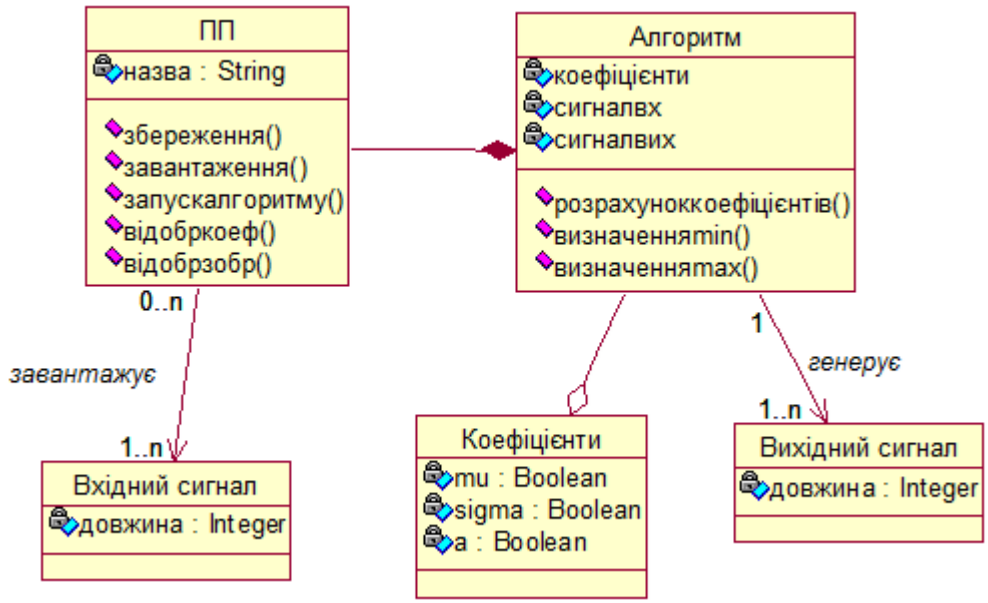


Рис. 10.2. Приклад діаграми класів для програмної системи з тришаровою архітектурою

Третій шар забезпечує обмін даними з базою даних. Він складається зі специфічних структур даних, кожна з яких відповідає рядку відповідної таблиці бази даних, контейнерів структур даних, класу інтерфейсу бази даних та прав доступу до таблиць БД. Програмний продукт є поєднанням цих трьох шарів, кожен з яких може бути представленим окремою бібліотекою, які можуть бути реалізовані різними мовами програмування. На рис. 10.3 представлена діаграма компонентів для даного програмного комплексу.

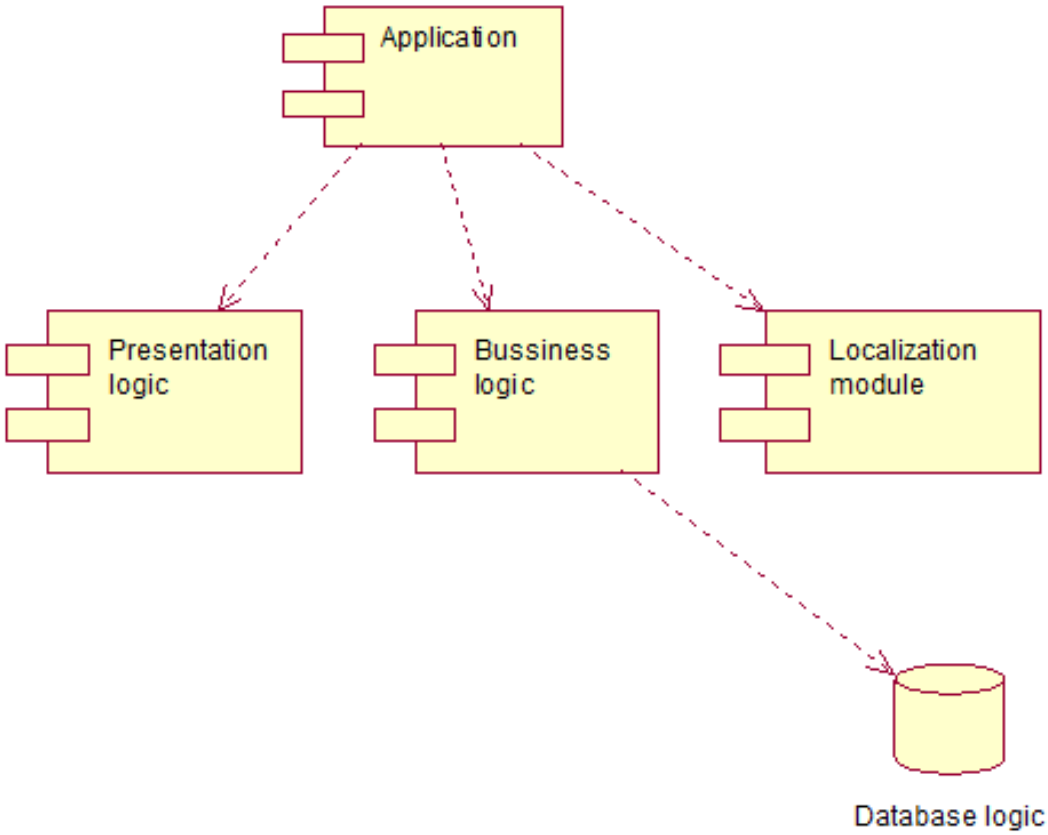


Рис. 10.3. Діаграма компонентів програмного продукту

КОНТРОЛЬНІ ЗАПИТАННЯ:

Для чого призначена діаграма послідовності (sequence diagram)? Наведіть її основні елементи.

Дайте характеристику компонента. Що може виступати в ролі компонента? Наведіть приклад.

Назвіть головну відмінність між пакетами та компонентами.

Що таке 3 tier архітектура? Наведіть приклад системи, для якої можна використати дану архітектуру.

Які характеристики визначаються для кожного компонента на діаграмі?

Яким чином класи розподіляються за компонентами?

Яким чином визначаються відношення між компонентами?

Додаток А. Оформлення звіту за результатами виконання комп'ютерного практикуму

Звіт по виконанню комп'ютерного практикуму оформляється у вигляді звітного документу.

Вимоги до звітного документу:

- усі поля параметрів сторінки – 2 см;
- верхній та нижній колонтитули – 0 см;
- текст матеріалів набирається шрифтом Times New Roman, кегль 12, міжрядковий інтервал 1,0; абзацний відступ – 1 см; вирівнювання тексту за шириною сторінки.

Звіт по комп'ютерному практикуму (Приклад 1) формується на основі протоколу, який ведеться під час виконання поточної роботи та результатів домашньої (теоретичної та практичної) підготовки.

Звіт повинен містити наступні розділи:

1. ***Вступна частина*** (назва: ВУЗу, факультету, кафедри, дисципліни; номером поточної лабораторної роботи та тема; номер групи та П.І.Б. виконавця; П.І.Б. викладача, що перевірятиме роботу);
2. ***Основна частина:***
   – мета роботи;
   – завдання до роботи;
   –результати поставленого завдання у вигляді діаграм та моделей;
3. ***Відповіді на контрольні запитання.***

Приклад 1

Факультет Біомедичної інженерії
Національного технічного університету України «КПІ ім. І. Сікорського»
Кафедра біомедичної кібернетики

Дисципліна «Управління ІТ-проектами»

Комп'ютерний практикум № ___

Тема: _____

Виконав (ла):
студент (ка) групи ___,
(ПІБ)_____

Перевірив:
викладач (ПІБ)_____

дата _____ підпис _____

Мета роботи: _____.
Завдання до роботи: _____.

Результати

_____
_____

Відповіді на контрольні запитання

_____
_____

Література

Базова

1. A Guide to the Project Management Body of Knowledge: Fifth Edition (PMBOK Guide).-2013.-616 p.

2. Хелдман К. Профессиональное управление проектом [Текст] / К. Хелдман. - М.: БИНОМ. Лаборатория знаний, 2005. - 517 с.

3. Методология функционального моделирования IDEF0. М: Изд-во ИПК Издательство стандартов. - 2000. - 75 с. [Електронний ресурс]. - Режим доступу : http://www.staratel.com/iso/IDEF/IDEFO/IDEFORus.pdf

4. М.Н. Пущин Проектирование информационных систем: Учеб. пособие. - М: Изд-во МИЭТ. - 2008. - 234 с. [Електронний ресурс]. - Режим доступу : http://pmn.narod.ru/disciplins/cis/cis.pdf

Допоміжна

5. Леоненков А.В. Визуальное моделирование в среде IBM Rational Rose 2003. Интернет-университет информационных технологий. - ИНТУИТ.ру. - 2005. - [Електронний ресурс]. - Режим доступу : www.intuit.ru.

6. Маклаков СВ. Моделирование бизнес-процессов с AIIFusion Process Modeler. - М.: Диолог-МИФИ. - 2004. [Електронний ресурс]. - Режим доступу : http://www.twirpx.com/fde/164779/

7. Маклаков СВ. BP win и ERwin. CASE-средства разработки информационных систем. - 2-е изд., испр. и дополн. - М.: Диолог-МИФИ. - 2001. [Електронний ресурс]. -Режим доступу : http://www.twirpx.com/fde/22386/

8. INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEFO). Draft Federal Information Processing Standards Publication - 1993. - 183 p. [Електронний ресурс]. - Режим доступу : http://www.standartization.com/IDEF/IDEFO/IDEFO.pdf

9. Управление инновационными проектами [Текст] : уч.пособ.для студ.ВУЗ / под ред. проф. Попова В.А. - М.: ИНФРА-М, 2009. - 336 с.