Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

# ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

## КОМП’ЮТЕРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною Радою КПІ ім.Ігоря Сікорського
як навчальний посібник для студентів,
які навчаються за спеціальністю 122 «Комп’ютерні науки»*

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти: *Козак Л. М.,* д-р біол. наук, провідний наук. співр. Міжн. наук.-навч. центру інформаційних технологій і систем НАН та МОН України

*Антонова-Рафі Ю.В.,* канд. техн. наук, доц. кафедри БЗЛ ФБМІ НТУУ «КПІ ім. Ігоря Сікорського»

Відповідальний редактор *Носовець О. К.*, канд. техн. наук

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського*
*(протокол № 7 від 13.05.2021 р.)*
*за поданням Вченої ради інституту/факультету*
*(протокол № 12 від 29.03.2021 р.)*

*Електронне мережеве навчальне видання*

*Добровська Людмила Миколаївна,* к. пед. н.
*Аверьянова Ольга Анатоліївна*

# ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

## КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Проєктування інформаційних систем: Комп'ютерний практикум [Електронний ресурс]: навчальний посібник для студентів спеціальності 122 "Комп'ютерні науки" для всіх спеціалізацій / Укладачі: Л. М. Добровська, О.В. Аверьянова; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 7,043 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021 – 202 с.

Посібник містить матеріал з навчальної дисципліни «Проєктування інформаційних систем», яка включена у навчальні плани підготовки бакалаврів спеціальності 122 «Комп'ютерні науки» і знайомить студентів із сучасними підходами до проєктування ІС. Науковою основою дисципліни є методології системного аналізу і моделювання, структурний та об'єктно–орієнтований підходи до проєктування програмного забезпечення (ПЗ).

Посібник містить опис процесу проєктування моделей даних в обсязі, необхідному для практичної роботи. Детально на конкретних прикладах розглянуто застосування CASE-засобів для автоматизації етапів аналізу, проєктування і генерації коду інформаційних систем.

Посібник призначений для студентів, які вивчають основи проєктування інформаційних систем.

**З М І С Т**

## СКОРОЧЕННЯ

База даних - БД

Електронна документація – ЕД

Інформаційна система - ІС

Ліва кнопка миші - ЛКМ

Медична інформаційна система – МІС

Міжнародна класифікація хвороб – МКХ

Модель даних - МД

Права кнопка миші - ПКМ

# ВСТУП

В сучасному світі інформація перетворилася в один із найважливіших ресурсів, а інформаційні системи (ІС) стали необхідним інструментом практично в усіх сферах діяльності людини. Тенденції розвитку сучасних інформаційних технологій приводять до постійного зростання складності ІС, що створюються в різних областях діяльності. Для успішної реалізації ІС, об'єкт проєктування повинен бути адекватно описаний, повинні бути побудовані повні та несуперечливі функціональні та інформаційні моделі ІС.

Проєктування ІС - складна, трудомістка і тривала за часом робота, яка вимагає високої кваліфікації фахівців, задіяних в цій роботі. Це сприяє появі програмно-технологічних засобів спеціального класу - CASE-засобів (Computer Aided Software Engineering), які реалізують CASE-технологію створення і супроводу ІС.

CASE-технологія є методологією проєктування ІС, вона має вигляд набору інструментальних засобів, які дозволяють в наочній формі моделювати предметну область, аналізувати отриману модель на всіх етапах розробки і супроводу ІС, розробляти програми відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів засновано на методологіях структурного або об'єктно-орієнтованого аналізу і проєктування, що використовують специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв'язків між моделями системи, динаміки поведінки системи та архітектури програмних засобів.

Навчальний посібник є компіляцією популярних матеріалів, які розкривають практичні аспекти застосування таких CASE-засобів візуального проєктування, як:

а) інструмент візуального моделювання бізнес-процесів AIIFusion Process Modeler (BPwin);

б) засіб моделювання даних і проєктування баз даних AllFusion ERwin Data Modeler;

в) засіб об'єктно-орієнтованого моделювання та розробки проєктів програмних додатків Rational Rose.

У *першому розділі* навчального посібника розглядається інструмент візуального моделювання бізнес-процесів AIIFusion Process Modeler (BPwin). Дано опис методології та інструментальних засобів, а також набір практичних завдань, що дозволяють освоїти техніку створення функціональних моделей. Послідовно розглядаються три нотації моделювання, підтримувані BPwin: IDEF0, IDEF3 і DFD.

Процес побудови моделі розбивається на такі етапи :

1. Виділення множини вимог в основні функціональні групи (процесів).
2. Виявлення зовнішніх об’єктів, зв’язаних з розроблюваною системою.
3. Ідентифікація основних потоків інформації, яка циркулює між системою і зовнішніми об’єктами.
4. Попередня розробка контекстної діаграми. Перевірка попередньої контекстної діаграми і внесення в неї змін.
5. Побудова контекстної діаграми шляхом об’єднання всіх процесів попередньої діаграми в один процес, а також групування потоків.
6. Перевірка основних вимог контекстної діаграми.
7. Декомпозиція кожного процесу поточної DFD з допомогою деталізуючої діаграми чи специфікації процесу.
8. Перевірка основних вимог по DFD відповідного рівня.
9. Додавання визначень нових потоків у словник даних при кожній появі їх на діаграмі.
10. Перевірка повноти і наочності моделі після побудови кожних двох-трьох рівнів.

*Другий розділ* присвячений засобу моделювання даних і проєктування баз даних AllFusion ERwin Data Modeler (ERwin). Описано механізм розробки моделі бази даних на логічному та фізичному рівнях. Розглядається наочна модель бази даних.

Предметом *третього розділу* є розгляд практичних особливостей процесу

об'єктно-орієнтованого моделювання та розробки проєктів програмних додатків з використанням CASE-засобу Rational Rose 2007. Описуються елементи робочого інтерфейсу програми та рекомендації щодо виконання проєкту в нотації UML. Процес розробки моделей в середовищі Rational Rose ілюструється практичними прикладами побудови конкретних діаграм в нотації UML.

Серед основних етапів підготовки та проведення практичних занять можна виділити такі: 1) проведення попереднього контролю підготовленості студентів до виконання конкретної практичної роботи; 2) виконання конкретних завдань у відповідності із запропонованою тематикою, оформлення індивідуального звіту; 3) оцінювання результатів роботи студентів викладачем.

Студент формулює індивідуальне завдання у відповідності з обраною темою бакалаврської роботи чи іншою самостійно обраною темою, яка цікавить студента з практичних або теоретичних міркувань.

Остаточний звіт з роботи містить такі складові структури проєкту:

РОЗДІЛ І. Технічне завдання: 1) загальні відомості (назва системи, найменування організації-замовника та організацій-учасників робіт, передумови для проведення робіт, порядок оформлення та передачі замовнику результатів робіт по розробці програмного забезпечення); 2) призначення системи; 3) характеристика об’єкта автоматизації; 4) вимоги до системи (до системи в цілому, до функціональних характеристик, до надійності, до захисту інформації від несанкціонованого доступу та забезпечення анонімності іспитів, до структури та складу програмного комплексу); вимоги до функцій (задач), що виконуються системою (функції модулів); вимоги до програмного забезпечення; вимоги до технічного забезпечення; вимоги до організаційного забезпечення; вимоги до лінгвістичного забезпечення; вимоги до ергономіки та технічної естетики.

РОЗДІЛ ІІ. Опис проєкту - склад і зміст робіт по створенню системи на етапі проєктування: 1) модель життєвого циклу; 2) контекстна діаграма IDEF0; 3) діаграма декомпозиції IDEF0; 4) методологія IDEF3; 5) методологія DFD;

6)діаграма варіантів використання; 7) діаграма кооперацій; 8) діаграма послідовності; 9) діаграма класів; 10) діаграма станів; 11) діаграма діяльності; 12)діаграма компонентів; 13) діаграма розгортання; 14) алгоритм роботи додатку.

Автори висловлюють вдячність і подяку студентам НТУУ «КПІ ім.Сікорського» кафедри біомедичної кібернетики, які взяли участь в підготовці навчального посібника.

# РОЗДІЛ 1. МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ ЗАСОБАМИ BPWIN

## 1. Постановка завдання та опис предметної області процесу. Середовище проєктування AllFusion Process Modeler 4.1

*Мета та завдання практикуму:* ознайомитись із

1) процедурою опису задачі (опис функціональної моделі);

2) робочим інтерфейсом програми AllFusion Process Modeler 4.1 (BPwin);

3) процедурою побудови діаграми декомпозиції першого рівня.

4) освоїти технологію створення нової моделі, ознайомитись з принципами редагування отриманої діаграми.

План

1. Моделювання предметної області: метод функціонального моделювання SADT (IDEF0) [3 с. 133-147].

**Завдання**

1. Обрати приклад бізнес-процесу та описати його:
- цільові функції підприємства (медичного закладу або системи);
- нормативні документи підприємства (медичного закладу або системи);
- підрозділи підприємства (медичного закладу або системи);

2. Запустити програму та створити нову модель. Обрати тип діаграми *Business Process* (IDEF0).

3. Побудувати контекстну діаграму. Якщо текст блоку відображено некоректно, потрібно вирішити цю проблему.

4. Побудувати дуги керування. Ідентифікувати дуги керування.

5. Встановити «тільди».

6. Змінити колір тексту, фону блоків, кольору та стилю дуг.

7. Навчитись видаляти блоки.

8. Зберегти проєкт.

9. Відповісти на контрольні запитання.

10. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи.**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту.

2. Продемонструвати виконане завдання викладачу.

**Контрольні питання**

1. Визначіть поняття «модель бізнес-процесу».
2. Що входить до складу контекстної діаграми ?
3. Як називається етап, під час якого створюють діаграми найвищого рівня?
4. Для чого призначена ліва сторона блоку?
5. Яка команда із контекстного меню дозволяє змінити стиль дуги?
6. Для чого призначений інструмент **T** ?

## Аудиторна робота

Для спрощення побудови бізнес-процесу IDEF0, можна використати CASE-інструмент **AllFusion Process Modeler 4.1,** який дозволяє проілюструвати функціональну декомпозицію системи. Цей інструмент підтримує стандарт IDEF0, прийнятий на початку 90-х років XX ст. в США на основі методології SADT.

Основна ідея методології SADT - побудова **деревоподібної функціональної моделі** підприємства.

На початку функціональність системи описується загально, без подробиць.

Взаємодія із навколишнім світом описується в термінах **входу** (данні чи об'єкти, що використовуються або змінюються функцією), **виходу** основний результат діяльності функції, кінцевий продукт), **управління** (стратегії та процедури, якими керується функція) і **механізмів** (необхідні ресурси).

При створенні контекстної діаграми формулюють *мету моделювання*, *область* (опис того, що буде розглядатись як компонент системи, а що як зовнішній вплив) і точка зору (позиція, з якої буде будуватись модель). Зазвичай в якості точки зору обирається точка зору суб'єкта, який відповідає за роботу моделі.

Загальна функція розділяється на підфункції. Цей процес називається ***функціональною декомпозицією.*** Кожна підфункція розбивається на більш дрібні і цей процес триває до отримання необхідної деталізації опису. Таким чином, формується діаграма IDEF0.

Процес побудови моделі охоплює такі етапи:

1. Визначення основного бізнес-процесу.
2. Побудова контекстної діаграми.
3. Побудова діаграми процесу найвищого рівня.
4. Функціональна декомпозиція процесу за допомоги функціональних діаграм.

Для прикладу, розглянемо *модель бізнес-процесу (діяльності) амбулаторії сімейної медицини.* Для підтримки її діяльності необхідно розробити медичну інформаційну систему(МІС).

**Приклад 1.1.** Опис бізнес-процесів

**Постановка задачі.** Адміністрація лікарні замовила розробку ІС для відділу «Амбулаторія сімейної медицини». Нова система призначена для обробки даних про лікарів, пацієнтів, прийом пацієнтів і курси лікування. Система повинна видавати звіти за запитом лікарів або адміністрації.

Під час обстеження складено опис діяльності розглянутого підрозділу. Перед прийомом в лікарню проводиться зустріч пацієнта і лікаря. Лікар отримує інформацію про запис на прийом пацієнта та очікуваний прийом хворого і передає про нього дані. Пацієнт може бути прийнятий в амбулаторії більш ніж один раз (якщо пацієнт раніше не лікувався, то йому присвоюють

реєстраційний номер і записують дані: прізвище, ім'я та по батькові пацієнта, адреса і дата народження, мобільний телефон). Пацієнт повинен бути зареєстрований в системі до прийому в амбулаторії.

Через деякий час лікар оформляє прийом хворого (при цьому визначається порядковий номер прийому і запам'ятовуються дані прийому пацієнта). Після цього відділ прийому надсилає повідомлення лікаря для підтвердження прийому хворого. У цей запис включаються реєстраційний номер пацієнта і його прізвище, порядковий номер прийому, дата початку лікування і номер палати.

У день прийому пацієнт повідомляє у відділ прийому про своє прибуття і передає дані про себе (або зміни в даних). Відділ прийому перевіряє і при необхідності коригує дані про пацієнта. Якщо пацієнт не пам'ятає свій реєстраційний номер, то виконується відповідний запит.

На пацієнта заводиться реєстраційна картка, яка містить прізвище, ім'я та по батькові пацієнта, адресу, дату народження, номер телефону, групу крові, назву страхової компанії та номер страховки (якщо вона є).

Під час перебування в амбулаторії курс лікування призначається тільки одним лікарем (у лікарні пацієнт може лікуватися у кількох лікарів, кожен з яких призначає один або більше курсів лікування). Дані про курси лікування передаються в реєстратуру (відповідну БД), яка займається координацією лікування пацієнтів, дані реєструються і зберігаються там. Дані включають номер лікаря, номер пацієнта, порядковий номер прийому, назва курсу лікування, дату призначення, час і примітки. При необхідності лікар запитує в реєстратурі історію хвороби пацієнта, яка містить дані про курси лікування, отриманих пацієнтом. Історія хвороби подається у вигляді табл. 1.1.

Після закінчення курсу лікування лікуючий лікар приймає рішення про виписку пацієнта. Коли пацієнт виписується, він повідомляє про це у відділ прийому (відділ прийому реєструє дані про виписку, включаючи дату виписки, і дає пацієнтові довідку про закінчення перебування в амбулаторії).

Лікар передає дані про себе і зміни в даних у відділ прийому. Дані про лікаря включають код лікаря, прізвище, ім'я та по батькові, адресу, дату народження, домашній телефон, спеціалізацію, номер кабінету, робочий телефон.

Адміністрація лікарні може запросити звіт про пацієнтів. У запиті вказується інтервал часу (початкова та кінцева дати). Звіт повинен бути поданий у вигляді табл. 1.2. Адміністрація лікарні запитує статистичні дані про захворювання за останній тиждень щопонеділка о 9.00.

Таблиця 1.1

*Історія хвороби пацієнта*

| Історія хвороби | | Дата 01-02-1998 | |
|---|---|---|---|
| Дані з 01-01-1992 по 01-02-1998 | | | |
| Код пацієнта<br>732 | ПІП<br>Іванов С.П. | Дата народження<br>15-07-1955 | |
| Номер прийому<br>649 | Дата прийому<br>06-07-1992 | Палата 4 | |
| Захворювання: перелом ноги | | Лікар: Петров В. С., хірург | |
| | Дата | Час | Примітка |
| | 06-07-1992 | 12-00 | Накладено гіпс |
| | 20-07-1992 | 14-00 | Стан хороший |
| | 22-07-1992 | 10-00 | Лікування закінчено |
| Номер прийому<br>749 | Дата прийому<br>10-07-1992 | Палата 3 | |
| Захворювання: ОРЗ | | Лікар: Іванов А.С., терапевт | |
| | Дата | Час | Примітка |
| | 10-07-1992 | 10-00 | Призначено терапію |
| | 15-07-1992 | 10-00 | Стан задовільний |
| | 20-07-1992 | 10-00 | Стан хороший |
| | 25-07-1992 | 10-00 | Виписаний |
| Номер прийому<br>711 | Дата прийому<br>12-04-1993 | Палата 8 | |
| Захворювання: серцевий приступ | | Лікар: Сидорбов І.С., кардіолог | |
| | Дата | Час | Примітка |
| | | | |

Таблиця 1.2

| *Звіт про пацієнтів* | | | | | | |
|---|---|---|---|---|---|---|
| Дані від 01-12-1997 до 01-02-1998 | | | | | | |
| Номер | ПІП | Адреса | Телефон | Дата народження | Страхова компанія | Номер страхового полісу |
| | | | | | | |

### *Опис контексту системи прийому пацієнтів в амбулаторії та побудова початкової контекстної діаграми*

Побудуємо початкову контекстну діаграму потоків даних в нотації Гейне - Сєрсона. Намалюємо нульовий процес і дамо йому ім'я системи (Система прийому пацієнтів). Оскільки моделюється діяльність відділу амбулаторії (прийому пацієнтів), зовнішніми сутностями є Лікар, Пацієнт і Адміністрація лікарні. Намалюємо зовнішні сутності і з'єднаємо їх з нульовим процесом за допомогою потоків даних.

**Специфікація структур даних.** Визначимо склад потоків даних і підготуємо вихідну інформацію для конструювання концептуальної моделі даних. Помітимо символами «*», «°» і «|» все структури і елементи даних типу «ітерація», «умовне входження» і «альтернатива» відповідно. Після об'єднання

структур і елементів даних в більш великі структури для кожного потоку даних повинна бути Сформована структура даних.

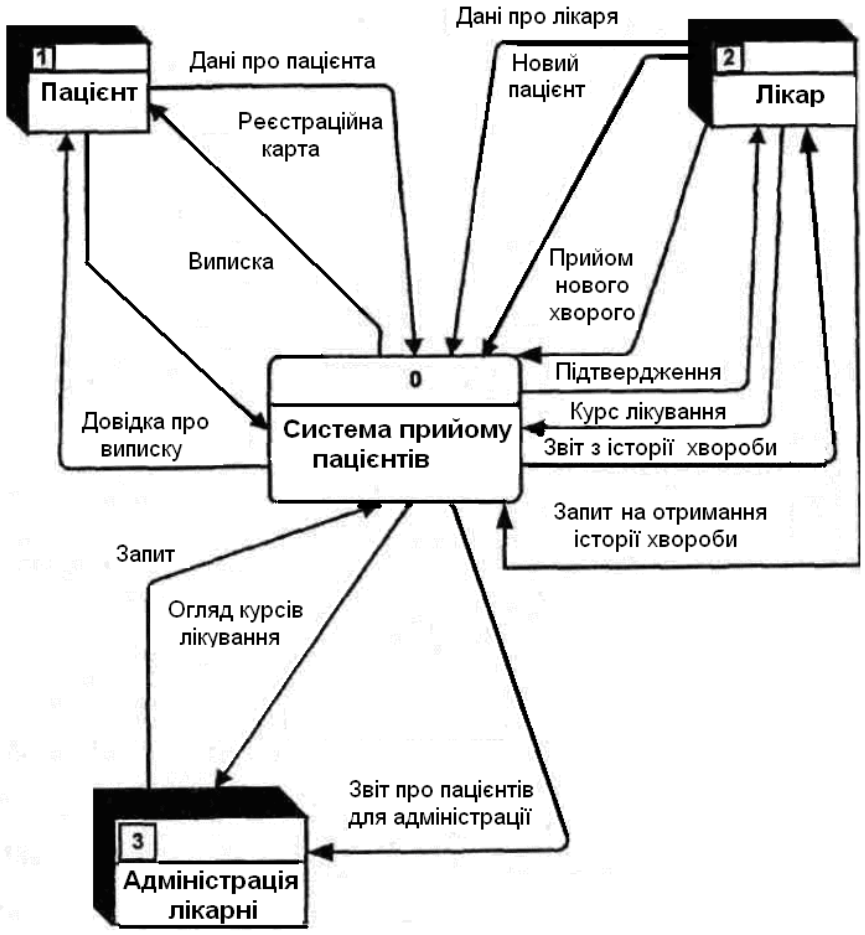



*Рис.* 1.1. Початкова контекстна діаграма

**Приклад специфікації структур даних**

**1. ЗВІТ_З_ІСТОРІЇ_ХВОРОБИ**

поточна_дата
початкова_дата
кінцева_дата
РЯДОК_ПАЦІЄНТА: номер пацієнта; ПІП; дата_народження
РЯДОК_ПРИЙОМУ *
номер_прийому / * порядковий номер прийому нового хворого * /
дата_прийому
номер_палати
РЯДОК_КУРСУ_ЛІКУВАННЯ *
Найменування захворювання
ім’я_лікаря
спеціалізація
РЯДОК_РЕЗУЛЬТАТІВ_ЛІКУВАННЯ *
дата
година
Примітка

**2. ІФОРМАЦІЯ_ВІД_ЛІКАРЯ**

прийом_нового_хворого
дані_про_лікаря
новий_пацієнт_амбулаторії
запит_на_отримання_історії_хвороби
курс_лікування

**3. ЗВІТ_ПРО_ПАЦІЄНТІВ**

поточна_дата
початкова_дата
кінцева_дата

РЯДОК_ПАЦІЄНТА*

номер пацієнта
ПІП
адреса
телефон
дата_народження

СТРАХУВАННЯ

страхова_компанія
номер_страховки

**Побудова початкового варіанту концептуальної моделі даних**

Виділимо і намалюємо сутності для кожного класу об'єктів даних в системі «Прийому пацієнтів». Розглянемо кожну можливу пару сутностей і встановимо існування зв'язку між ними. Намалюємо діаграму «сутність-зв'язок». Дамо найменування кожного зв'язку і поставимо її характеристики:

- потужність зв'язку між Пацієнтом і Прийомом, Прийомом і Курсом лікування, Лікарем і Курсом лікування дорівнює 0,
- відсутній зв'язок між Лікарем і Пацієнтом.

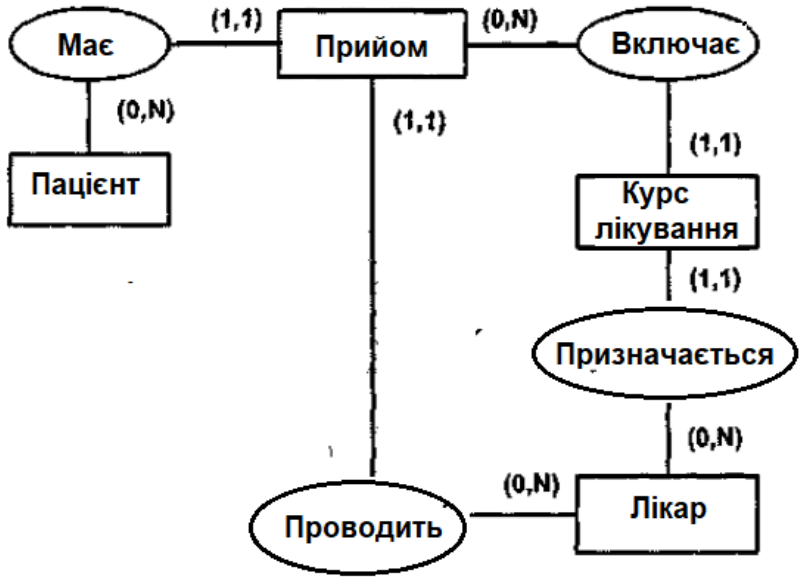



*Рис*. 1.2. Початковий варіант концептуальної моделі даних

### Побудова діаграм потоків даних нульового і наступних рівнів

Декомпозуємо початкову контекстну діаграму та складні процеси, перевіримо відповідність різних рівнів моделі процесів. Наведемо накопичувачі даних за допомогою структур даних. Опишемо процеси нижнього рівня за допомогою специфікацій. Результати наведено на рис. 1.3-1.9.

Опис накопичувачів даних і приклад специфікації процесу наведені нижче.

| **БД: пацієнти** | **БД: прийоми** | **БД: курси лікування** |
|---|---|---|
| номер пацієнта<br>ПІП<br>адреса<br>телефон<br>дата народження<br>група крові<br>страхування | номер прийому<br>номер пацієнта<br>дата прийому<br>дата виписки<br>номер палати | номер прийому<br>номер лікаря<br>найменування захворювання<br>ПРИМІТКА<br>дата<br>час |

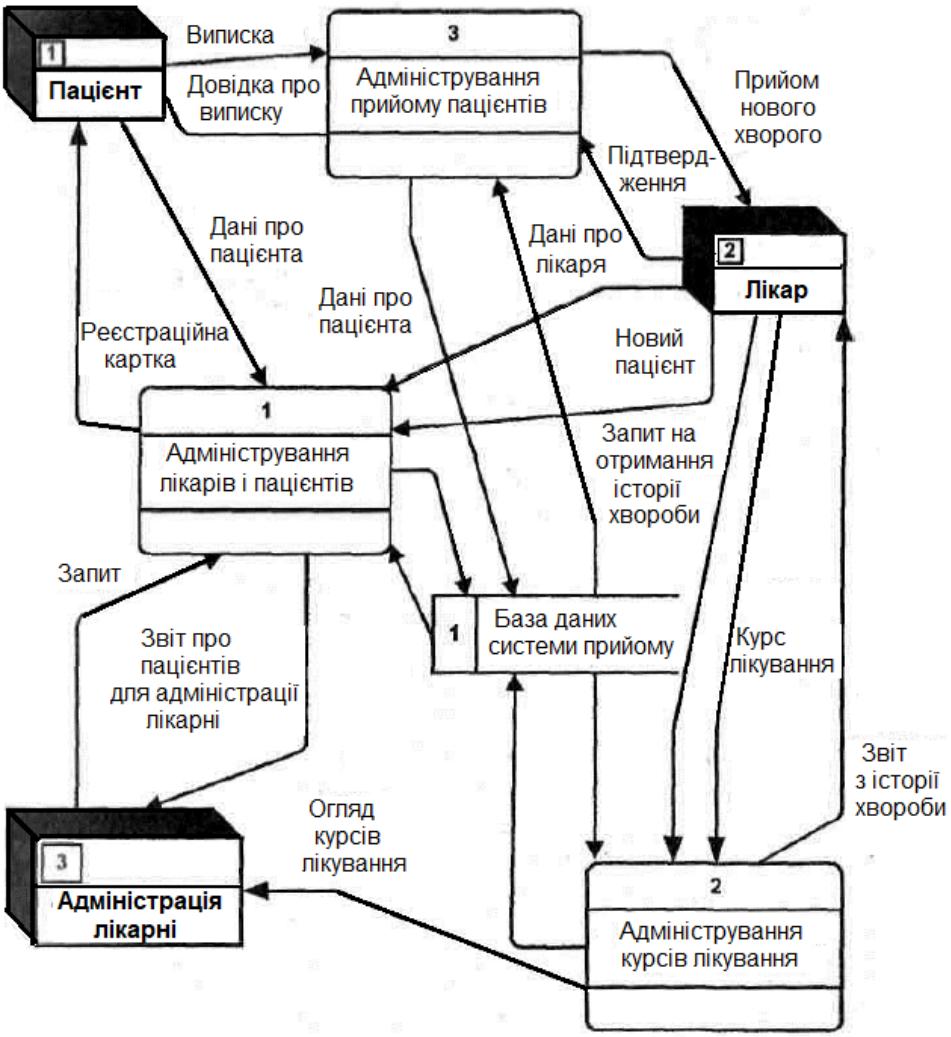



*Рис. 1.3*. Діаграма потоків даних нульового рівня

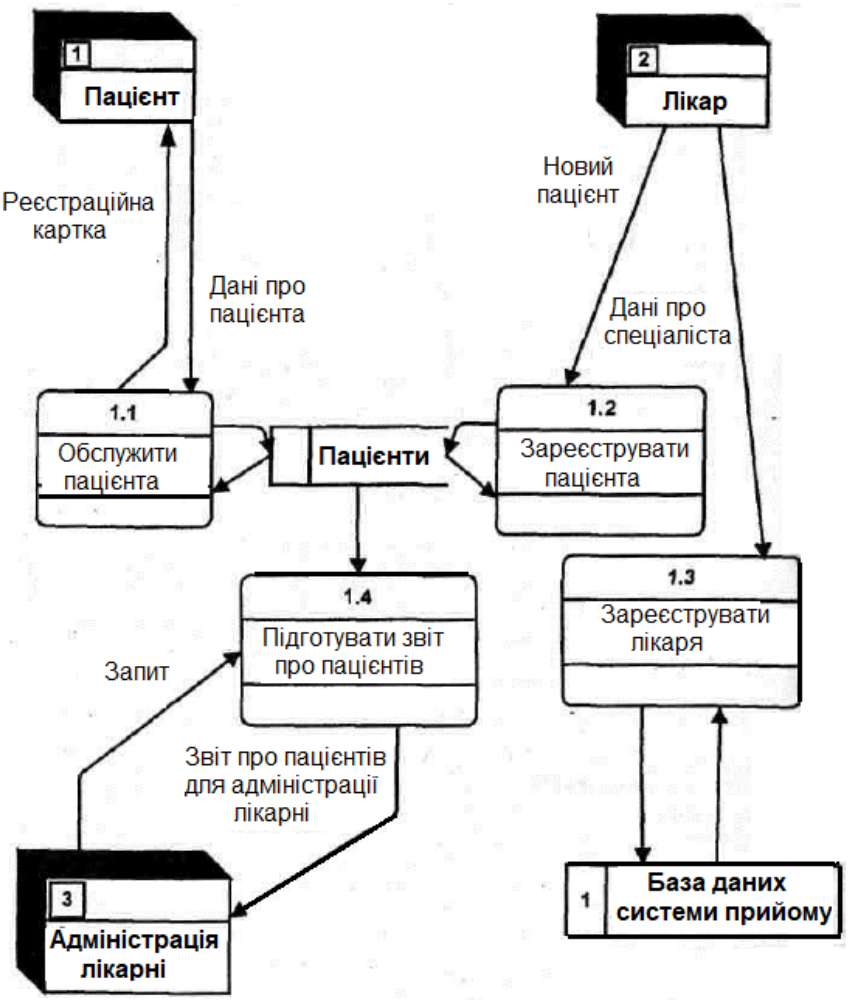



*Рис. 1.4.* Діаграма потоків даних першого рівня для процесу 1

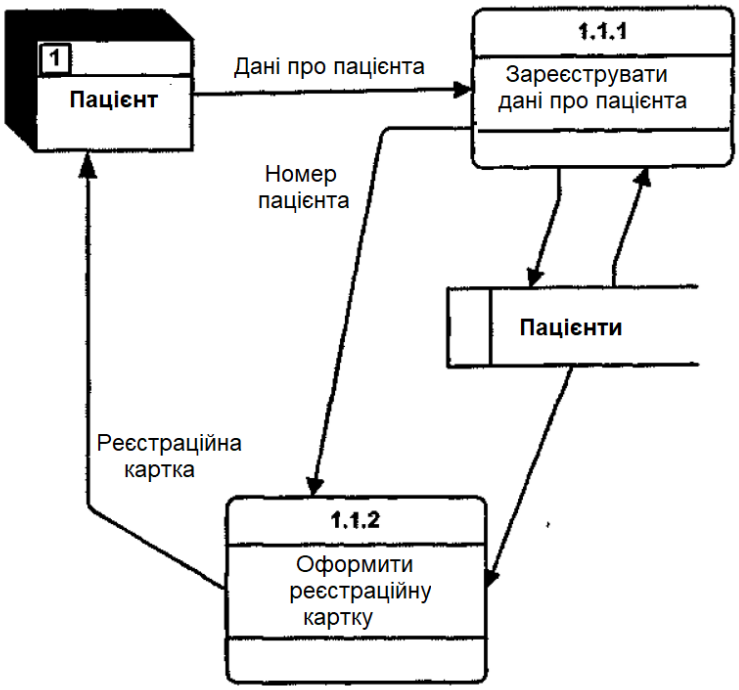



*Рис. 1.5.* Діаграма потоків даних другого рівня для процесу 1.1

**Накопичувач даних: лікарі**

номер лікаря
ім'я лікаря
адреса
телефон
дата народження
спеціалізація
номер кабінету
робочий телефон

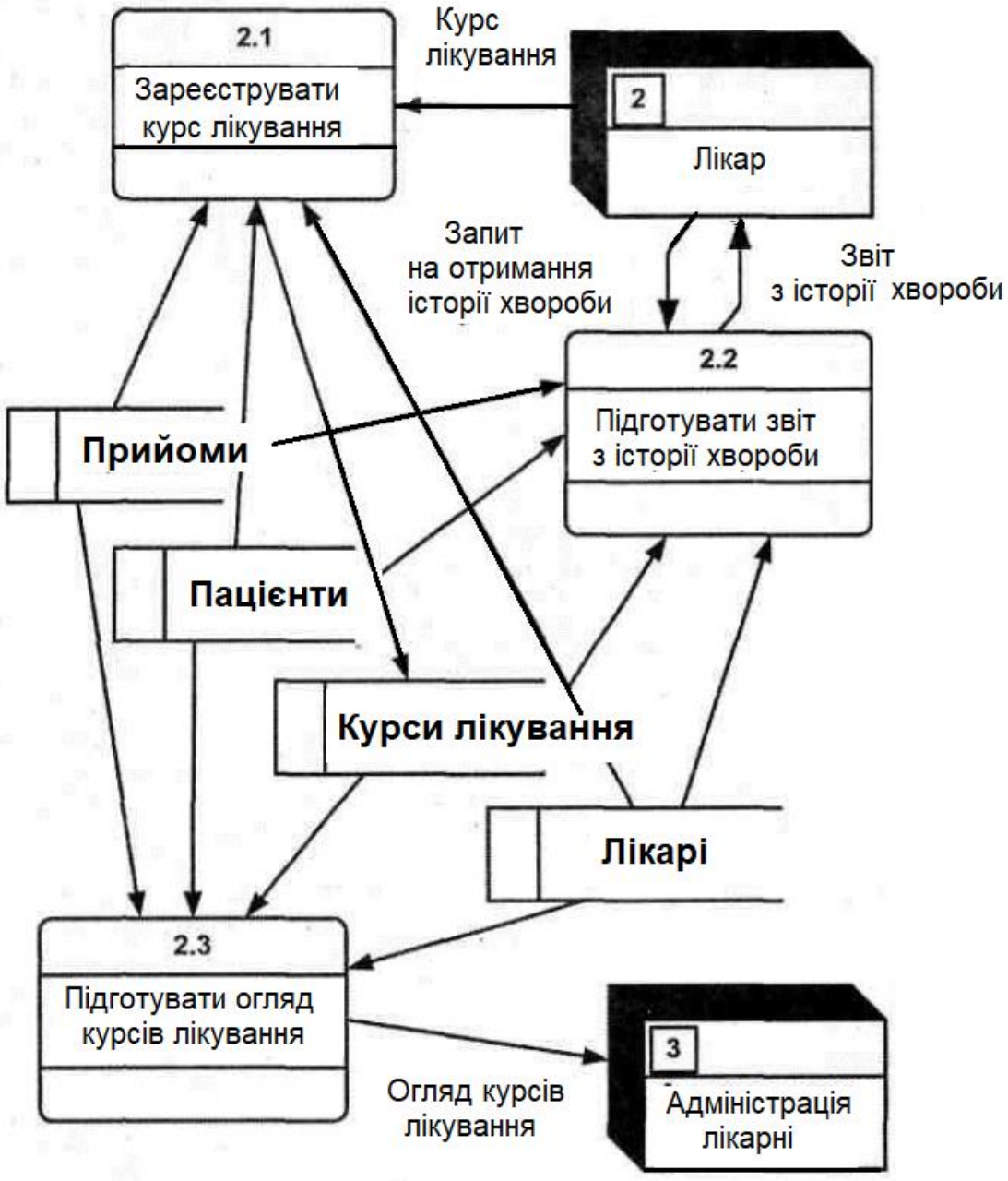



*Рис. 1.6*. Діаграма потоків даних першого рівня для процесу 2


*Рис.* 1.7. Уточнений варіант концептуальної моделі даних

**Уточнення концептуальної моделі даних**

Використовуючи побудовані структури даних, визначимо атрибути сутностей та уточнимо побудовану модель даних. Зовнішні ключі можна не показувати, оскільки вони визначаються зв'язками між сутностями. Виділимо атрибути-ідентифікатори і підкреслимо їх (рис. 1.8).

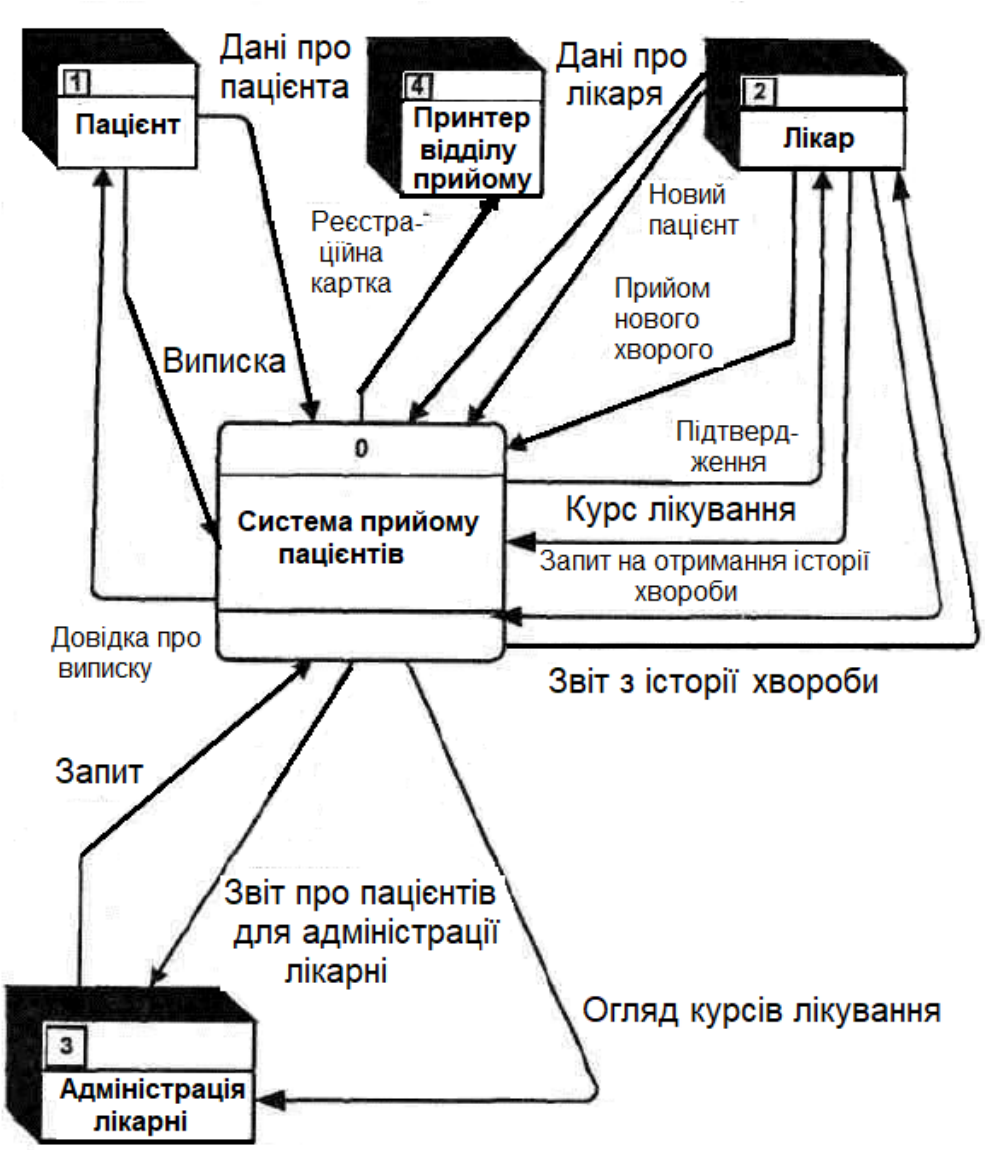



*Рис*. 1.8. Діаграма системних процесів нульового рівня

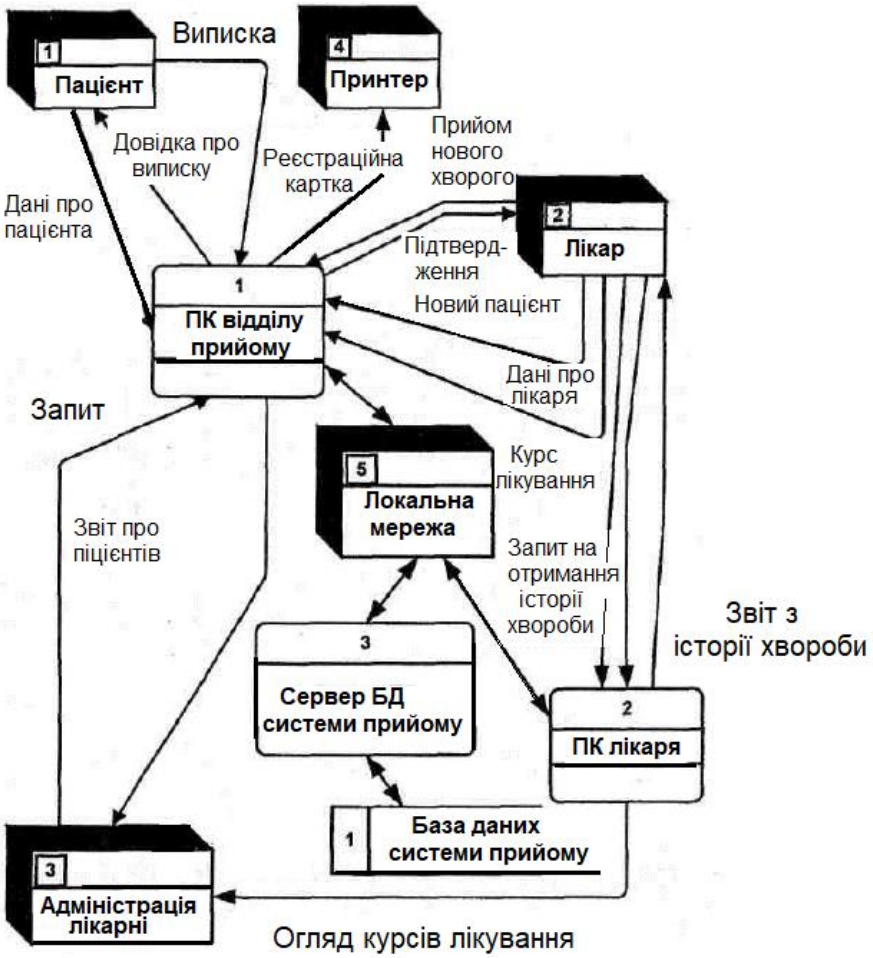



*Рис*. 1.9. Діаграма системних процесів першого рівня

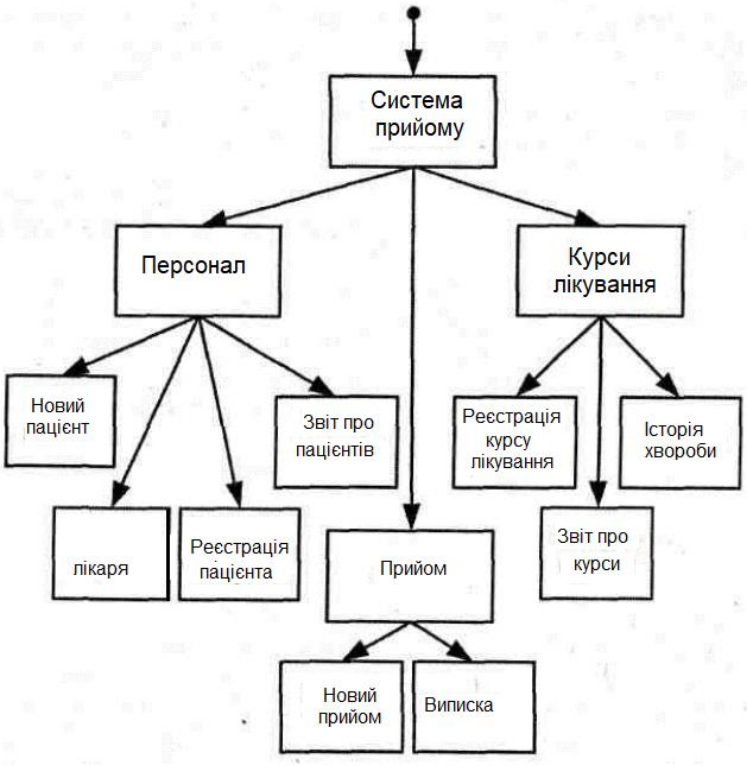



*Рис*. 1.10. Діаграма послідовності екранних форм

**Приклад 1.2.** Основи роботи в AllFusion Process Modeler 4.1

Побудова первинної моделі є динамічним процесом, протягом якого автори опитують компетентних осіб про структуру різних процесів, створюючи моделі діяльності підрозділів. При цьому їх цікавлять відповіді на такі питання:

1. Що надходить до підрозділу «на вході»?
2. Які функції і в якій послідовності виконуються в рамках підрозділу?
3. Хто є відповідальним за виконання кожної з функцій?
4. Чим керується виконавець при виконанні кожної з функцій?
5. Що є результатом роботи підрозділу (на виході)?

На основі наявних положень, документів і результатів опитувань створюють чернетку моделі.

Під час проведення аналізу роботи МІС були виявлені її цільові задачі, особливості функціонування кожного з модулів МІС та функціональна взаємодія між ними; зовнішні до МІС об'єкти та інформаційний вплив, а також нормативно-інформативна документація.

До літа 2018 діяла *Міжнародна класифікація хвороб десятого перегляду* (**МКХ-10**, *ICD-10*). З 18 червня 2018 введено в дію *Міжнародну класифікація хвороб одинадцятого перегляду* (**МКХ-11**, *ICD-11*), яку заплановано повністю впровадити у 2021 році (складається з 21-го розділу, кожен з яких містить підрозділи з кодами хвороб і станів).

*Цільова функція МІС* амбулаторії сімейної медицини: 1) надання своєчасної медичної інформації пацієнтам і лікарям; 2) обслуговування пацієнтів у вигляді надання медичної допомоги.

*Нормативна документація МІС амбулаторії сімейної медицини:* база лікарів амбулаторії; МКХ-10; довідник препаратів.

*Функції МІС амбулаторії сімейної медицини:*

- організація запису пацієнта на прийом до лікаря;
- підтримка режиму ведення даних про пацієнта - результату огляду сімейного лікаря;
- підтримка режиму постановки діагнозу та прийняття рішень.

Вхідними даними є пацієнт, який надходить до амбулаторії сімейної медицини.

Визначимо основний бізнес-процес, скориставшись визначеними цільовими функціями. Враховуючи той факт, що основне призначення МІС – надання своєчасної медичної інформації пацієнтам і лікарям, основним бізнес-процесом є ОГЛЯД ЛІКАРЯ.

Створимо контекстну діаграму, вона є вершиною деревовидної структури діаграм і зображує загальний опис системи та її взаємодію із зовнішнім середовищем. В моделі може бути ЛИШЕ ОДНА контекстна діаграма

Для побудови контекстної діаграми «процес роботи *амбулаторії сімейної медицини*» визначимо необхідну інформацію: 1) **вхід** - дані про пацієнта, що надходить до амбулаторії сімейної медицини; 2) **вихід** – електронна медична документація (ЕД); 3) **управління** – база даних про лікарів амбулаторії, МКХ-

10, довідник препаратів; 4) **механізми** – лікар, медична сестра, персональний комп'ютер.

Процес створення нової моделі охоплює такі кроки:

1. Запустіть **AllFusionBP** та оберіть режим роботи: «створити нову модель (**Create model**)».

2. В полі **Name** введіть ім'я моделі: **MIC амбулаторії сімейної медицини.**

3. В групі **Type** оберіть тип діаграми **Business Process- IDEF0** (рис. 1.11). Натисніть OK. Після цього з'явиться діалогове вікно **Properties for New Models** (рис. 1.12).

4. Вкажіть своє прізвище та ініціали. Натисніть OK. Відкриється стандартне вікно програми (рис. 1.13).

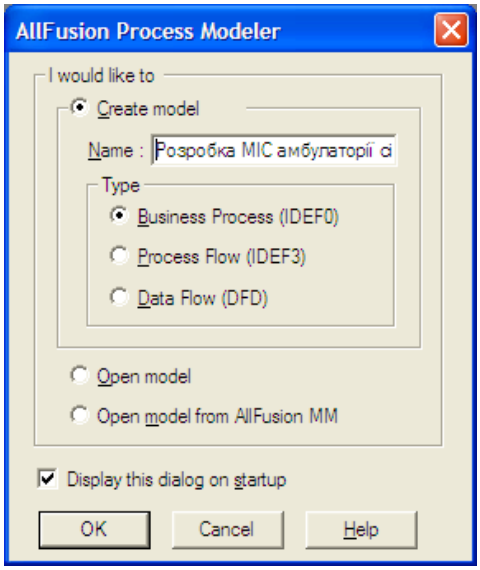



*Рис. 1.11.* Діалогове вікно створення моделі

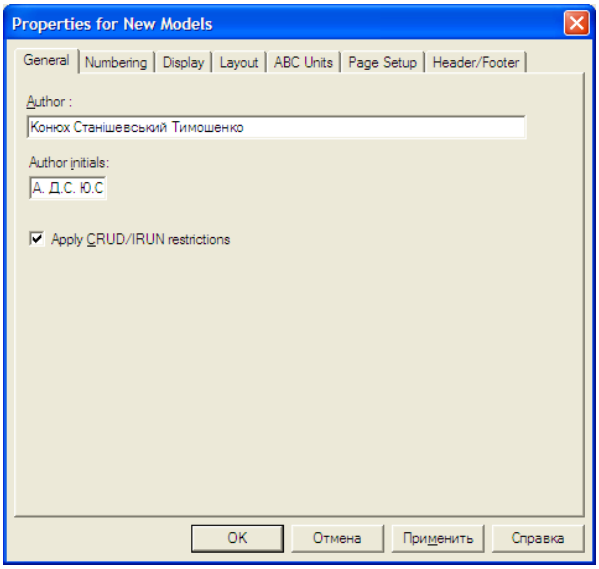



*Рис. 1.12.* Діалогове вікно Properties for New Models

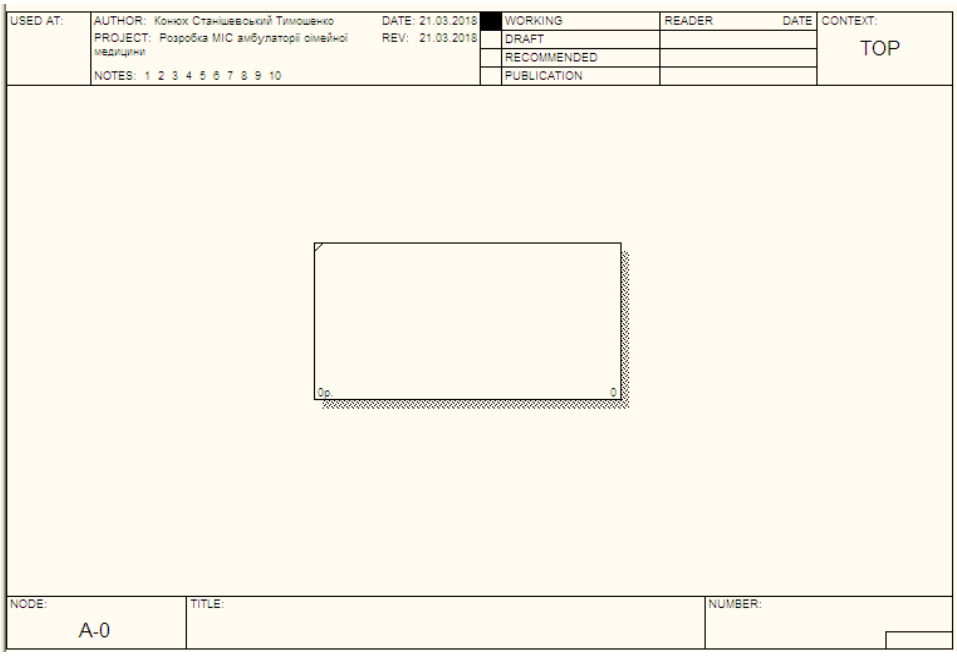



*Рис. 1.13.* Стандартне вікно

**Основні інструменти.** Усі основні дії з діаграмами (створення, редагування та інші) можна виконати за допомоги головного або контекстного меню. Принцип роботи з меню є стандартним Windows: об'єкт спочатку активується, а далі над ним виконуються дії. На основній панелі інструментів розміщено елементи керування. (рис. 1.14):


*Рис. 1.14.* Елементи керування

Функціональність панелі інструментів доступна з основного меню(табл. 1.1). На основній панелі інструментів розміщені елементи редагування для IDEF0-діаграм (рис. 1.6).

Побудова контекстної діаграми процесу «МІС амбулаторії сімейної медицини». Щоб ввести ім'я блоку, необхідно:

1. Натиснути ПКМ на блоку та обрати команду **Name**.
2. У діалоговому вікні ввести назву (рис.1.16).
3. Щоб введений текст мав коректне відображення, в контекстному меню обираємо пункт **Font**

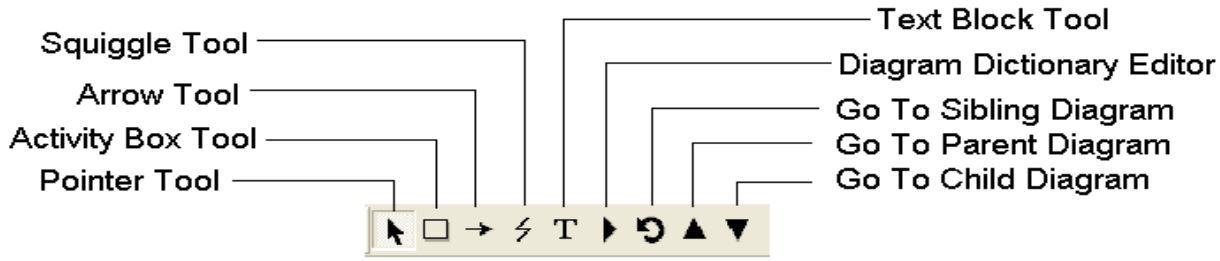



*Рис. 1.15*. Основна панель інструментів

Таблиця 1.1

| Позначка | Опис | Відповідний пункт Меню |
|---|---|---|
| | Створити нову модель | File/New |
| | Відкрити модель | File/Open |
| | Зберегти | File/Save |
| | Роздрукувати | File/Print |
| | Генератор звітів | Tools/Report Builder |
| 75% | Налаштування масштабу | View/Zoom |
| | Тип масштабування | View/Zoom |
| | Перевірка правопису | Tools/Spelling |
| | Провідник моделі | View/Model Explorer |
| | Додаткова панель інструментів | ModelMart |

Таблиця 1.2

| Позначка | Опис | Назва |
|---|---|---|
| | Вибір та виявлення позицій елементів, доданих до діаграми. | Pointer Tool |
| | Встановлення блоків діаграми | Activity Box Tool |
| | Встановлення дуг | Arrow Tool |
| | Створення "тільд" для поєднання блоку діаграми та її опису | Squiggle Tool |
| T | Створення текстових блоків | Text Block Tool |
| | Відображення наступної діаграми того ж рівня | Go to Sibling Diagram |
| | Діалогове вікно редагування діаграм | Diagram Dictionary Editor |
| | Батьківська діаграма | Go to Parent Diagram |
| | Діаграма - нащадок | Go to Child Diagram |

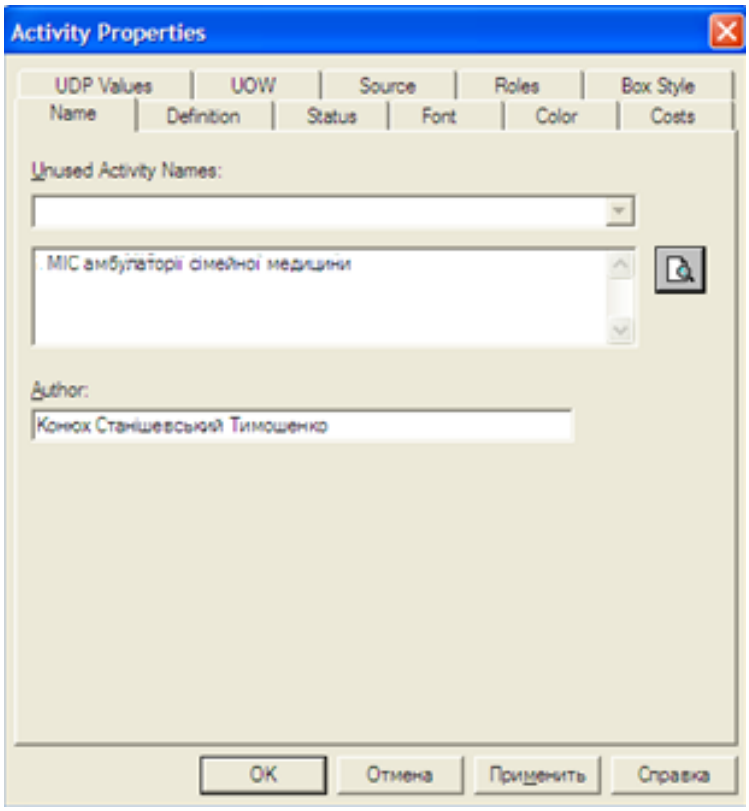



*Рис. 1.16.* Діалогове вікно

4. У діалоговому вікні **Activity Properties** в нижній частині вкладки **Font** встановлюємо позначки в опціях **Apply setting to**, що дозволяють змінити шрифт для всіх робіт на даній діаграмі, в моделі та групі **Global**, що дозволяє змінити шрифт одночасно для всіх об'єктів моделі. В опції **Script** обираємо **«кириллический».**

5. Встановлюємо шрифт **Arial Unicode MS**, курсив, 16 пт (рис. 1.17).

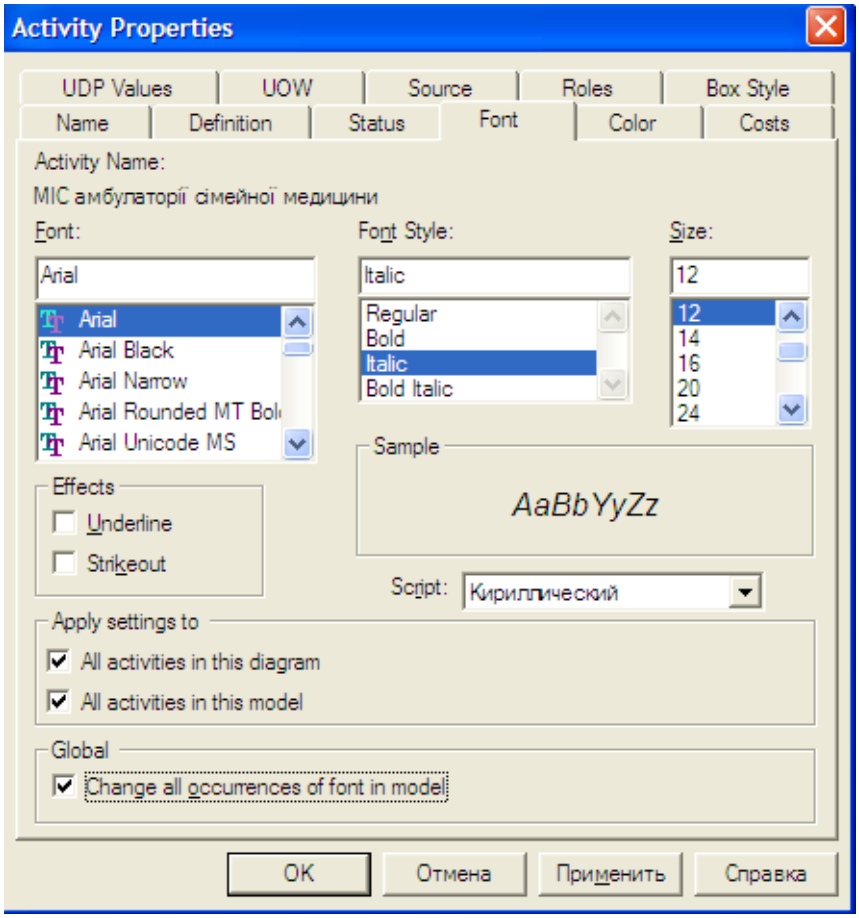



*Рис. 1.17.* Вкладка Font діалогу Activity Properties

В результаті буде отримана робоча область (рис. 1.18)

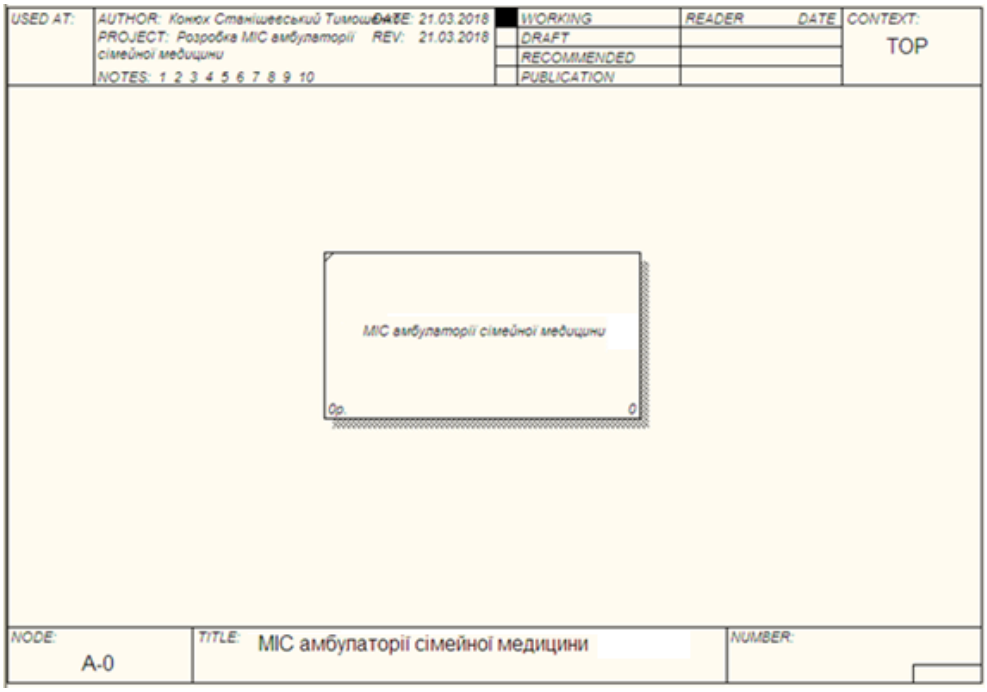



*Рис. 1.18.* Робоча область

**Приклад 1.3.** *Діаграма декомпозиції першого рівня*. Для побудови дуги керування необхідно:

1. Натиснути кнопку  .
2. Підвести курсор до верхнього краю робочого вікна побудови діаграми, поки не з’явиться чорна лінія (рис. 1.19).
3. Провести лінію до блоку і натиснути на трикутник (рис. 1.20).

*Рис.* 1.19. Початок побудови дуги

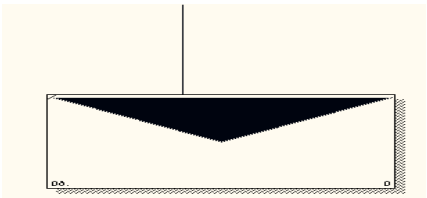


*Рис.* 1.20. Побудова дуги керування

Побудова інших дуг проводиться аналогічно, окрім дуги виходу. Вона будується в зворотному порядку.

Ідентифікація дуги керування:

1. Натисніть на кнопку  .
2. Виділіть дугу і натисніть ПКМ.
3. Оберіть команду **Name** (рис. 1.21).
4. Введіть назву дуги: «Нормативна документація» (рис. 1.22).

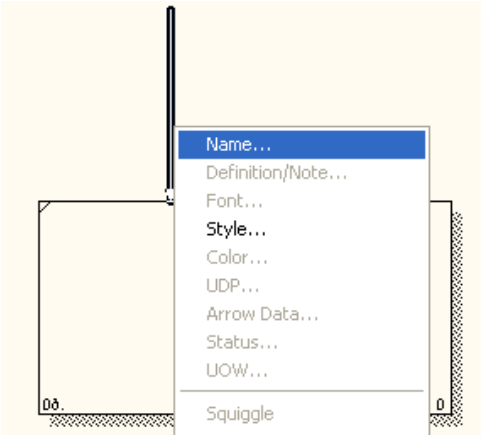



*Рис.* 1.21. Контекстне меню

BPwin дозволяє встановити шрифт за замовчуванням для об'єктів певного типу (наприклад, дуг) на діаграмах і в звітах (рис. 1.22):

**Context Activity** – робота на контекстній діаграмі;
**Context Arrow**- стрілки на контекстній діаграмі;
**Decomposition Activity**– роботи на діаграмі декомпозиції;
**Decomposition Arrow**– стрілки на діаграмі декомпозиції;
**Node Tree Text** – текст на діаграмі дерева вузлів;
**Frame User Text** – текст у каркасі діаграм;
**Frame System Text** – системний текст у каркасі діаграм;
**Text Blocks** – текстові блоки;
**Parent Diagram Text** – текст батьківської діаграми;
**Parent Diagram Title Text** – текст заголовка батьківської діаграми;
**Report Text**– текст звітів.

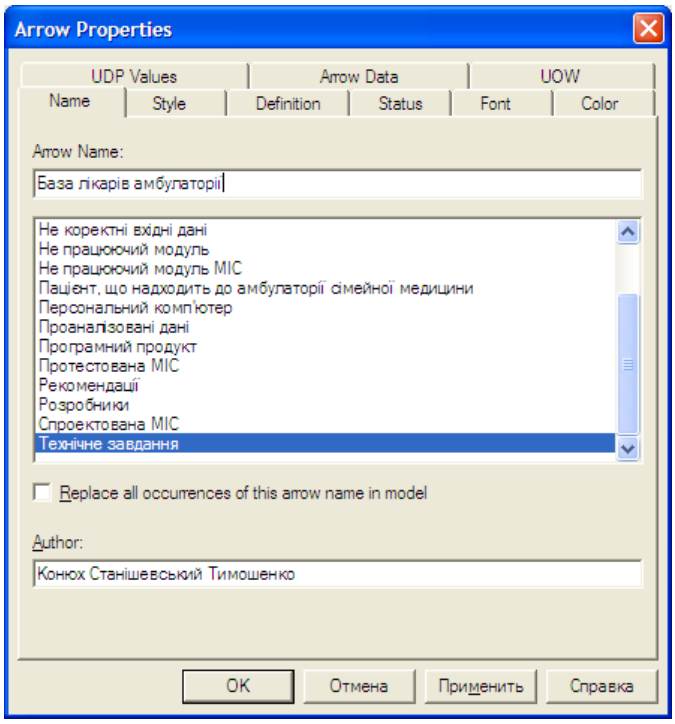



*Рис.* 1.22. Діалогове вікно Arrow Properties

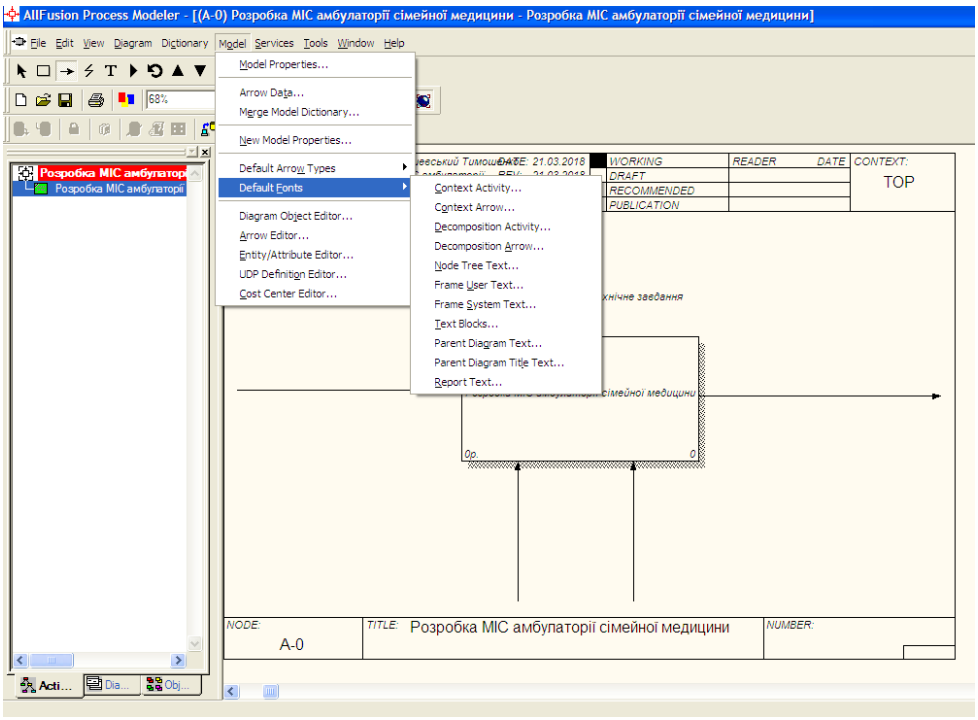



*Рис.* 1.23. Діалогове вікно Model - Default Fonts

5. В діалоговому вікні **Default Context Arrow Name Text Font** в нижній частині позначте **Change all occurrences**, це дозволить змінити шрифт для назв усіх дуг на даній діаграмі, в опції **Script** оберіть **«кириллический»** (рис. 1.24)**.**

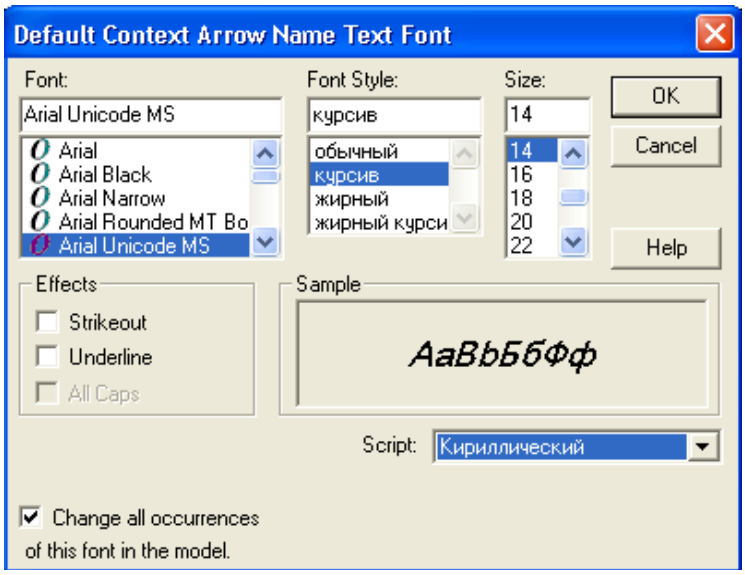



*Рис*. 1.24. Діалогове вікно Default Context Arrow Name Text Font

**Встановлення тільд.** Назва дуги є незалежним об’єктом, який можна зміщувати відносно дуги. Для встановлення тільди необхідно:

1. Натиснути на панелі інструментів ;
2. Натиснути ЛКМ по тексту, а потім по дузі;
3. Або в контекстному меню обрати **Squiggle** (рис. 1.25).

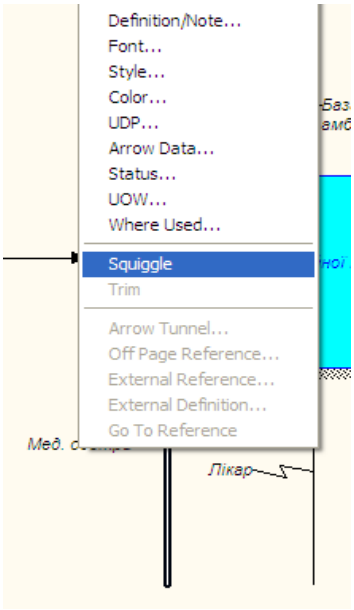



*Рис.* 1.25. Контекстне меню

Зміна кольорів тексту, блоку, зміна стилю і кольору дуг.

1. Для зміни кольору тексту виконайте команду з контекстного меню **Color** (рис. 1.26). Оберіть колір та натисніть кнопку OK (рис. 1.27).
2. Для зміни фону оберіть **Background Color** і колір (рис. 1.28).
3. Для зміни стилю дуг скористайтесь командою в контекстному меню **Style.** Вкажіть тип і стиль дуги, натисніть на кнопку ОК (рис. 1.29).

Для видалення блоку або дуги необхідно виділити потрібний об’єкт та натиснути кнопку **Delete**.

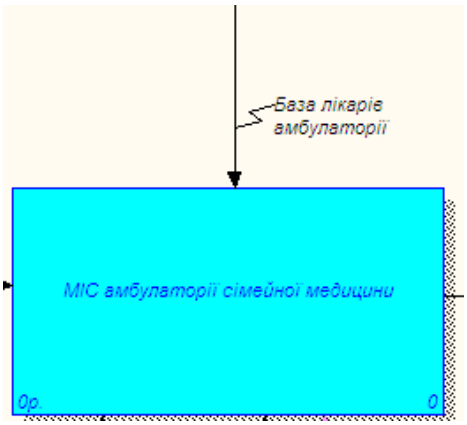



*Рис. 1.26.* Контекстне меню

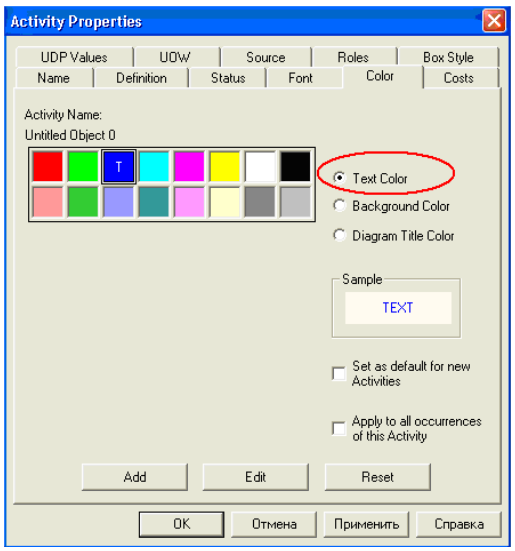



*Рис. 1.27.* Вікно вибору кольорів

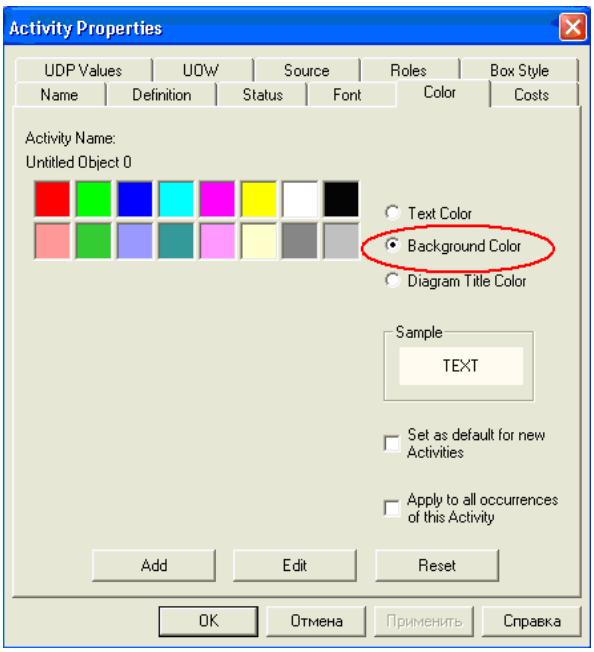



*Рис. 1.28.* Вкладинка Color діалогу Activity Properties

4. Необхідно відредагувати діаграму. Приблизний вигляд того, що маємо отримати зображено на рис. 1.30.

Збереження діаграми.

1. Вменю**File** натисніть**Save as**.
2. Вкажіть шлях та ім’я.
3. Натисніть **Save**.

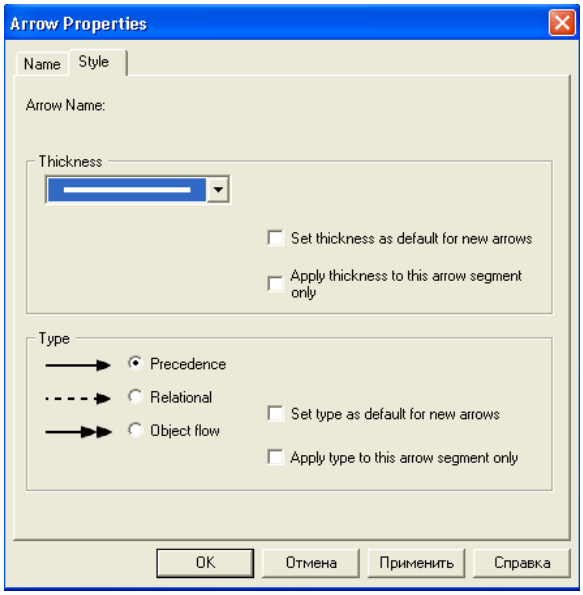



*Рис. 1.29.* Діалогове вікно Arrow Properties

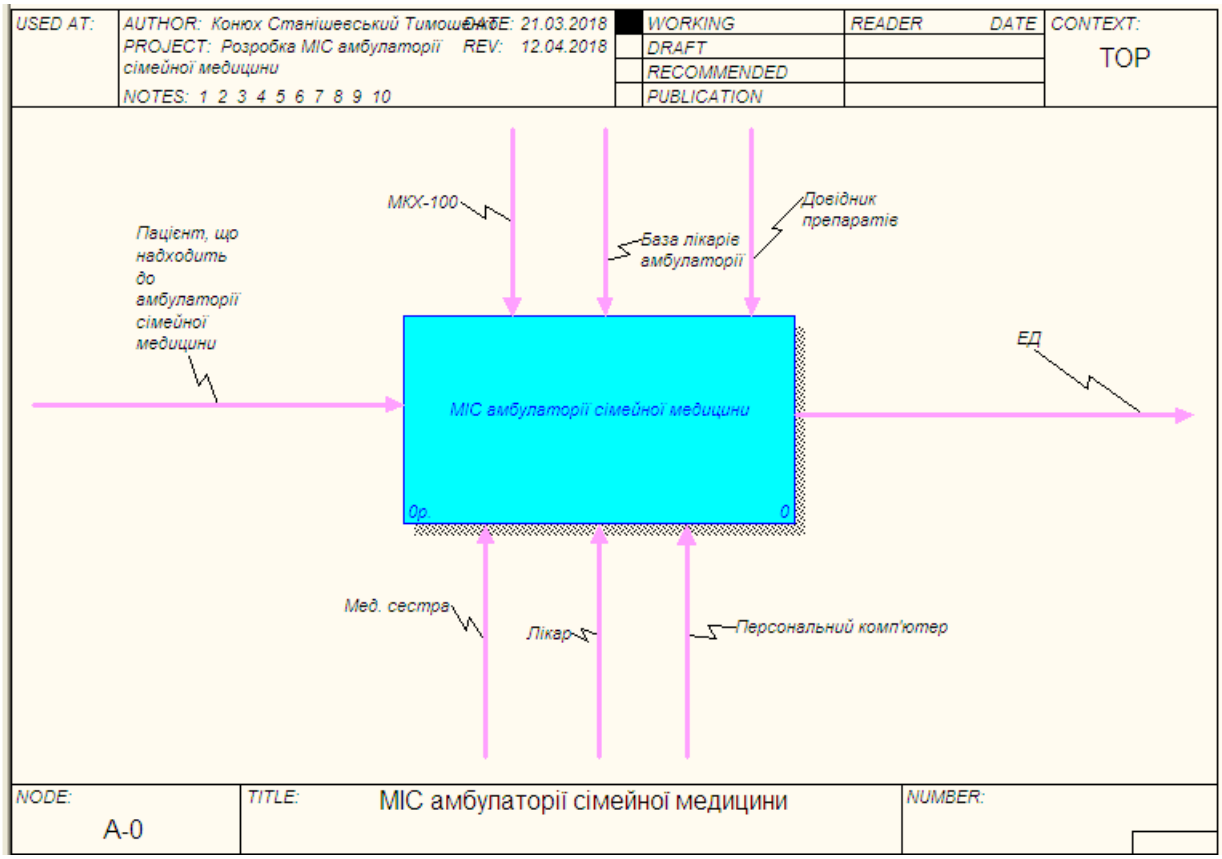



*Рис. 1.30.* Контекстна діаграма процесу «МІС амбулаторії сімейної медицини»

## Теоретичні відомості

**I. Моделювання предметної області: метод функціонального моделювання SADT** (IDEF0) [3 с. 133-147].

**Основні етапи моделювання.** Перш ніж розпочати моделювання SADT-аналітик здійснює підготовку до нього, збирає інформацію, декомпозує об'єкт і узагальнює цю декомпозицію. Підготовка включає:

1) *вибір мети* моделі (наприклад, опис того, як механічний цех виготовляє деталі),

2) *вибір точки зору, з якою буде подана модель* (наприклад, майстер, робочий);

3) *вибір типу створюваної моделі* (наприклад, модель "потокового" процесу);

4) *передбачуване використання побудованої* та перевіреної моделі (наприклад, *підготувати нового оператора*).

Таким чином, підготовка повинна максимально полегшити *збір інформації.*

Збір інформації може включати будь-яку комбінацію таких видів діяльності: читання документів, спостереження за існуючими операціями, анкетування групи експертів, опитування одного або декількох експертів, використання власних знань і придуманого опису роботи системи, яке згодом може бути відкориговане.

Декомпозуючи об'єкт, потрібно звернути увагу на вхідні та вихідні дані для всієї системи. Декомпозиція системи розпочинається зі складання списку основних типів даних та функцій. У процесі роботи необхідно визначити основні функції системи, розглядаючи всі нормальні та аномальні ситуації, зворотні зв'язки та випадки потенційних помилок. Потім ці списки забезпечуються коментарями для вказівки основних типів даних і функцій системи (або їх різних поєднань). Ці списки з коментарями використовують для створення діаграми, яка потім узагальнюється за допомогою контекстної діаграми.

**Джерела інформації.** Джерелом інформації можуть бути експерти, яким знайомі поточні нюанси й незадокументовані аспекти системи. Експертам відомі факти, які не відображені в документах або важко пояснити (ці факти іноді називають "казуальним знанням"), їх можна отримати тільки шляхом опитування експертів. Щоб підготуватися до такого опитування, необхідно досліджувати інші джерела інформації, наприклад документи. Існують різні стратегії виокремлення інформації з джерел: 1) читання документів; 2)спостереження за виконуваними операціями; 3) анкетування; 4) використання власних знань; 5) складання опису.

Документи є гарним джерелом інформації, тому що вони найчастіше доступні. *Читання документів* – це спосіб отримати первинне уявлення про систему, сформулювати питання до експертів і підтримувати бібліотеку документів. Для SADT-проєктів такі документи можуть зберігатися в

*бібліотеці проєкту* і розповсюджуватися у вигляді невеликих робочих пакетів (теків).

*Анкетування* проводиться для того, щоб опитати великі групи експертів в стислі терміни. Його можна використовувати, наприклад, коли необхідно швидко отримати відомості про роботу певної частини системи з різних позицій. Анкетування при опитуванні експертів дозволяє виявити, які частини системи понад усе потребують поліпшення. Проте, на практиці, часто інформація, отримана від експертів за допомогою анкет, виявляється недостовірною, тому необхідно ставити конкретні, чітко сформульовані питання.

Іноді досвідчені аналітики досліджують велику кількість систем певного типу (наприклад, виробництво, телефонна мережа, бухгалтерія). Під час реалізації багатьох проєктів вони набувають фундаментальних знань у відповідній наочній сфері щодо певного класу систем. Ці знання дуже корисні, оскільки для багатьох різних систем існують загальні поняття, що можуть мати широке застосування. Якщо аналітик добре знає предметну область, то може сам стати джерелом інформації.

У процесі аналізу, незалежно від джерел інформації, проводяться опитування декількох типів. Вибір того або іншого типу залежить від виду необхідної інформації та поставленої мети: 1) опитування для збору фактів; 2)опитування для визначення проблем; 3) наради для ухвалення рішень; 4) діалоги – автор/читач.

*Опитування для збору фактів* проводяться, коли намагаються визначити як функціонує система в конкретний час. *Опитування для визначення проблем* корисні, коли потрібно з'ясувати, що в системі не в порядку. *Наради для ухвалення рішень* проводяться, коли потрібно отримати уявлення про те, як повинна функціонувати майбутня система, щоб усунути недоліки у існуючій. Діалоги автор/читач – це неформальні обговорення при розбіжностях між автором і експертом.

**Вибір мети та точки зору.** Мета й точка зору моделі визначаються на першій стадії створення моделі. Вибір мети здійснюється з урахуванням питань, на які повинна відповісти модель, а вибір точки зору – відповідно до вибору позиції, з погляду якої описується система. Іноді мету й точку зору можна вибрати до того, як буде зроблена перша діаграма. Наприклад, мету моделі експериментального механічного цеху можна визначити наперед, тому що вона очевидна в постановці завдання: "зрозуміти обов'язки тих, хто працює в цеху так, щоб пояснити їх новому персоналу". Необхідно якомога раніше визначити мету й вибрати точку зору нової моделі. Мета повинна бути сформульована точно, з декількох різних точок зору, перш ніж вибрати одну з них.

Іноді виявляється, що визначити мету й точку зору на початку моделювання надзвичайно важко. У такому випадку необхідно скласти списки даних і функцій і, можливо, намалювати діаграму А0. Зробивши це, можна "відчути" систему і встановити, чи описує її діаграма А0 з *потрібної точки зору*. В цьому випадку необхідно розробити декілька альтернативних А0-діаграм, перш ніж з'явиться достатня упевненість, щоб здійснити вибір правильної мети й точки зору.

**Складання списку даних.** Списки об'єктів системи, що створюються під час моделювання, в SADT прийнято називати «списками даних» (термін «дані» тут вживається як синонім слова «об'єкт»). Отже, при обговоренні різних аспектів моделювання в SADT застосовуватимемо термін "список даних". Складання списку даних є початковим етапом створення кожної діаграми функціональної SADT-моделі.

Правило полягає в тому, щоб спочатку скласти список даних, а потім список функцій. Розробка списку діаграми розпочинається з виділення всіх основних груп і категорій даних, що використовуються і генеруються системою.

У сучасних аналітичних методах підвищена увага часто приділяється функціям у збиток (втрату) даним. Починаючи зі складання списку даних,

можна уникнути переходу до негайної функціональної декомпозиції. Списки даних допоможуть виконати глибший аналіз і при цьому не концентруватися на функціях системи, уникаючи пропусків, які часто виникають з упереджених уявлень про функціональні декомпозиції. Крім того, можна приділити належну увагу даним та ідентифікувати обмеження, що визначають функціональну декомпозицію.

SADT-діаграми зображують межу функцій і обмеження, що накладаються на них, причому обмеження повинні бути присутніми в усіх системах. Зазначаючи спочатку обмеження, виявляють природну структуру системи. Без обмежень функціональна SADT-діаграма є не більше ніж схемою потоків даних. Без обмежувальних дуг діаграми не зможуть розповісти проєктувальникові, чому аналітик вибрав конкретну декомпозицію. Завдяки тому, що в SADT розрізняються вхідні дуги і дуги управління (інформація, необхідна для пояснення процесу декомпозиції), SADT-діаграми чітко пояснюють систему, що вивчається, і причину такої декомпозиції.

**Складання списку функцій.** Закінчивши *список даних*, необхідно приступити з його допомогою до складання *списку функцій*. Для цього уявіть собі функції системи, що використовують той або інший клас (тип) або набір даних. Декілька різних типів даних можуть використовуватися однією функцією. Необхідно вибрати, які типи або набори даних необхідні для кожної конкретної функції. Це дозволить виділити дані схожих типів, які потім можна об'єднати в метатипи.

У міру просування за списком, необхідно перевіряти, чи правильні первинні уявлення, які часто можуть не збігатися з вибраною метою та точкою зору моделі. З іншого боку, не слід автоматично відкидати первинні ідеї, якщо вони здаються неправильними. Подальші роздуми можуть прояснити внутрішні аспекти роботи системи, не очевидні на перший погляд, і ви, можливо, відновите початкові ідеї після побудови декількох інших діаграм.

*Список функцій повинен знаходитися на одній сторінці із списком даних.* При цьому при складанні початкового списку не потрібно об'єднувати функції

між собою. Натомість спочатку необхідно зосередитися на кожній конкретній функції та її відношенні до груп даних. Крім того, потрібно підбирати такі функції, які могли б працювати з найбільш загальними типами даних з вашого списку. Що стосується прикордонних функцій (функцій, які можуть виконуватися або системою, або її оточенням), то спочатку дуже важко визначити, входять вони в модель чи ні.

Після цього необхідно об'єднати функції в "агрегати" – організація 3 – 6 функціональних угрупувань. Основною вимогою є те, щоб ці угрупування мали один і той же рівень складності, містили приблизно однаковий "обсяг" функціональності та функції в кожній з них мали схожі операції й цілі.

**Процес моделювання.** Процес моделювання в SADT включає такі кроки:

1) збір інформації про досліджувану область,

2) документування отриманої інформації і подання її у вигляді моделі,

3) уточнення моделі за допомогою ітеративного рецензування.

Цей процес підказує певний шлях виконання узгодженої та достовірної структурної декомпозиції, що є ключовим моментом у кваліфікованому аналізі системи. SADT унікальна за своєю здатністю забезпечити як графічну мову, так і процес створення несуперечливої та корисної системи описів.

SADT об'єднує ітеративний процес створення моделі, нотації щодо керування конфігурацією моделі, мову посилань для діаграм, мову функцій моделей з графічною мовою опису системи, а також рекомендації щодо реалізації аналітичних проєктів. Нотації щодо керування конфігурацією гарантують, що нові діаграми будуть коректно вбудовані в ієрархічну структуру моделі. *Мова посилань* в SADT, правила скорочень для посилань, адресованих до окремих частин діаграми, полегшують оформлення зауважень при рецензуванні моделі. *Мова функцій* дозволяє декларативно визначати правила роботи системи, що часто є особливо важливим завершальним кроком в описі системи.

*Процес моделювання в* SADT є ітеративною послідовністю кроків, що приводять до точного опису системи.

1. *Отримання знань у процесі опитування.* У процесі моделювання відомості про систему, що вивчається, отримують за допомогою випробуваної методики збору інформації –опитувань або інтерв'ю. Для отримання найповнішої інформації SADT пропонує використовувати різні її джерела (наприклад, читати документи, опитувати людей, спостерігати за роботою системи). Незалежно від конкретного джерела інформації методологія SADT рекомендує керуватися певною метою при його використанні. Це означає, що ви повинні визначити свої потреби в інформації перш ніж вибрати чергове джерело. Під час опитування графічна мова SADT використовується як засіб для нотаток, які є основою для побудови діаграм.

2. *Документування отриманих знань.* Створення моделі – це другий важливий етап у процесі моделювання, на якому аналітик документує отримані ним знання про задану проблемну область, подаючи їх у вигляді однієї або декількох SADT-діаграм. Процес створення моделі здійснюється за допомогою спеціального методу деталізації обмеженого суб'єкта. Тобто в SADT автор спочатку аналізує об'єкти, які входять в систему, а потім використовує отримані знання для аналізу функцій системи. На основі цього аналізу створюється діаграма, в якій об'єднуються схожі об'єкти і функції. Цей конкретний шлях проведення аналізу системи і документування його результатів є унікальною особливістю методології SADT.

3. *Коректність моделі перевіряється у процесі ітеративного рецензування.* Моделі створюють виходячи з реальної ситуації і ці моделі проходять через серію послідовних покращень, поки вони не будуть зображувати реальний світ. Однією з основних компонент методології SADT є ітеративне рецензування, у процесі якого автор та експерт багато разів радяться (усно й письмово) щодо достовірності створюваної моделі.

*Ітеративним рецензуванням* називається цикл «автор – читач».

Цикл «автор – читач» розпочинається в момент, коли автор ухвалює рішення розповсюдити інформацію про будь-яку частину своєї роботи, щоб отримати відгук про неї. Матеріал для розповсюдження оформляється у вигляді

"***тек***" – невеликих пакетів з результатами роботи, які протягом певного часу критично обговорюються іншими фахівцями. Зроблені письмові зауваження також розміщують в теку у вигляді нумерованих коментарів. Теки із зауваженнями є, таким чином, зворотним зв'язком, який автори отримують на свою роботу.

*Читачі* – це ті, хто читає і критикує створювану модель, а потім поміщає зауваження в теки. Їх робота можлива завдяки тому, що графічна мова SADT-діаграм дозволяє створювати діаграми і моделі, які можна легко і швидко читати.

Зазвичай окремий тек рецензується одночасно декількома читачами, і всі їх зауваження до певного часу надходять до автора. Потім автор відповідає на кожне зауваження й узагальнює критику, що міститься в зауваженнях. За допомогою таких обговорень можна швидко обмінюватися ідеями.

Таким чином, методологія SADT підтримує як паралельний, так і асинхронний перегляд моделі, що є ефективним способом розподілу роботи в колективі. Зазвичай, над різними частинами моделі можуть спільно працювати багато авторів, тому що кожен функціональний блок моделі подає окремий суб'єкт, який може бути незалежно проаналізований і декомпозований. Таким чином, модель сама координує роботу колективу авторів, тоді як процес моделювання SADT координує сумісне рецензування ідей, що виникають.

4. *Координація процесу рецензування.* Організація своєчасного зворотного зв'язку має найважливіше значення для ефективного моделювання, тому що застаріла інформація потенційно здатна звести нанівець усі зусилля щодо розробки системи. Ось чому SADT виділяє спеціальну роль спостерігача за процесом рецензування.

5. *Використання моделі після її схвалення.* SADT-моделі створюються із конкретною метою, яка визначена на діаграмі A-0 моделі. У деякому сенсі ця мета визначає як використовуватиметься модель. Таким чином, як тільки завершено створення моделі з необхідним рівнем деталізації та модель перевірена, її можна застосовувати для досягнення поставленої мети.

Наприклад, модель експериментального механічного цеху створена для опису діяльності різних працівників механічного цеху, хоча результуюча модель завжди призначалася як основа навчального допоміжного матеріалу для нового персоналу. Якщо ця модель описує роботу персоналу в цеху, але не може служити для підготовки навчального допоміжного матеріалу – вона даремна.

У процесі SADT-моделювання рекомендується виділити спеціальну групу людей, відповідальних за те, що створювана в процесі аналізу модель буде точна та використовувана надалі. Це – Комітет технічного контролю, який знаходиться в найбільш вигідному положенні при визначенні поточного напряму розвитку проєкту і виробленні пропозицій з його корегування. Комітет реалізує це за допомогою рецензій. Моделі, які досягли бажаного рівня деталізації та точності з погляду технічних вимог, прямують членам Комітету технічного контролю для обговорення і затвердження. Комітет оцінює, наскільки ця модель може бути реалізована.

Висока ефективність цього процесу обумовлена його організацією, в основі якої лежить розподіл функцій, які виконуються учасниками створення проєктів: експерти є джерелами інформації, автори створюють діаграми та моделі, бібліотекар координує обмін письмовою інформацією, читачі рецензують і затверджують моделі, а Комітет технічного контролю приймає і затверджує модель.

## II. Bpwin: загальні відомості

BPwin підтримує три методології моделювання:

1) функціональне моделювання (IDEF0),

2) опис бізнес-процесів (IDEF3),

3) діаграми потоків даних (DFD).

Тут можлива побудова змішаних моделей: модель може містити одночасно діаграми IDEF0, IDEF3 і DFD.

Якщо збираються автоматизувати роботу організації, то на початкових етапах створення відповідної ІС необхідно зрозуміти, як працює ця організація. Керівник добре знає роботу в цілому, але не в змозі знати деталі роботи

кожного рядового співробітника. Рядовий співробітник добре знає, що діється на його робочому місці, але може не знати, як працюють колеги. Тому для опису роботи підприємства необхідно побудувати модель, яка буде адекватна предметній області та містити в собі знання всіх учасників бізнес-процесів організації.

Найбільш зручною мовою моделювання бізнес-процесів є IDEF0, де систему зображують як сукупність взаємодіючих робіт або функцій. Функції системи аналізуються незалежно від об'єктів, якими вони оперують, що дозволяє чітко змоделювати логіку і взаємодію процесів організації.

Основу методології IDEF0 складає графічна мова опису бізнес-процесів. Модель у нотації IDEF0 має вигляд сукупності ієрархічно впорядкованих і взаємопов'язаних діаграм. Кожна діаграма є одиницею опису системи і розташовується на окремому аркуші. Модель може містити чотири типи діаграм:

1) контекстну діаграму (кожна модель може мати тільки одну контекстну діаграму);

2) діаграми декомпозиції;

3) діаграми дерева вузлів;

4) діаграми тільки для експозиції (FEO).

Процес моделювання системи в IDEF0 розпочинається зі створення контекстної діаграми - діаграми найбільш абстрактного рівня опису системи в цілому, що містить визначення 1) суб'єкта моделювання, 2) мети, 3) точки зору на модель.

1. Під *суб'єктом* розуміється сама система, при цьому необхідно встановити, що входить в систему, а що лежить за її межами (тобто визначити, що буде розглядатися як компоненти системи, а що є зовнішнім впливом). На визначення суб'єкта системи істотно впливають 1) позиція, з якої розглядається система, 2) мета моделювання. Таким чином, на початку необхідно визначити область моделювання (в процесі моделювання область може корегуватися): опис області як системи в цілому, так і її компонентів є основою побудови

моделі. При формулюванні області визначення необхідно враховувати два компоненти: широту і глибину. Широта має на меті визначення меж моделі (що буде розглядатися всередині системи, а що зовні). Глибина визначає, на якому рівні деталізації модель є завершеною.

2. *Мета моделювання* визначається з відповідей на такі питання:

Чому цей процес повинен бути змодельований?

Що повинна показувати модель?

Що може отримати клієнт?

3. *Під точкою зору* розуміють перспективу, з якою спостерігалася система при побудові моделі. Точка зору повинна відповідати меті і межам моделювання. Зазвичай, вибирають точку зору людини, відповідальної за роботу в цілому, яку моделюють.

Модель в BPwin розглядається як сукупність робіт (дій), кожна з яких оперує деяким набором даних. Дії (activity) позначають зазначені процеси, функції або завдання, які відбуваються протягом певного часу і мають розпізнавані результати. Дії зображують у вигляді прямокутників. Всі дії повинні бути названі і визначені. Ім'я дії повинно мати вираз віддієслівного іменника, що позначає дію.

Стрілки описують взаємодію робіт і зображують собою певну інформацію, виражену іменниками. В IDEF0 розрізняють п'ять типів стрілок:

1. **Вхід** - матеріал або інформація, які використовуються або перетворюються дією (роботою) для отримання результату (виходу). Припускається, що дія (робота) може не мати жодної стрілки входу.

Кожен тип стрілок підходить до певної сторони прямокутника, що зображує дію (роботу), або виходить з неї. Стрілка входу малюється якщо входить в ліву грань роботи. Зазвичай складно визначити, чи є дані входом або управлінням: в цьому випадку підказкою може служити інформація про те, переробляються / чи змінюються дані під час роботи чи ні (якщо змінюються, то, швидше за все, це вхід, якщо ні - управління).

2. **Управління** - правила, стратегії, процедури або стандарти, якими керується робота. Кожна робота повинна мати хоча б одну стрілку управління, яку малюють зверху до дії. Управління впливає на роботу, але не перетворюється роботою. Якщо мета роботи - змінити процедуру або стратегію, то така процедура або стратегія буде для дії входом. У разі виникнення невизначеності в статусі стрілки (управління або вхід) рекомендують малювати стрілку управління.

3. **Вихід** - матеріал або інформація, які виробляються дією (роботою). Кожна дія повинна мати хоча б одну стрілку виходу (робота без результатів не має сензу). Стрілку виходу малюють з правої сторони дії.

4. **Механізм** - ресурси, які виконують дію (роботу), наприклад, верстати, пристрої тощо. Стрілку механізму малюють в нижній частині дії.

5. **Виклик** - спеціальна стрілка, яка вказує на іншу модель дії (роботи). Стрілку виклику малюють знизу. Її використовують для вказівки того, що інша робота виконується за межами системи, яку моделюють. У BPwin стрілки виклику використовуються в механізмі злиття та розділення моделей.

Стрілки на контекстній діаграмі служать для опису взаємодії системи з оточуючим світом. Вони можуть розпочинатися на межі діаграми і закінчуватися у дії, або навпаки. Такими стрілки називають граничними.

## 3. Побудова контекстної діаграми (діаграми А0).

Початковий зміст діаграми декомпозиції першого рівня А0 забезпечують списки даних і функцій. Для правильного опису системи змісту треба надати форму. У SADT це робиться за допомогою побудови низки діаграм і здійснюється таким чином: 1) розташуванням блоків на сторінці; 2)малюванням основних дуг, які подають обмеження; 3) малюванням зовнішніх дуг; 4)малюванням всіх дуг, що залишилися.

Правильне розташування блоків є найважливішим етапом побудови діаграми. Блоки розташовують відповідно до їх домінування (за ступенем важливості або за порядком проходження).

*Найдомінантніший блок* зазвичай розташовується у верхньому лівому кутку, а найменш домінантний – в нижньому правому. Це приводить до розташування, при якому більш домінантні блоки обмежують менш домінантні, утворюючи схему у вигляді сходинок. Домінування має найважливіше значення для чіткого подання процесу (наприклад, не має сенсу говорити про контроль за виконанням завдання до закінчення процесу виконання).

Після цього зображають основні дуги, що зображують обмеження: це є другою важливою частиною побудови діаграми А0. Вони дають підставу для розбиття об'єкта діаграми на 3 – 6 системних функцій, що зображаються блоками. Наприклад, *довідник стандартів якості* здійснює вирішальний вплив на те, як контролюються незавершені деталі. Малюючи ці дуги, перевіряйте, чи дійсно кожна з них здійснює вплив, який відповідає декомпозиції об'єкта. Прослідкуйте за списком даних, чи не відсутні якісь дуги, що зображують обмеження.

Основними дугами, які зображують обмеження, завжди є зовнішні дуги (дуги, що подають дані, які надходять з безпосереднього оточення діаграми).

Наступним кроком у побудові діаграми є розміщення решти зовнішніх дуг і призначення їм відповідних ICOM-кодів. Таким чином, усі дані, які входять в систему або виходять з неї, виявляються врахованими на рисунку. Втрата зовнішньої дуги – це помилка інтерфейсу, одна з найпоширеніших в системному аналізі. Займаючись декомпозицією об'єкта, можна забути про інтерфейсні дані, тому що легко зосередитися на деталях. Точність діаграми підвищиться, починаючи із зображення всіх зовнішніх дуг, включаючи всі інтерфейсні дані.

Останній етап – малювання решти всіх дуг, що відображають деталі роботи системи в цілому:

1) малювання обмежень, що залишилися, діють між блоками (наприклад, дане *креслення* впливає на перевірку деталі);

2) малювання основного потоку даних;

3) розгляд та аналіз всіх неправильних потоків даних (випадки виникнення помилок);

4) уточнення зворотних зв'язків у потоках даних (наприклад, *забраковане завдання* знову потрапляє в цикл як *брак*);

5) зображення всіх зворотних зв'язків, що викликаються помилковими ситуаціями.

Щоб надати деяку форму даним і функціям, найкраще зробити чернетку. Аналітикові рекомендується спочатку робити нарис-чернетку діаграми, а потім перемальовувати діаграму начисто, щоб уточнити своє розуміння, прояснити ситуацію та створити опис, який можуть подивитися інші.

**Узагальнення діаграми А0.** Узагальнення є останнім важливим кроком початкового етапу моделювання. Для будь-якої SADT-діаграми є батьківська діаграма, яка містить її контекст (де під контекстом розуміється блок з набором вхідних дуг, дуг управління і вихідних дуг). Верхня діаграма моделі (діаграма А0) не є винятком. Контекстом для неї буде діаграма А-0, що є узагальненням усієї моделі. Діаграма А-0 має декілька призначень:

1) вона оголошує загальну функцію всієї системи;

2) вона надає множину основних типів або наборів даних, які використовує або проводить система;

3) А-0-діаграма вказує взаємовідношення між основними типами даних, здійснюючи їх розмежування.

Таким чином, А-0-діаграма є загальним видом системи, яка вивчається.

При створенні діаграми А-0 використовують інформацію, зафіксовану на діаграмі А0. Спочатку в центрі SADT-діаграми малюють один великий блок, назва якого збігається з назвою діаграми А0. У цей момент слід перевірити, чи адекватно назва відображає те, що робить система. Всі зовнішні дуги діаграми А-0 зображуються на діаграмі А-0 та входять у відповідну сторону блоку. При цьому потрібно перевірити, що назва кожної дуги описує те, чим обмінюються система та її середовище. Після того, як аналітик зобразить вхідні і вихідні дуги, потрібно перевірити точність опису потоку даних. Намалювавши дуги

управління, переконайтеся, що саме вони управляють тим, як система перетворить вхідні дані у вихідні. Після закінчення проведених робіт необхідно перевірити мету з точки зору моделі під основним блоком і звірити її з тим, що подано блоком і його дугами.

У процесі узагальнення А-0 допомагає прояснити опис системи, тому що при узагальненні проглядаються мітки дуг для точнішого найменування даних, якими обмінюються система та її середовище. Крім того, дуги часто об'єднуються для спрощення зображення моделі. В цьому випадку дуги розгалужуються на свої складові на діаграмі А0. Побудова діаграми А-0 свідчить про закінчення початкового етапу моделювання. До цього моменту зроблена перша спроба узагальнити й описати основну діяльність системи та показати зв'язок системи з її середовищем.

Діаграми А-0 і А0 відображають всі основні входи, управління, виходи та функції системи. Загальний вигляд системи, отриманий за допомогою цих діаграм, є основною метою аналітика на початковому етапі побудови SADT-моделі.

## 2. Декомпозиція контекстної діаграми

*Мета та завдання практикуму:*1) навчитись проводити декомпозицію контекстної діаграми; 2) освоїти правила побудови дуг та «тунелювання» стрілок.

План.

1. Створення діаграми декомпозиції

**Завдання:**

1. Деталізувати головний бізнес-процес.
2. Деталізувати отримані підпроцеси.
3. Змінити напрям дуги.
4. Побудувати відгалуження дуги.
5. Провести «тунелювання» стрілок.
6. Створити зворотній зв’язок керування.
7. Зберегти роботу.
8. Дати відповідь на контрольні питання.
9. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи**

1.Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту

2.Продемонструвати виконане завдання викладачу.

**Контрольні запитання:**

1. Як створити діаграму верхнього рівня?
2. Як на діаграмі відображена декомпозиція?
3. Як регулювати кількість блоків для декомпозиції?
4. Для чого призначений зворотній зв’язок керування та тунелювання?
5. Що таке «мігруюча дуга»?

### Аудиторна робота

**Приклад 2.1.** Деталізація бізнес-процесу «МІС амбулаторії сімейної медицини».

Відкрийте збережений проєкт. Виконаємо деталізацію контекстного процесу за допомогою діаграми верхнього рівня. Ця діаграма містить в собі чотири процеси:

- запис на прийом;
- огляд сімейного лікаря;
- діагностичне відділення;
- підтримка прийняття рішення.

Проведіть деталізацію процесу «МІС амбулаторії сімейної медицини», задавши потрібну кількість нових блоків. Для цього:

1. Натисніть на блок «МІС амбулаторії сімейної медицини» і виберіть інструмент  .

2. У діалоговому вікні введіть число, на яке буде проведена декомпозиція – 4.

3. Вкажіть тип діаграми IDEF0 (рис. 2.1.) и натисніть OK.

4. Вкажіть назву нових блоків.

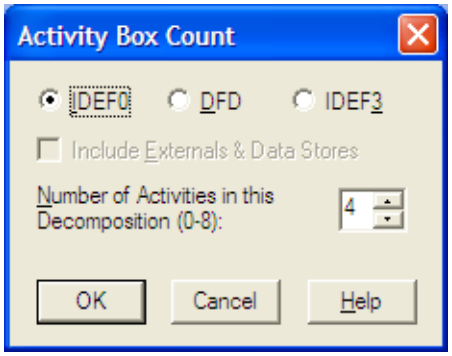



*Рис.* 2.1. Діалогове вікно декомпозиції блоку

При декомпозиції функції, вхідні в неї и вихідні із неї дуги автоматично з'являються на діаграмі декомпозиції (міграція дуг), але при цьому не доторкаються блоків. Такі стрілки називаються *незв'язаними* и сприймаються програмою як синтаксична помилка (рис. 2.2).

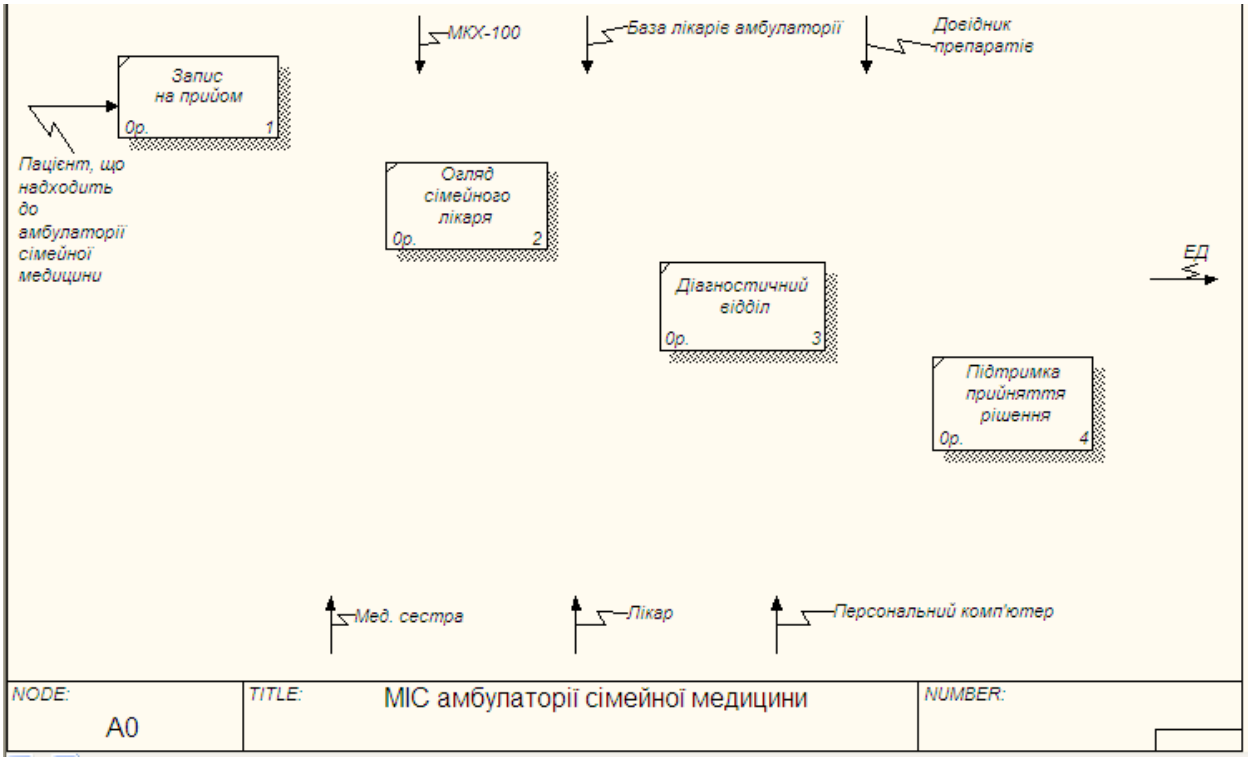



*Рис.* 2.2.Декомпозиція верхнього рівня

Визначимо вхідні и вихідні потоки для нових процесів.

**Процес 1.1. Запис на прийом**

1. Вхід – пацієнт, що надходить до амбулаторії сімейної медицини.
2. Вихід – зареєстрований пацієнт або не коректна реєстрація.

Виконаємо зв'язування мігруючих дуг:

1) виберіть інструмент створення дуг;
2) клацніть мишею по накінцівнику вхідного потоку ВХІДНІ ДАНІ;
3) клацніть по вхідній стороні блоку аналізу даних.

Для побудови вихідного потоку ЗАРЕЄСТРОВАНИЙ ПАЦІЄНТ виконайте дії:

1. Виберіть інструмент створення дуг .
2. Натисніть ЛКМ по вихідній стороні блоку ЗАПИС НА ПРИЙОМ.
3. Натисніть по вхідній стороні блоку ОГЛЯД СІМЕЙНОГО ЛІКАРЯ.
4. Виберіть інструмент текст, в контекстному меню – команду **Name**, вкажіть назву дуги ЗАРЕЄСТРОВАНИЙ ПАЦІЄНТ. Перевірте себе (рис. 2.3).

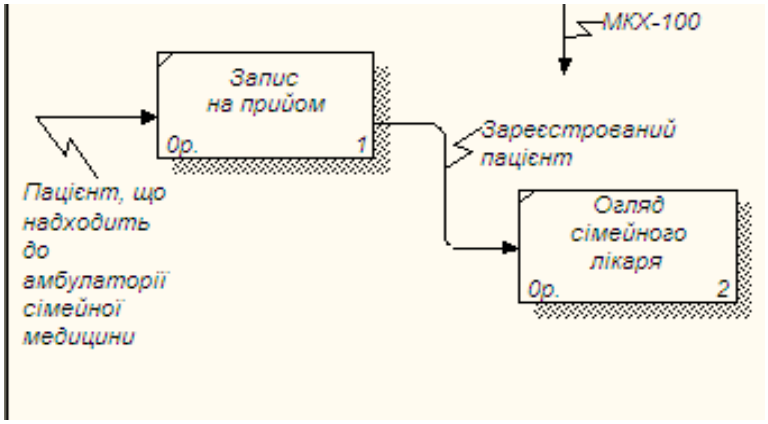



*Рис.* 2.3. Фрагмент діаграми

**Деталізація процесу «МІС амбулаторії сімейної медицини».**

1. Слідуючи прикладу, необхідно виконати деталізацію процесів:

**Процес 1.2. Огляд сімейного лікаря**

1. Вхід – ЗАРЕЄСТРОВАНИЙ ПАЦІЄНТ.
2. Вхід – ДІАГНОСТИЧНІ ДАНІ
3. Вихід – НАПРАВЛЕННЯ.

**Процес 1.3. Діагностичне відділення**

1. Вхід – НАПРАВЛЕННЯ.
2. Вихід – ДІАГНОСТИЧНІ ДАНІ.

**Процес 1.4. Підтримка прийняття рішення**

1. Вхід – ДІАГНОСТИЧНІ ДАНІ.
2. Вихід – ЕД.

**Зміна напрямку дуги:**

На Виході ДІАГНОСТИЧНІ ДАНІ не виходить за межу моделі, а повертається в процес ОГЛЯД СІМЕЙНОГО ЛІКАРЯ.

1. Виберіть інструмент створення дуг .
2. Натисніть ЛКМ на Виході блоку ДІАГНОСТИЧНЕ ВІДДІЛЕННЯ.
3. Натисніть ЛКМ на Вході блоку ОГЛЯД СІМЕЙНОГО ЛІКАРЯ.
4. Назвіть нову дугу – ДІАГНОСТИЧНІ ДАНІ (рис. 2.4).

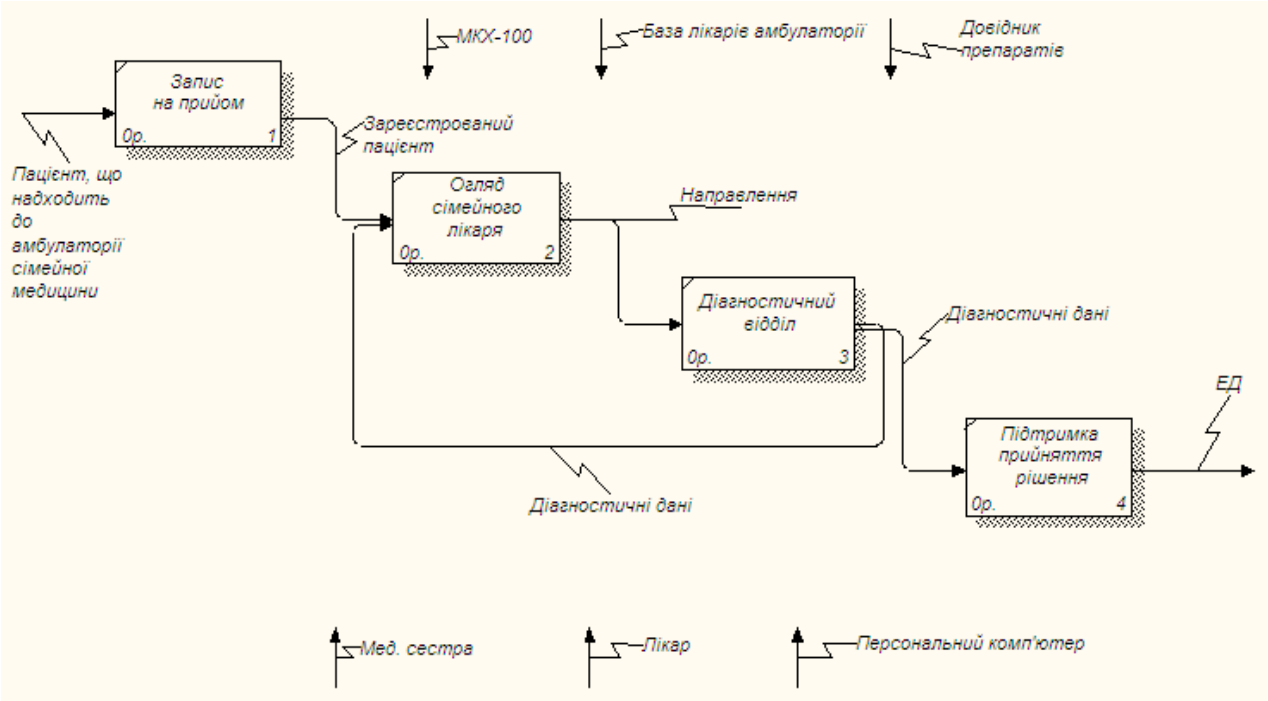



*Рис.* 2.4. Процес декомпозиції

**Побудова відгалужень дуг.** ЗАПИС НА ПРИЙОМ, ОГЛЯД СІМЕЙНИМ ЛІКАРЕМ та ПІДТРИМКА ПРИЙНЯТТЯ РІШЕННЯ здійснюються згідно МКХ-10, тому у керуючої стрілки МКХ-10 з'являться відгалуження: ЗАПИС НА ПРИЙОМ, ОГЛЯД СІМЕЙНИМ ЛІКАРЕМ та ПІДТРИМКА ПРИЙНЯТТЯ РІШЕННЯ. Виберіть інструмент створення дуг .

1. Натисніть мишкою по накінцівнику вхідного потоку МКХ-10.
2. Натисніть по вхідній стороні блоку ЗАПИС НА ПРИЙОМ.
3. Самостійно виконайте відгалуження дуги МКХ-10 на блоки ОГЛЯД СІМЕЙНИМ ЛІКАРЕМ та ПІДТРИМКА ПРИЙНЯТТЯ РІШЕННЯ. Перевірте себе (рис. 2.5).

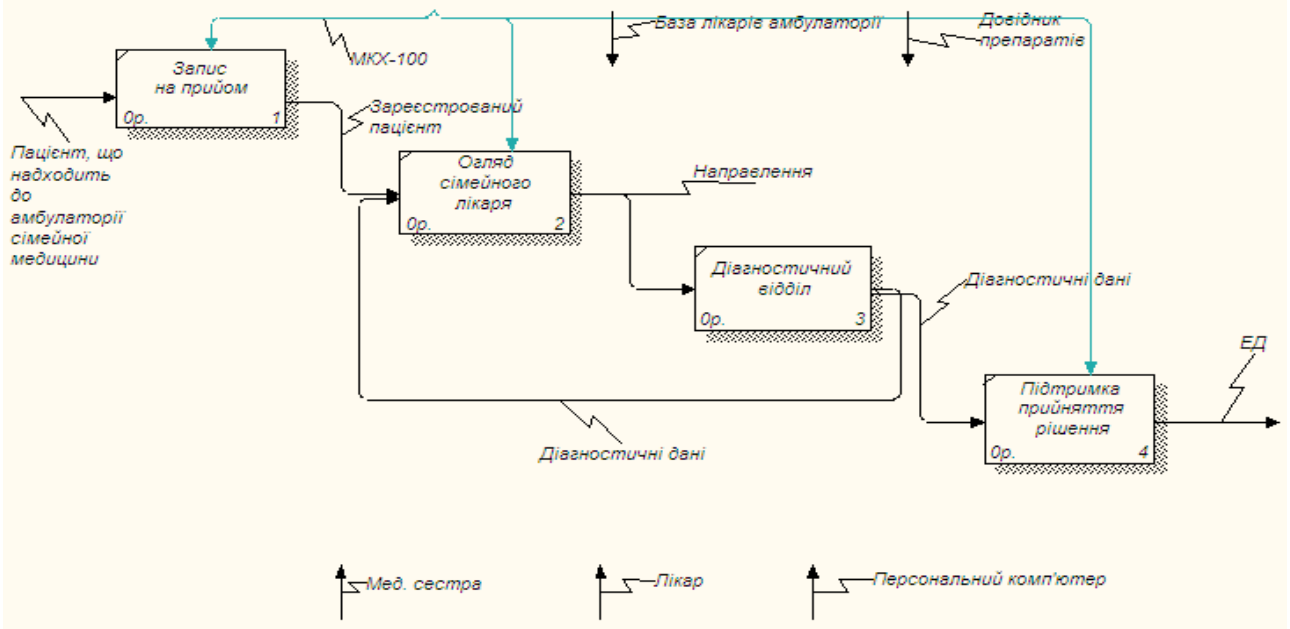



*Рис.* 2.5. Відгалуження дуги МКХ-10

**Побудова дуг Медична сестра, Лікар та Персональний комп'ютер, База лікарів амбулаторії та Довідник препаратів**

Дугу МЕДИЧНА СЕСТРА потрібно з'єднати із ЗАПИСОМ НА ПРИЙОМ та ДІАГНОСТИЧНИМ ВІДДІЛЕННЯМ. Аналогічно з'єднайте інші дуги та перевірте себе (рис. 2.6).

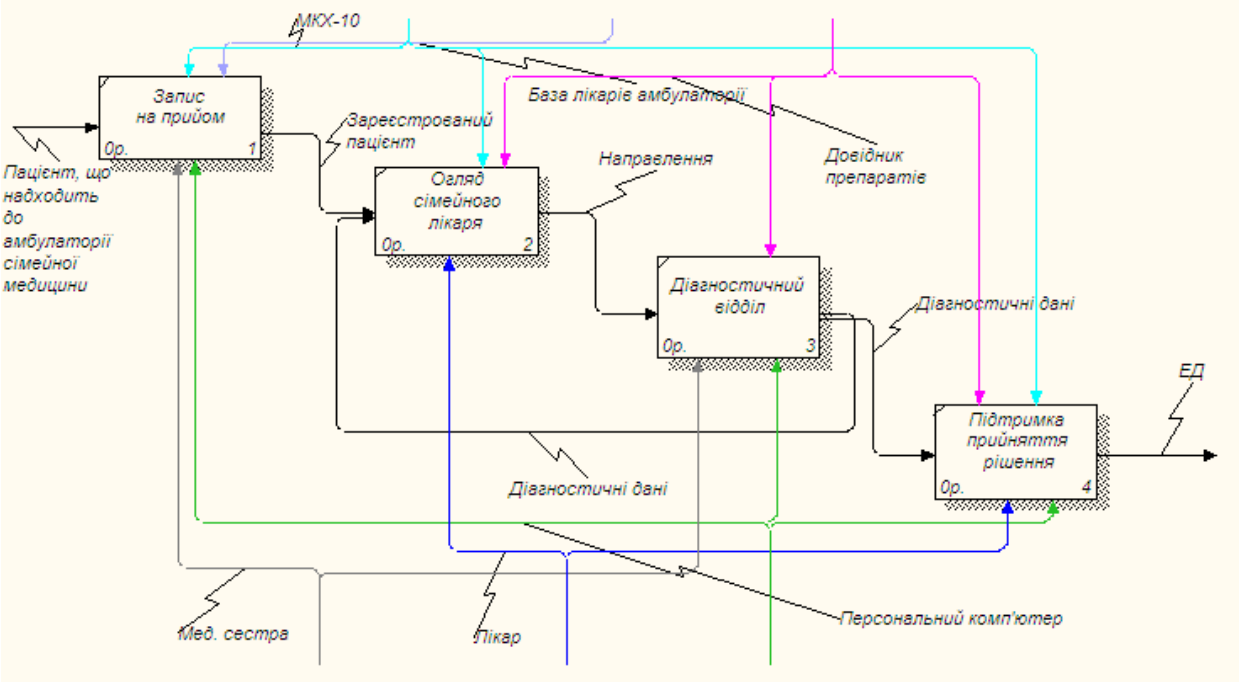



*Рис.* 2.6. Створення решти дуг

**«Тунелювання» стрілок**

В Процесі 1.2. ЗАПИС НА ПРИЙОМ побудуйте нову граничну дугу, якій задайте Вихід – НЕ КОРЕКТНА РЕЄСТРАЦІЯ.

Знову занесенні граничні дуги на діаграмі декомпозиції нижнього рівня зображуються в квадратних дужках і автоматично не з'являються на діаграмі верхнього рівня. Для їх «перетягування» наверх необхідно:

1. Вибрати інструмент редагування  .
2. Натиснути ПКМ по квадратним дужкам.
3. Вибрати в контекстному меню пункт **Arrow Tunnel**.
4. В діалозі, що з'явився **Border Arrow Editor** (рис. 2.7) натиснути по кнопці **Resolve it to border arrow** для міграції стрілки на діаграму верхнього рівня чи по кнопці **Change it to resovled rounded tunnel** для «тунелювання» дуги.

Тунельна дуга зображується з круглими дужками на кінці и не потрапляє на іншу діаграму (рис. 2.8). Таке тунелювання може бути застосовано для зображення малозначимих стрілок. Дугу «Не коректна реєстрація» необхідно відправити в тунель.

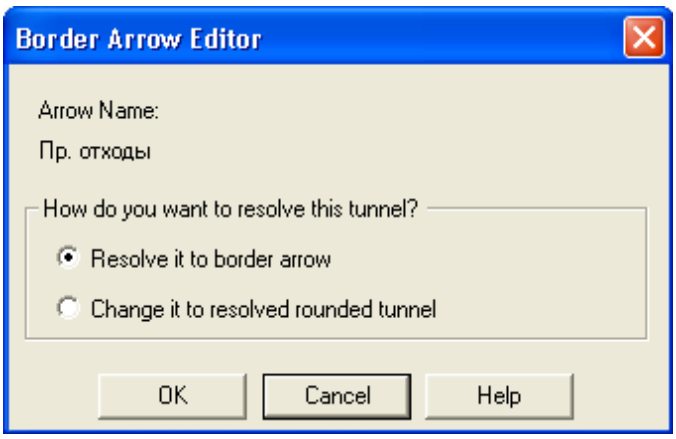



*Рис.* 2.7. Діалог Border Arrow Editor

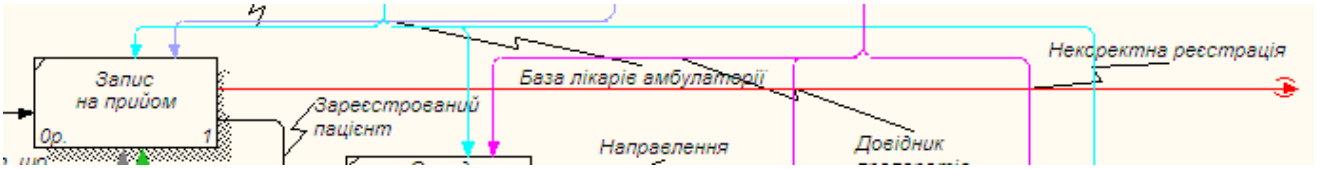



*Рис.* 2.8. Гранична дуга

**Створення зворотного зв'язку керування.** Якість прийнятого рішення можна підвищити шляхом безпосереднього регулювання процесів ОГЛЯД СІМЕЙНИМ ЛІКАРЕМ та ДІАГНОСТИЧНИЙ ВІДДІЛ залежно від результату (виходу) роботи ПІДТРИМКА ПРИЙНЯТТЯ РІШЕННЯ. Зворотний зв'язок керування свідчить про ефективність бізнес-процесу і створюється таким чином:

1. Виберіть інструмент створення дуг .
2. Натисніть мишкою по виходу ПІДТРИМКА ПРИЙНЯТТЯ РІШЕННЯ.
3. Натисніть по керуванню блоків ОГЛЯД СІМЕЙНИМ ЛІКАРЕМ та ДІАГНОСТИЧНИЙ ВІДДІЛ
4. Назвіть зворотній зв'язок ПОВТОРНИЙ ОГЛЯД. Маємо отримати діаграму, зображену на рис. 2.9.
5. В результаті отримаємо робочу область, зображену на рис. 2.10.

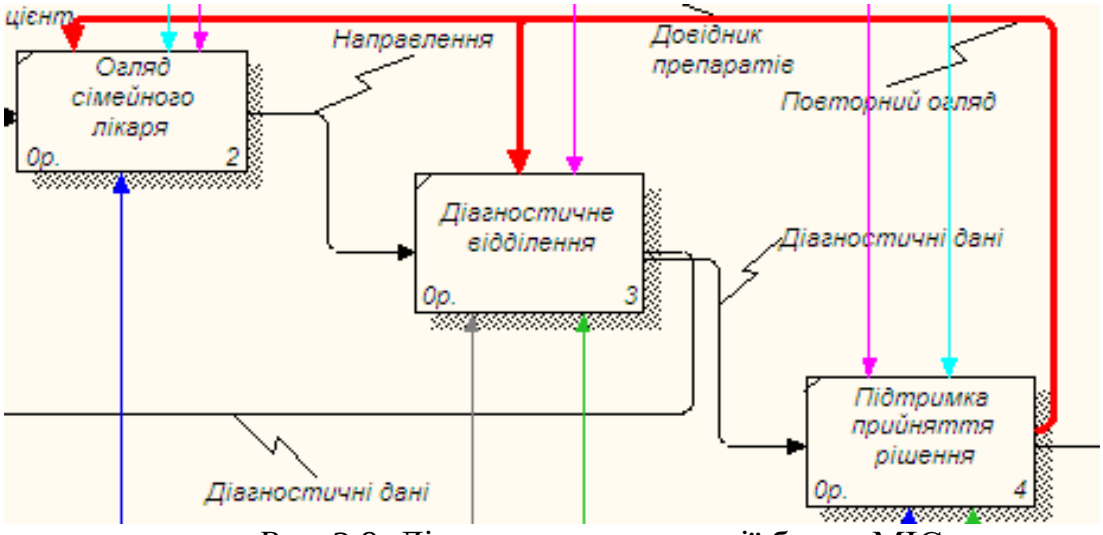



*Рис.* 2.9. Діаграма декомпозиції блоку МІС

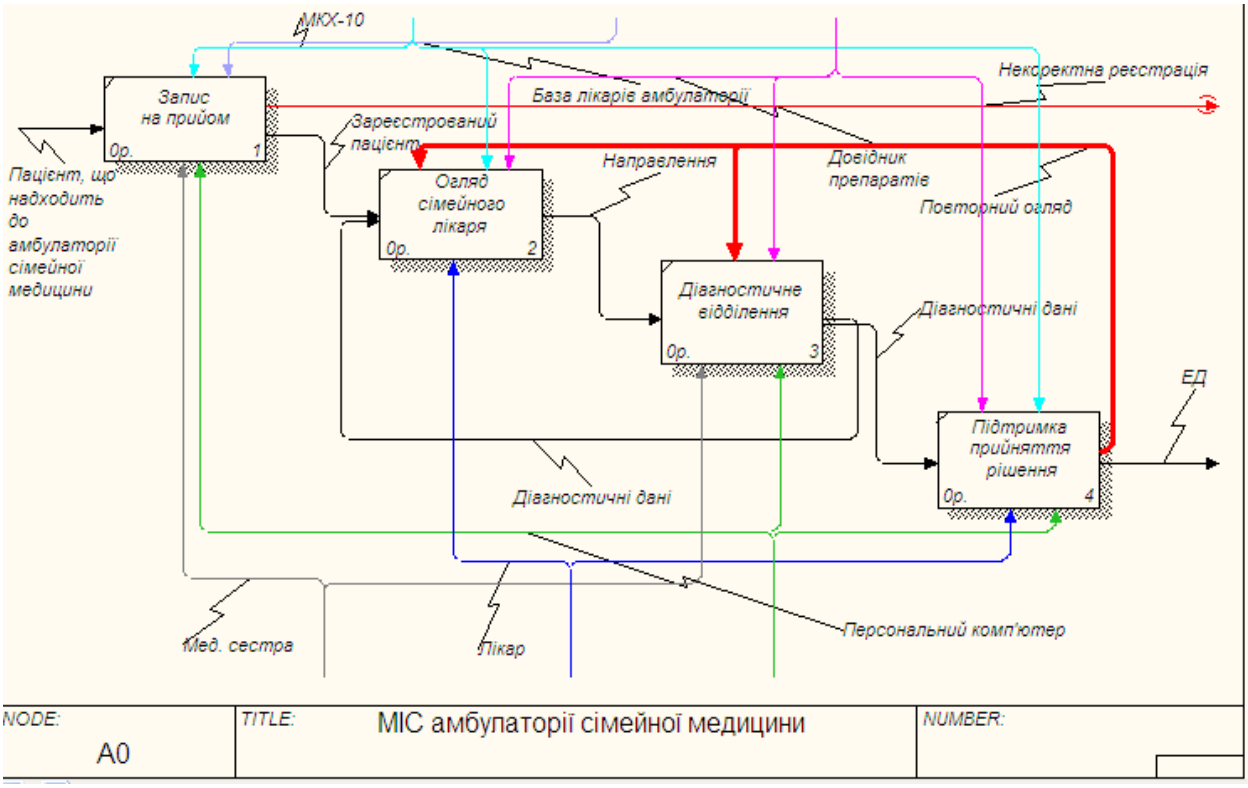



*Рис.* 2.10. Результат створення діаграми декомпозиції A0

## Теоретичні відомості

### 1. Декомпозиція контекстної діаграми

Після опису системи в цілому проводиться розбиття її на великі фрагменти. Цей процес називається *функціональною декомпозицією*, а діаграми, які описують кожен фрагмент і взаємодію фрагментів, називають *діаграмами декомпозиції.*

Після створення контекстної діаграми, яка є описом контексту модельованої системи, проводиться *функціональна декомпозиція*: система розбивається на підсистеми і кожна підсистема описується у тому ж синтаксисі, що і система в цілому. Потім кожна підсистема розбивається на більш дрібні і так до досягнення потрібного рівня деталізації. В результаті такого розбиття, кожен фрагмент системи зображується на окремій діаграмі декомпозиції, яка призначена для деталізації роботи. При декомпозиції процесу всі стрілки, вхідні або вихідні з нього, повинні бути перенесені на діаграму нижнього рівня і використані при її побудові. При цьому заборонені нові стрілки, що виходять за межі нової діаграми, крім спеціальних «тунельованих» стрілок.

*Діаграма верхнього рівня* створюється шляхом декомпозиції основної функції контекстної діаграми. На діаграмі декомпозиції функції нумеруються автоматично зліва направо. Номер функції показується в правому нижньому кутку. В лівому верхньому з'являється невелика діагональна лінія, яка показує, що задана функція була декомпозована. Після декомпозиції контекстної діаграми проводиться декомпозиція кожного великого фрагмента системи на більш дрібні і так далі до досягнення потрібного рівня подробиці опису.

Після кожного сеансу декомпозиції проводяться сеанси експертизи. Експерти предметної області вказують на відповідність реальних бізнес-процесів створеним діаграмам. Знайдені невідповідності виправляються, і тільки після проходження експертизи без зауважень можна приступати до наступного сеансу декомпозиції. Так досягається відповідність моделі реальним бізнес-процесам на будь-якому і кожному рівні моделі. Синтаксис опису системи в цілому і кожного її фрагмента однаковий в усій моделі.

Діаграма декомпозиції призначена для деталізації роботи. На відміну від моделей, що відображають структуру організації, робота на діаграмі верхнього рівня в IDEF0 - це не елемент управління роботами, розташованими нижче. Роботи нижнього рівня - це те ж саме, що роботи верхнього рівня, але в більш детальному викладі.

ICOM (абревіатура від Input, Control, Output і Mechanism) - коди, призначені для ідентифікації граничних стрілок. Код ICOM містить префікс, який відповідає типу стрілки (I, C, O або M), і порядковий номер. BPwin вносить ICOM-коди автоматично. Для відображення ICOM-кодів слід включити опцію «ICOM codes» на закладці Display діалогу Model Properties (меню Model / Model Properties).

Діаграми декомпозиції містять споріднені роботи, тобто дочірні роботи, які мають загальну батьківську роботу. Роботи на діаграмах декомпозиції зазвичай розташовуються по діагоналі від лівого верхнього кута до правого нижнього. Такий порядок називається *порядком домінування*. Згідно з цим принципом розташування в лівому верхньому кутку розміщується

найважливіша робота або робота, виконувана за часом першою. Далі вправо вниз розташовуються менш важливі або роботи, які виконуються пізніше. Таке розміщення полегшує читання діаграм, крім того, на ньому ґрунтується поняття взаємозв'язків робіт.

Кожна з робіт на діаграмі декомпозиції може бути в свою чергу декомпозована. На діаграмі декомпозиції роботи нумеруються автоматично зліва направо. Номер роботи показується в правому нижньому кутку. У лівому верхньому кутку зображується невелика діагональна риса, яка показує, що дана робота не була декомпозована.

## 3. Побудова функціональної моделі

*Мета та завдання практикуму*: 1)навчитись деталізовувати процеси; 2) засвоїти правила опису властивостей моделі; 3) навчитись складати звіт по властивостях моделі.

План

1. Функціональна декомпозиція. Дерево вузлів

**Завдання:**

1. Деталізувати процеси нижнього рівня.
2. Описати властивості моделі.
3. Скласти звіт.
4. Дати відповідь на контрольні питання.
5. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи.**

1.Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту.

2. Продемонструвати виконане завдання викладачу.

**Контрольні запитання:**

1. Як виглядає нумерація моделі в ієрархії IDEF0?
2. Що таке Дерево вузлів?
3. Який процес розробки називають функціональною декомпозицією?
4. Для чого необхідно складати звіт?

### Аудиторна робота

**Приклад 3.1.** Останнім кроком побудови моделі є функціональна декомпозиція. Побудована діаграма верхнього рівня також має множину процесів, які в свою чергу можуть бути деталізовані в діаграмі нижнього рівня. Таким чином будується ієрархія IDEF0 з контекстною діаграмою в вершині ієрархії.Цей процес декомпозиції триває до досягнення потрібного рівня деталізації. При такій побудові ієрархії IDEF0 кожен процес нижчого рівня необхідно співвіднести з процесом верхнього рівня. Зазвичай для цієї мети всі роботи моделі нумеруються.

Всі роботи (дії) моделі нумеруються. Номер складається з префікса і числа. Може бути використаний префікс будь-якої довжини, але зазвичай використовують префікс А. Контекстна (коренева) роботадерева має номер А0. Роботи і декомпозиції А0 мають номери А1, А2, А3 і т. д. Роботи декомпозиції нижнього рівня мають номер батьківської роботи і черговий порядковий номер, наприклад роботи декомпозиції А3 матимуть номери А31, А32, А33, А34 і т. д.

Роботи утворюють ієрархію, де кожна робота може мати одну батьківську і кілька дочірніх робіт, утворюючи дерево. Таке дерево називають *деревом вузлів*, а вищеописану нумерацію - нумерацією по вузлах.

Існують різні варіанти нумерації, які можна налаштувати у вкладці Numbering (рис. 3.1.) Діалогу Model Properties (меню Model - Model Properties).

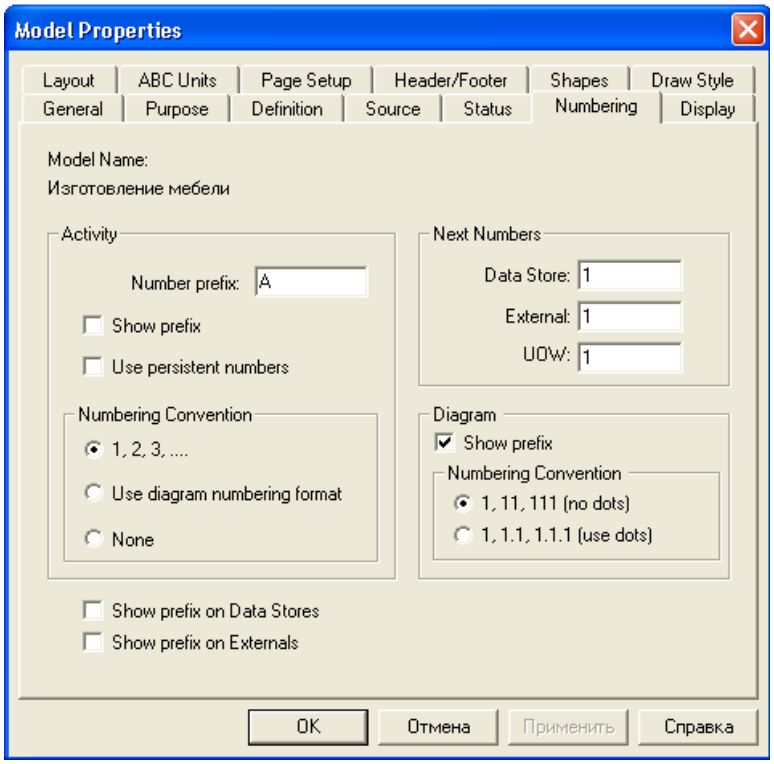



*Рис. 3.1.* Діалогове вікно налаштування нумерації робіт в діаграмі

Діаграми IDEF0 мають подвійну нумерацію. По-перше, діаграми мають номер по вузлу. Контекстна діаграма завжди має номер A-0, декомпозиція контекстної діаграми - номер A0, інші діаграми декомпозиції – номери, що відповідають вузлу (наприклад, A1, A2, A21, A213 і т. д.). BPwin автоматично підтримує нумерацію за вузлами, тобто при проведенні декомпозиції створюється нова діаграма та їй автоматично присвоюють відповідний номер.

У результаті проведення експертизи діаграми можуть уточнюватися і змінюватися, отже, можуть бути створено різні версії однієї і тієї ж (з точки зору її розташування в дереві вузлів) діаграми декомпозиції.

BPwin дозволяє мати в моделі тільки одну діаграму декомпозиції в даному вузлі. Старі версії діаграми можна зберігати у вигляді паперової копії або як **FEO-діаграму** (при створенні FEO-діаграм відсутня можливість повернення, можна отримати з діаграми декомпозиції FEO, але не навпаки).

Слід відрізняти різні версії однієї і тієї ж діаграми. Для цього існує спеціальний номер - C-number, який повинен присвоюватися автором моделі вручну. C-number - це довільний рядок, але рекомендується дотримуватися стандарту, коли номер складається з літерного префікса і порядкового номера, причому в якості префікса використовуються ініціали автора діаграми, а порядковий номер відстежується автором вручну, наприклад **КСТ001** (рис. 3.2.).

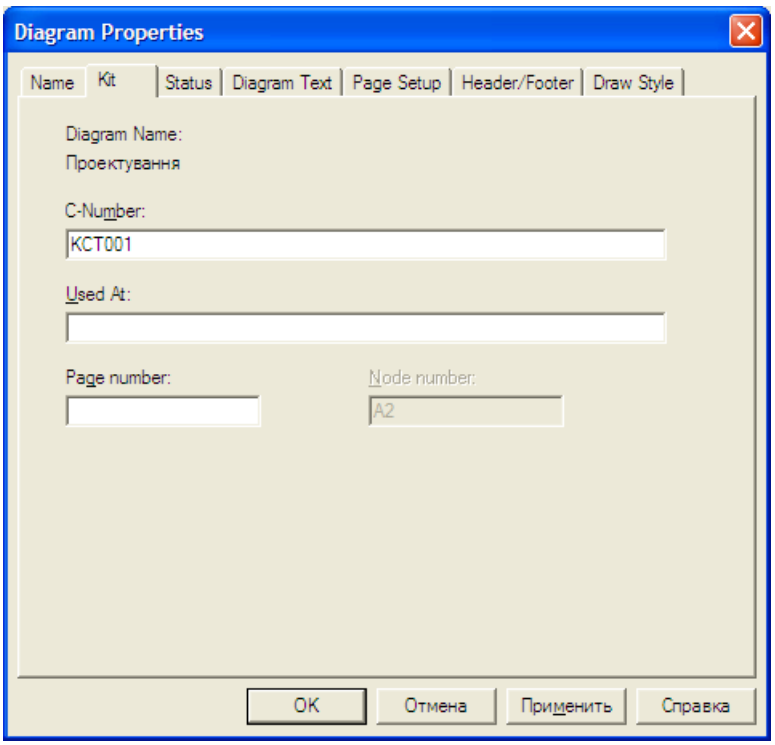



*Рис. 3.2.* Діалогове вікно присвоювання номеру даній версії діаграми

На попередніх лабораторних роботах ми побудували контекстну діаграму процесу "МІС амбулаторії сімейної медицини" і провели її деталізацію за допомогою діаграми верхнього рівня. Останнім кроком побудови моделі є **функціональна декомпозиція**, тобто розбиття складних процесів на більш прості. Цей процес декомпозиції триває до досягнення потрібного рівня деталізації.

**Деталізація процесу «Запис на прийом».**

1. Проведіть деталізацію процесу **1.2. ЗАПИС НА ПРИЙОМ** за допомогою діаграми нижнього рівня. Дані представлені в табл. 3.1.

Таблиця 3.1

*Деталізація процесу «Запис на прийом»*

| Процес | Вхід | Вихід |
|---|---|---|
| 1.1.1 – Перевірка попереднього надходження | Пацієнт, що надходить до амбулаторії сімейної медицини | Відсутність попередніх записів; Дані попереднього перебування |
| 1.1.2 – Реєстрація нового пацієнта | Відсутність попередніх записів | Дані про пацієнта |
| 1.1.3 – Запис пацієнта на прийом | Дані про пацієнта; Дані попереднього перебування | Зареєстрований пацієнт |
| Керуючі стрілки і стрілки механізмів, зазначені на діаграмі верхнього рівня повинні бути і в діаграмі деталізації. | | |

2. Виберіть інструмент  і клацніть по блоку ЗАПИС НА ПРИЙОМ.

3. У діалоговому вікні введіть число, на яке буде проведена декомпозиція - 3. Вкажіть тип діаграми IDEF0 (рис. 3.3) і натисніть OK. Ви отримаєте діаграму декомпозиції рівня **A2** (рис. 3.4.).

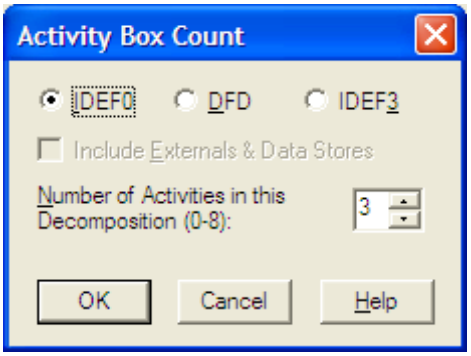



*Рис.* 3.3. Діалогове вікно декомпозиції блоку

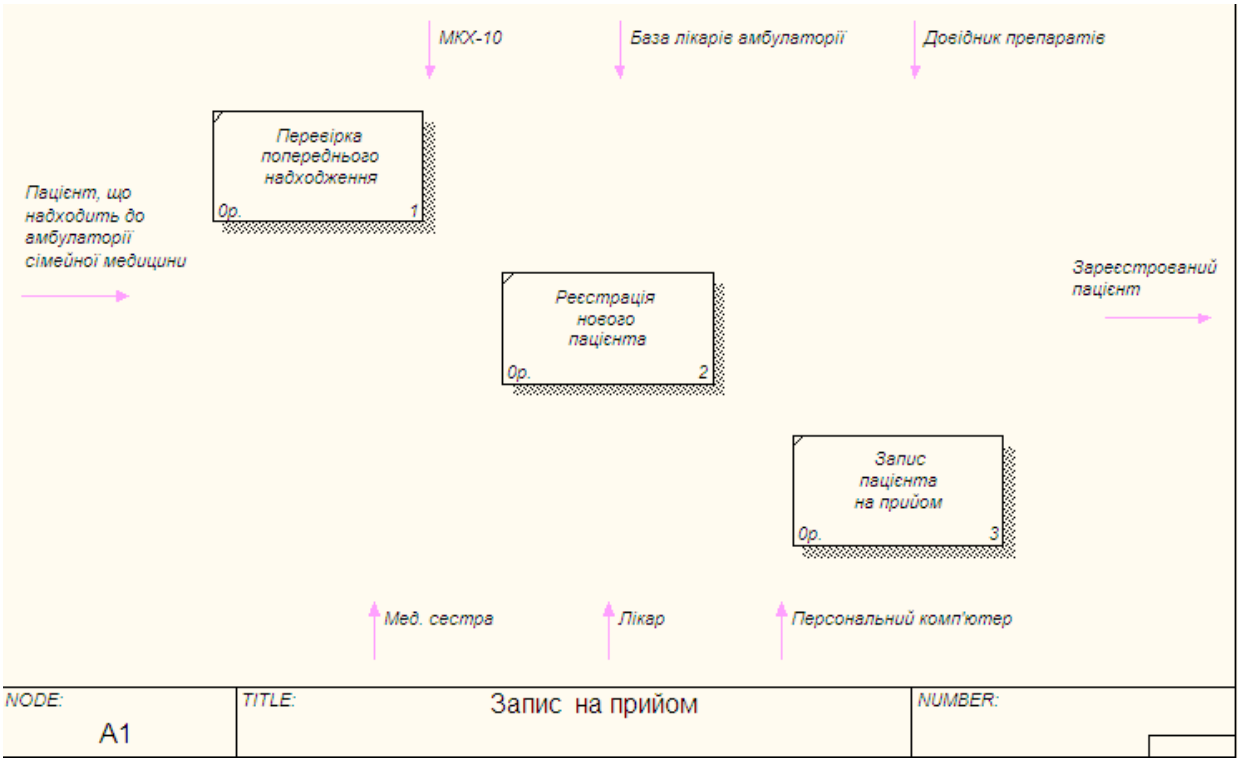



*Рис*. 3.4. Декомпозиція рівня A2

4. Необхідно зазначити назви процесів.

5. Поєднати дугами позначені процеси, використовуючи дані з табл. 3.1. Маємо отримати результат у вигляді рис. 3.5.

**Опис властивостей моделі**

**IDEF0-модель** передбачає наявність чітко сформульованої мети, єдиного суб'єкта моделювання і однієї точки зору. Для внесення області, мети і точки зору в моделі IDEF0 слід:

1. Вибрати пункт меню **Model - Model Properties**, що викликає діалог **Model Properties** (рис. 3.6).

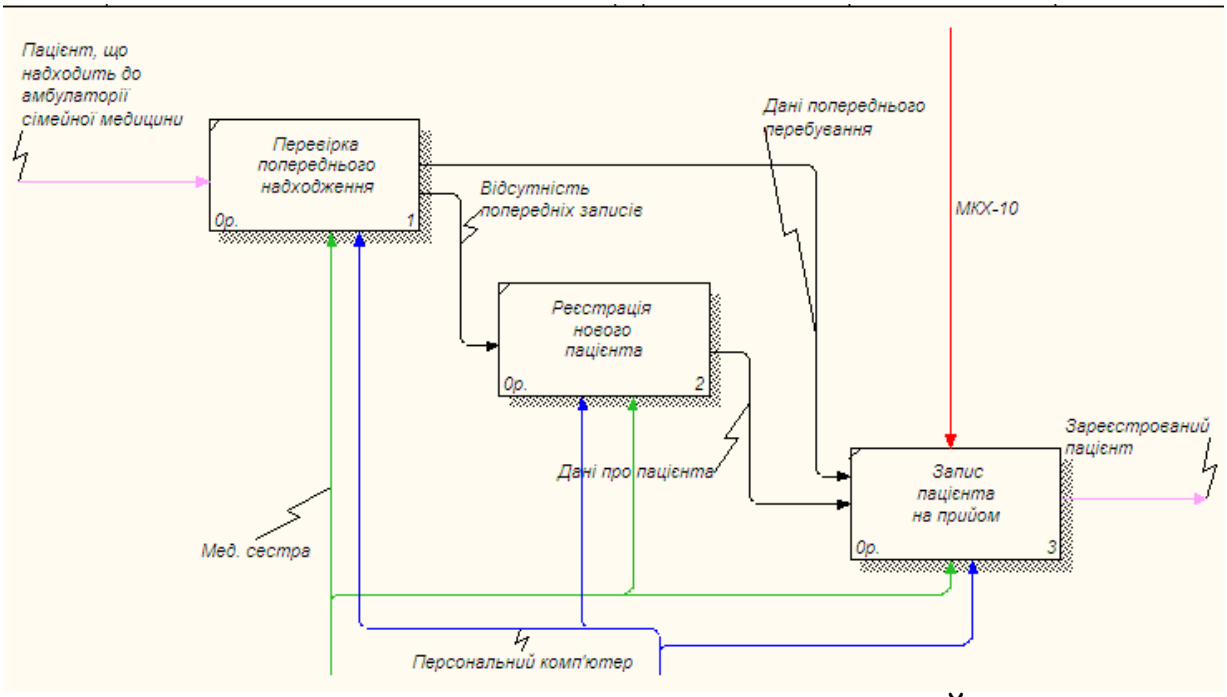



*Рис. 3.5.* Деталізація процесу ЗАПИС НА ПРИЙОМ

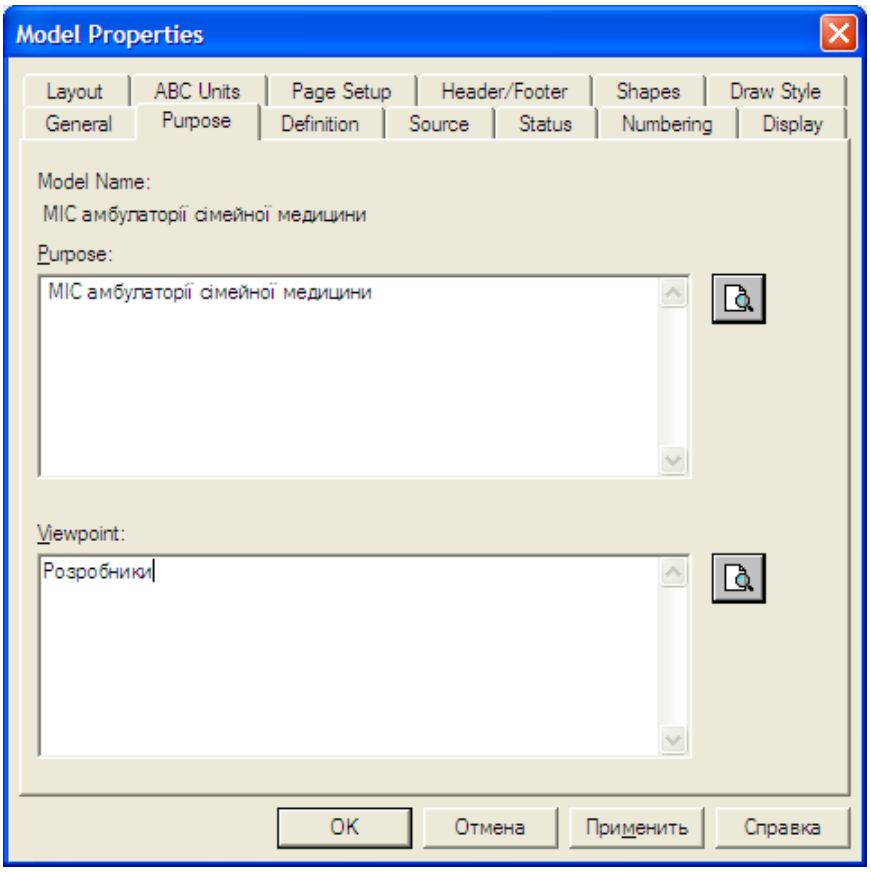



*Рис. 3.6.* Діалог завдання властивостей моделі

2. Під вкладку **Purpose** внести мету і точку зору, а у вкладку **Definition** - визначення моделі.

Мету і точку зору прийнято виносити на контекстну діаграму А-0 у вигляді текстового блоку. Після опису вони з'являться на контекстній діаграмі у вигляді текстового блоку. Опис проводиться на рівні контекстної діаграми.

Для описання мети і точки зору необхідно:

1. Перейти на рівень діаграми **A-0**.
2. Вибрати кнопку тексту **T** на палітрі інструментів.
3. Клацнути мишею в позиції передбачуваного введення тексту.
4. У діалоговому вікні набрати потрібний текст і встановити опцію значущості (звичайний текст, мета або точка зору) (рис. 3.7).

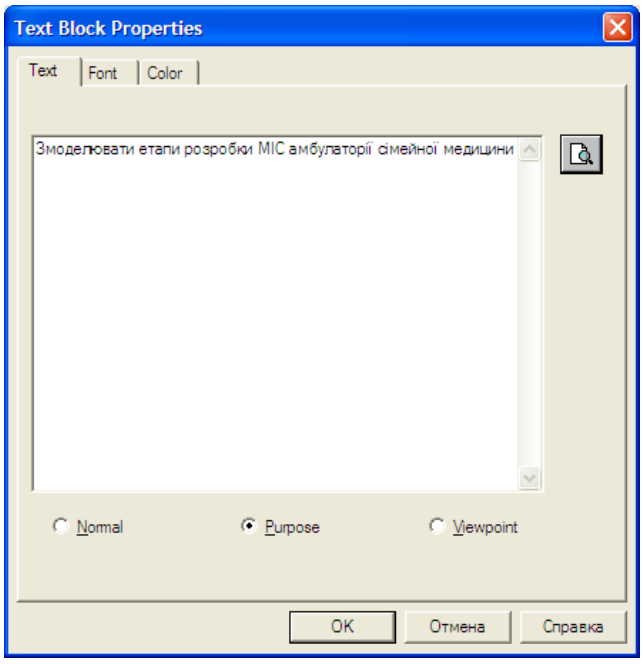



*Рис.* 3.7. Встановлення опції Text

5. У вкладці **Status** того ж діалогу можна описати статус моделі (чорновий варіант, робочий, остаточний і т.д.), час створення і останнього редагування (відстежується надалі автоматично з системної датою).

6. У вкладці **Source** можна вказати джерела інформації для побудови моделі (наприклад, «Опитування експертів предметної області і аналіз документації»).

7. Вкладка **General** служить для внесення імені проєкту та моделі, імені та ініціалів учасника і тимчасових рамок моделі.

**Складання звіту.** Результат опису моделі можна отримати в звіті **Model Report**.

1. Діалогове вікно налаштування звіту по моделі викличте з пункту меню **Tools – Reports - Model Report**.
2. Виберіть необхідні поля, при цьому автоматично відображається черговість виведення інформації у звіті (рис. 3.8).
3. Натисніть на кнопку **Preview**, щоб переглянути звіт (рис. 3.9).

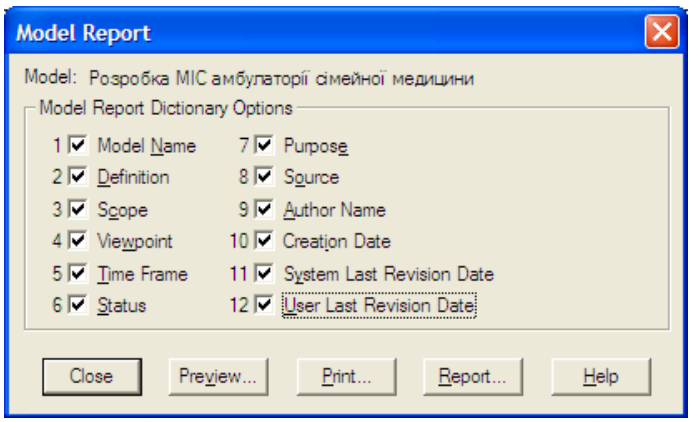



*Рис.* 3.8. Діалогове вікно вибору інформації для звіту

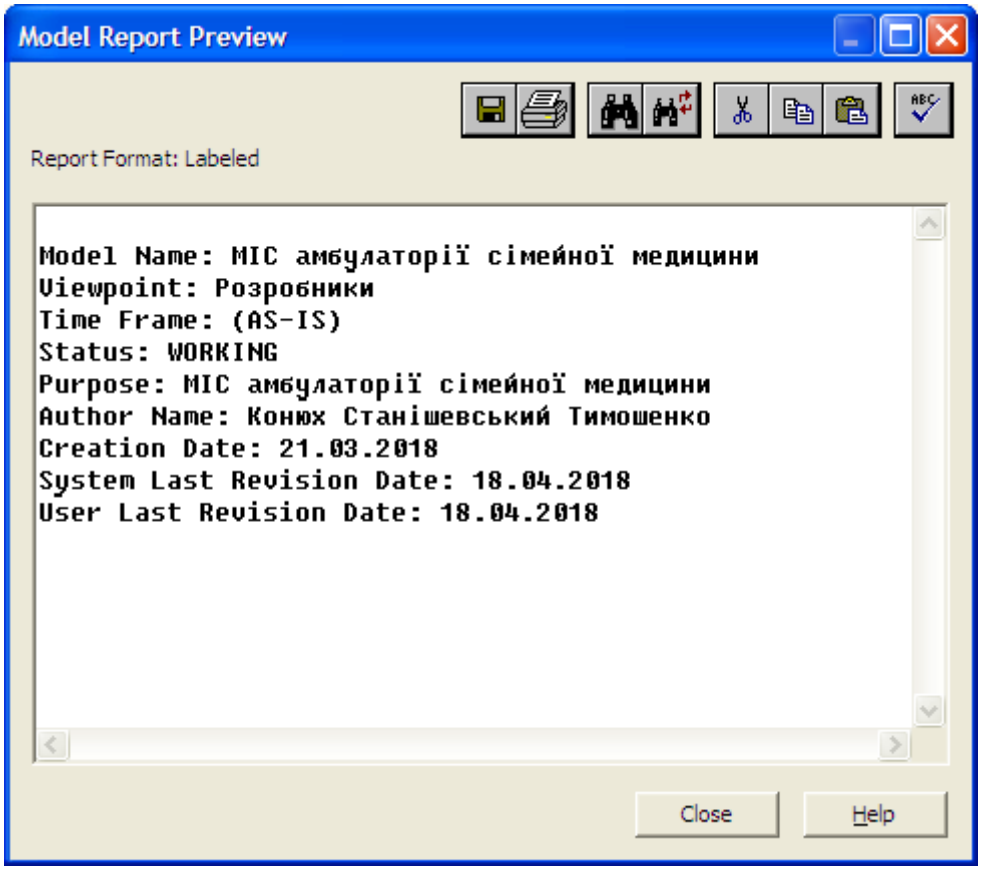



*Рис. 3.9.* Звіт про модель

## 4. Побудова діаграми дерева вузлів і діаграми FEO

*Мета та завдання практикуму:*1) освоїти принципи побудови діаграми дерева вузлів; 2) навчитись редагувати діаграму вузлів; 3) освоїти правила побудови діаграми FEO.

План

1. Діаграма дерева вузлів. Діаграма FEO.

**Завдання:**

1. Створити діаграму дерева вузлів різними способами.
2. Створити діаграму FEO.
3. Дати відповідь на контрольні питання.
4. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи.**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту.
2. Продемонструвати виконане завдання викладачу.

**Контрольні запитання:**

1. Для чого використовують діаграму дерева вузлів**?**
2. Які властивості та стиль може мати діаграма дерева вузлів?
3. Для чого використовують діаграму FEO?
4. Чим відрізняються діаграми дерева вузлів та FEO?

**Аудиторна робота**

**Приклад. 4.1**. Діаграма дерева вузлів показує ієрархію робіт у моделі і дозволяє розглянути всю модель вцілому, але не показує взаємозв'язок між роботами (рис. 4.1). Процес створення моделі робіт є ітераційним (повторюваним, багаторазово змінюваним), отже, роботи можуть змінювати своє розташування в дереві вузлів багаторазово. Щоб не заплутатися і перевірити спосіб декомпозиції, слід після кожної зміни створювати діаграму дерева вузлів. BPwin має потужний інструмент навігації по моделі - **Model Explorer** (рис. 4.2), який дозволяє зобразити ієрархію робіт і діаграм в компактному вигляді (проте цей інструмент не є складовою стандарту IDEF0).

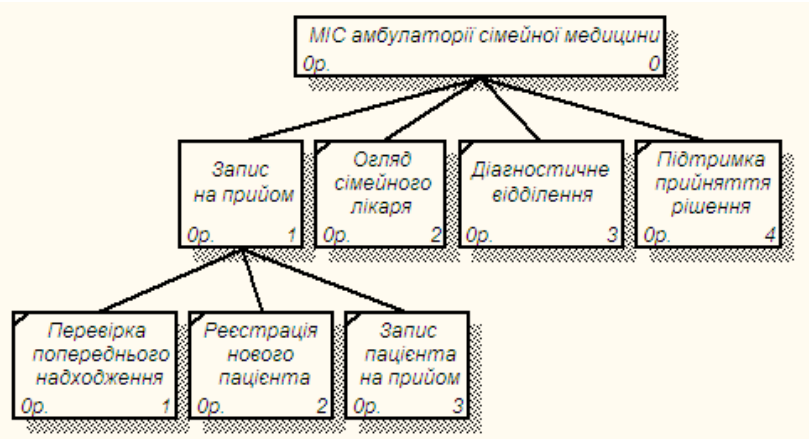



*Рис. 4.1.* Діаграма дерева вузлів

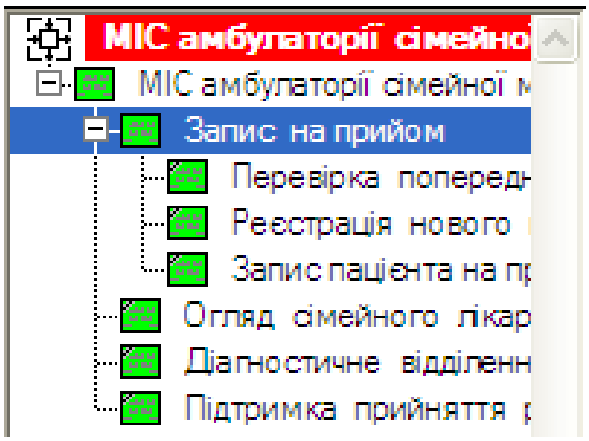



*Рис.* 4.2. Навігатор моделі Model Explorer

**Створення діаграми дерева вузлів.** Для створення діаграми дерева вузлів необхідно:

1. Вибрати в меню пункт **Diagram - Add Node Tree**.

З'явиться діалог створення діаграми дерева вузлів **Node Tree Wizard** (рис. 4.3).

2. У першому діалозі експерта введіть ім'я діаграми дерева вузлів, вузол верхнього рівня і глибину дерева - **Number of Levels** (за замовчуванням 3).

Дерево вузлів не обов'язково в якості верхнього рівня повинно мати контекстну роботу, воно може мати довільну глибину. В одній моделі можна створювати множину діаграм дерева вузлів.Ім'я дерева вузлів за замовчуванням збігається з ім'ям роботи верхнього рівня, а номер діаграми автоматично генерується як номер вузла верхнього рівня плюс літера «N», наприклад, A0N.

Другий діалог експерта **Node Tree Wizard** (рис. 4.4) дозволяє задати властивості діаграми дерева вузлів.

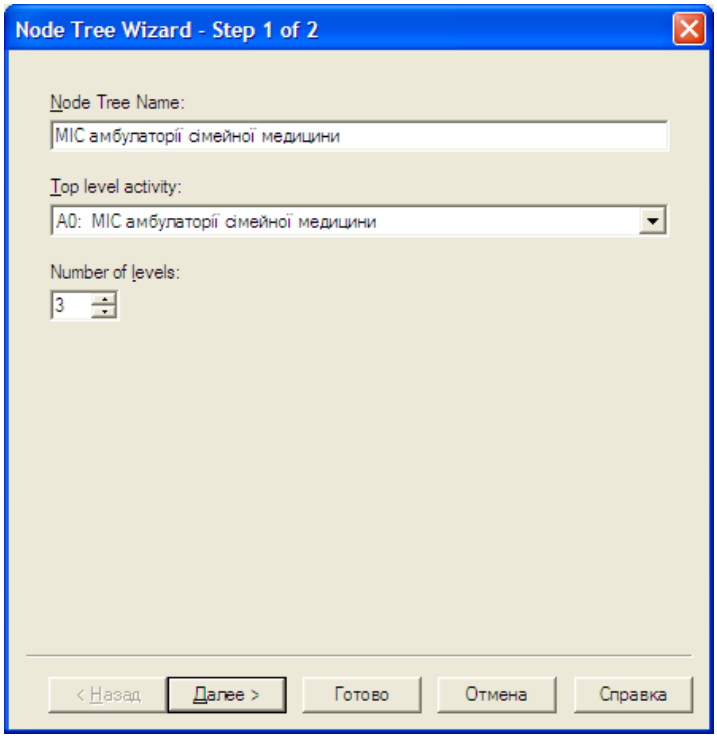



*Рис.* 4.3. Діалог створення діаграми дерева вузлів Node Tree Wizard

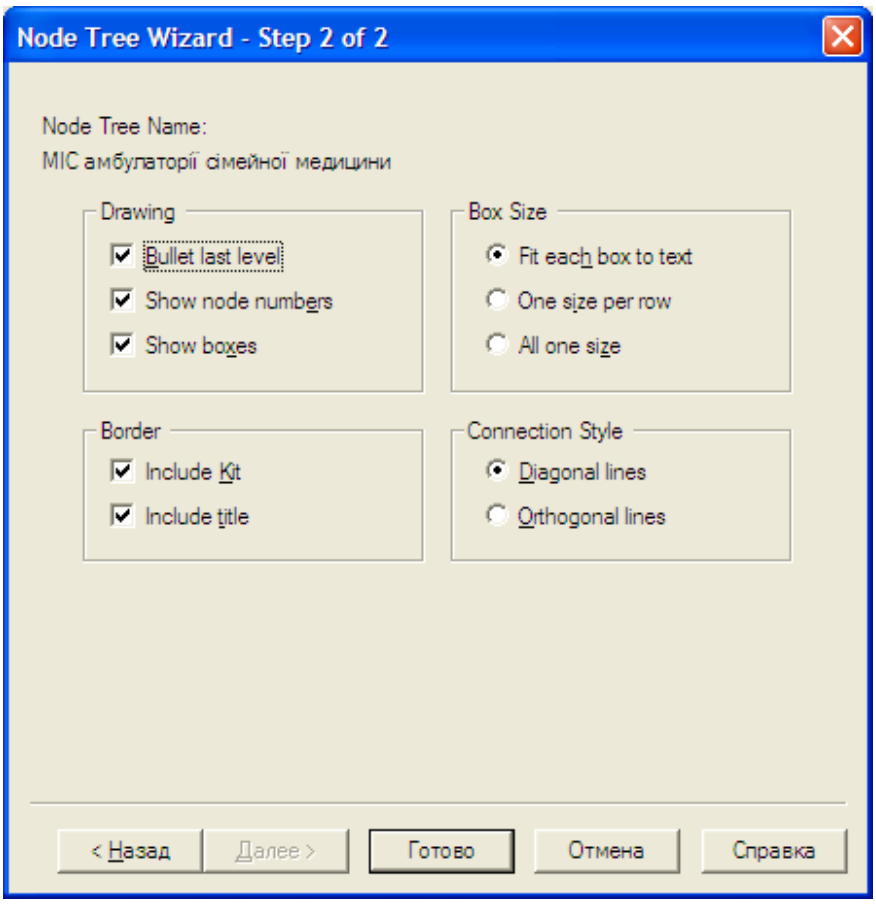



*Рис.* 4.4. Діалог експерта Node Tree Wizard

За замовчуванням нижній рівень декомпозиції показується у вигляді списку, решта робіт - у вигляді прямокутників (рис. 4.5).

Для відображення всього дерева у вигляді прямокутників слід прибрати опцію **Bullet Last Level**. Група **Connection Style** дозволяє вибрати стиль з'єднувальних ліній - діагональні (за замовчуванням) або ортогональні.

3. Клацніть правою кнопкою миші по вільному місцю, ще не зайнятому об'єктами, виберіть меню **Node tree Diagram Properties** (рис. 4.6).

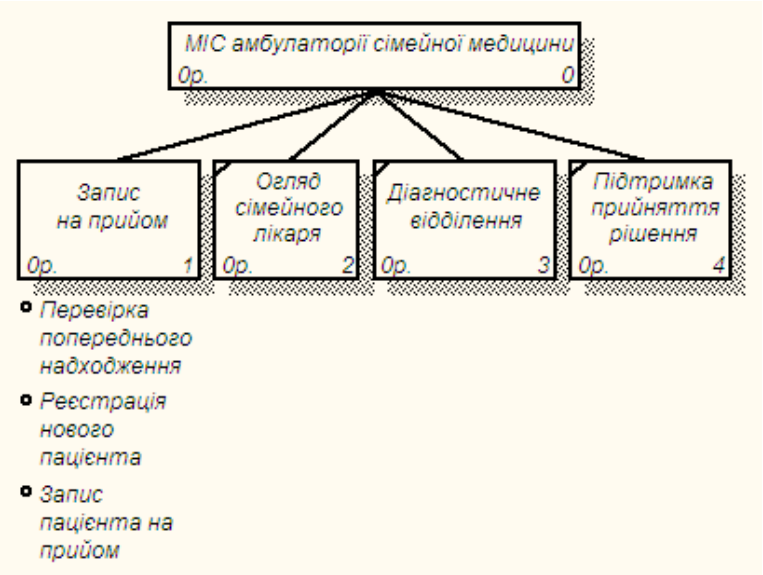



*Рис.* 4.5. Дерево вузлів з діагональними лініями

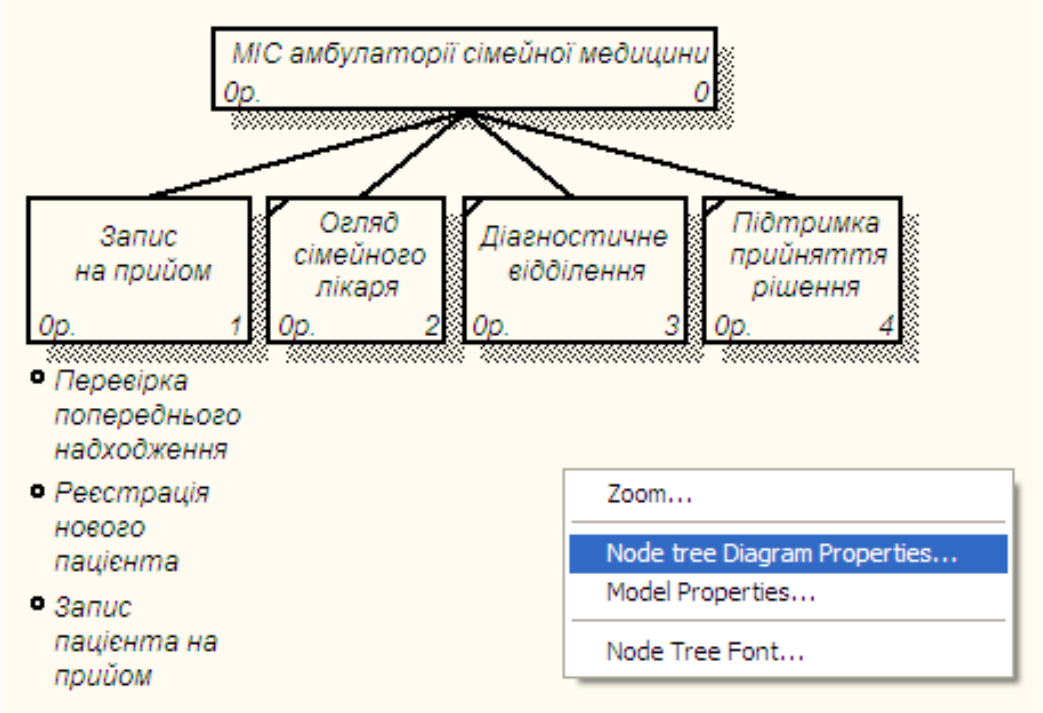



*Рис.* 4.6. Вибір меню Node tree Diagram Properties

4. У вкладці **Style** діалогу **Node Tree Properties** відключіть опцію **Bullet Last Level** (рис. 4.7). Клацніть по кнопці **ОК**. Маємо отримати (рис. 4.8).
5. Створимо діаграму Дерево вузлів з ортогональними лініями (рис. 4.9).

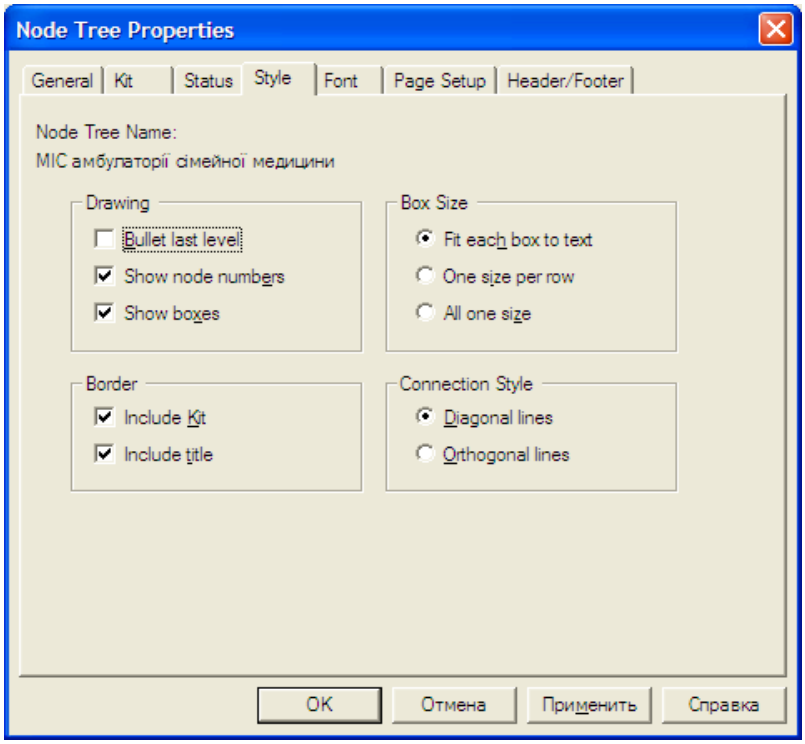



*Рис.* 4.7. Діалогове вікно Node Tree Properties

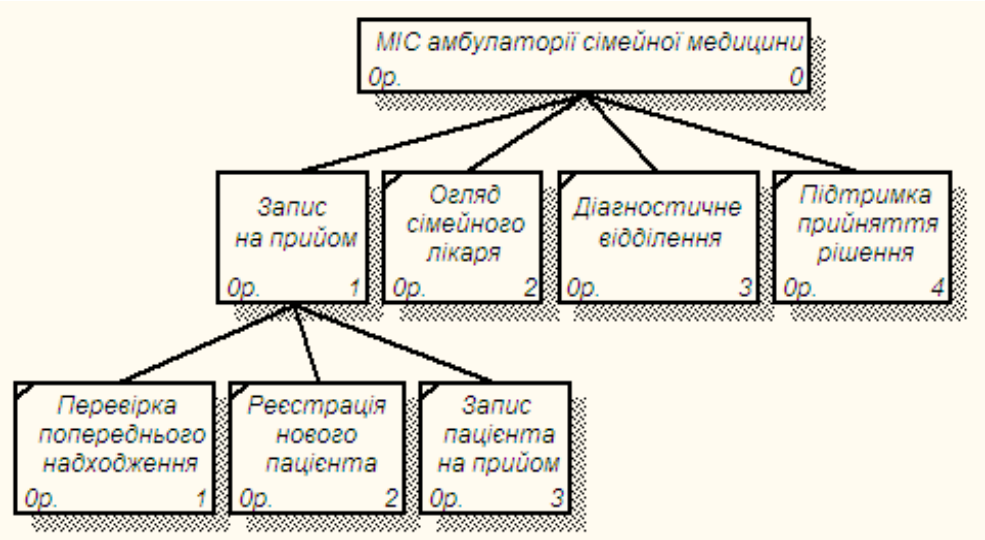



*Рис*. 4.8. Дерево вузлів

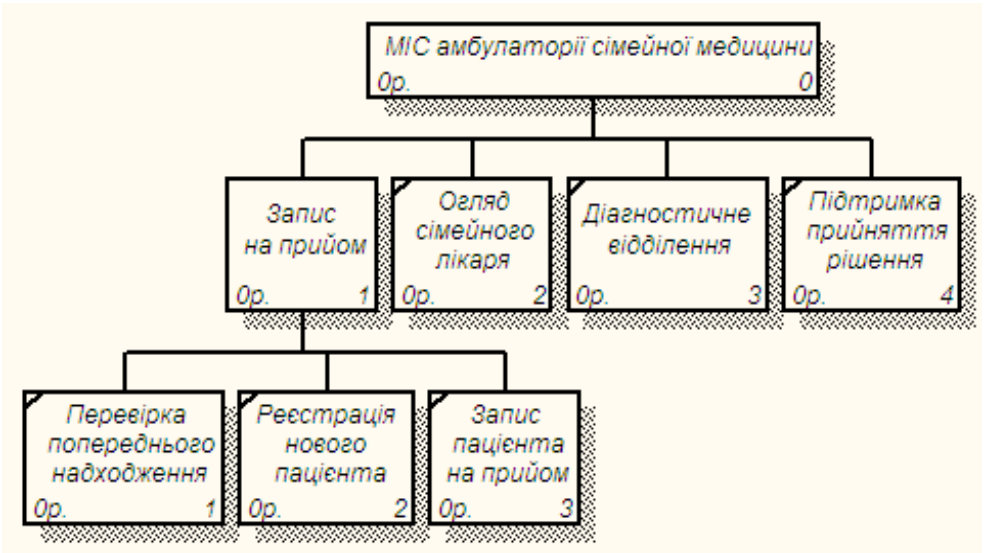



*Рис.* 4.9. Дерево вузлів до роботи «МІС амбулаторії сімейної медицини»

**Створення діаграми FEO.**Діаграми для експозиції (FEO) будуються для ілюстрації окремих фрагментів моделі, альтернативної точки зору, або для спеціальних цілей, які не підтримуються явно синтаксисом IDEF0. Вони по суті є просто картинками - копіями стандартних діаграм і не включаються в аналіз синтаксису.

Створимо діаграму FEO.

1. Виберіть пункт меню **Diagram - Add FEO Diagram**.
2. Вкажіть ім'я діаграми FEO і тип батьківської діаграми (рис. 4.10).
3. Натисніть на кнопку **ОК**.

Отримаємо зображення контекстної діаграми «МІС амбулаторії сімейної медицини» (рис. 4.11).

4. Створимо діаграму FEO для діаграми декомпозиції «Запис на прийом».
5. Маємо отримати (рис. 4.12).

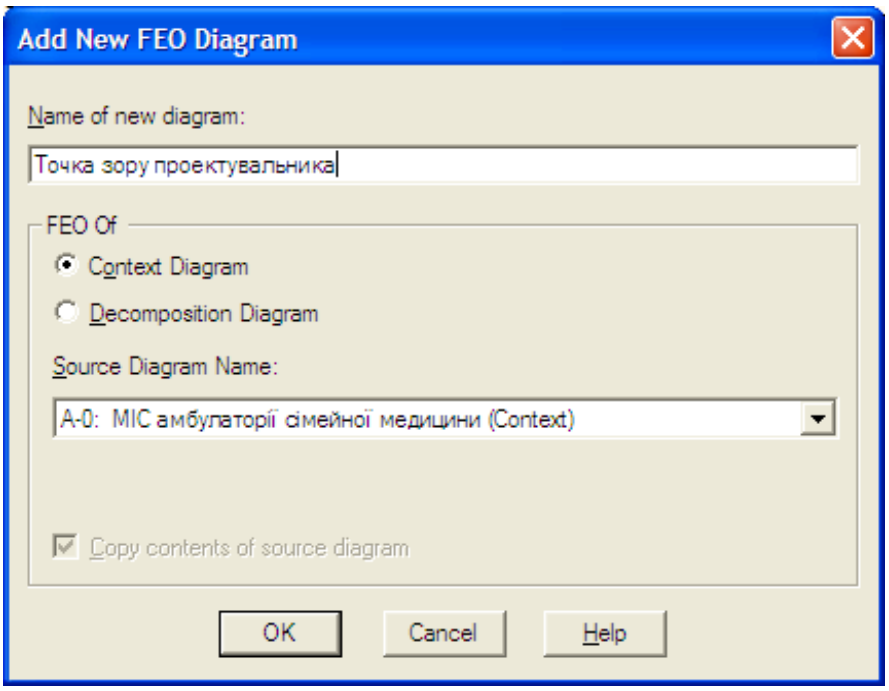



*Рис.* 4.10. Діалогове вікно Add New FEO Diagram

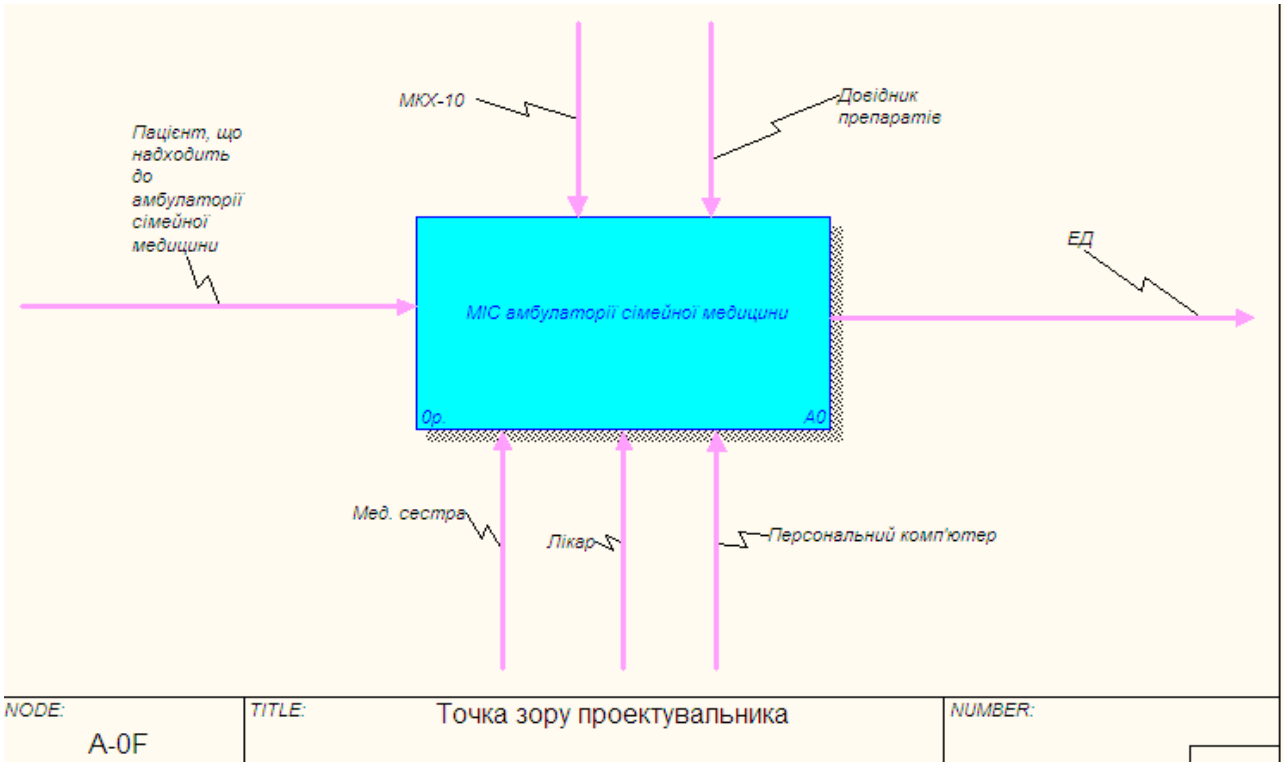



*Рис.* 4.11. Ілюстрація до контекстної діаграмі «МІС амбулаторії сімейної медицини»

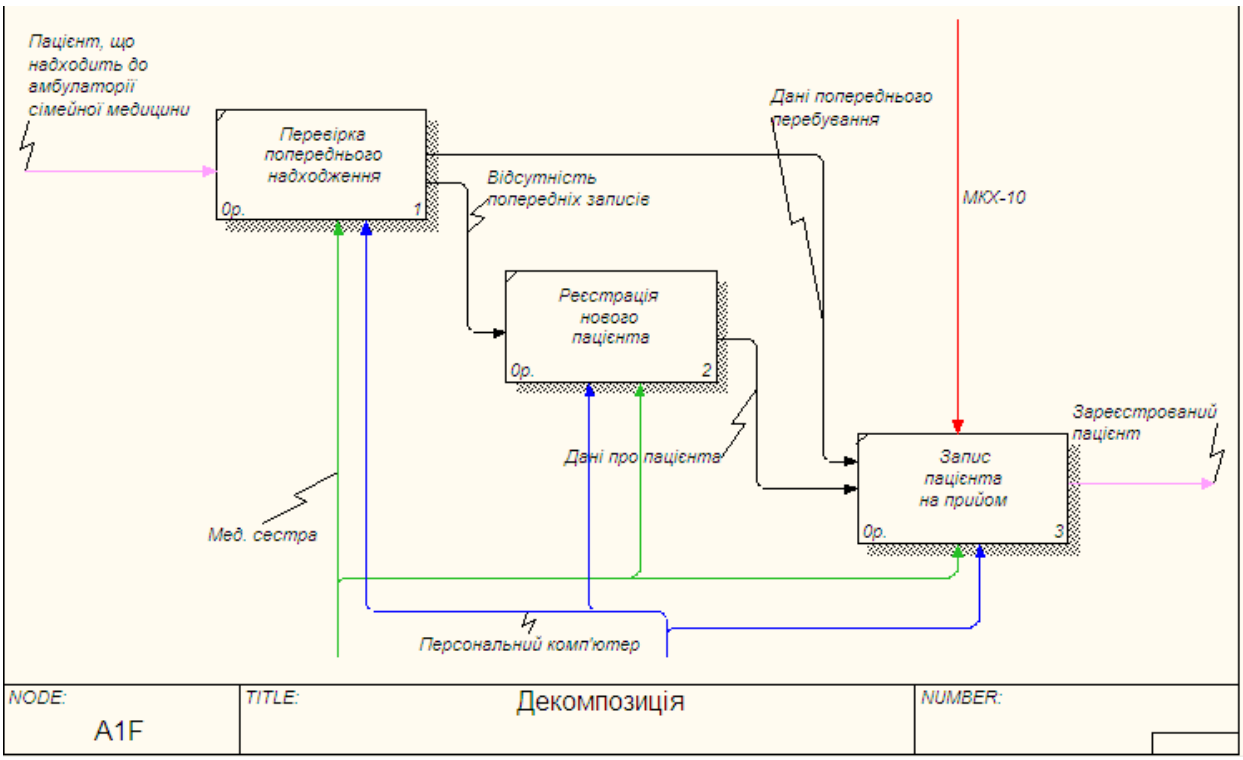



*Рис. 4.12.* Ілюстрація до діаграми декомпозиції «Запис на прийом»

## *5.* Побудова діаграми потоків даних DFD

*Мета та завдання практикуму:* 1) освоїти призначення та принципи побудови DFD– діаграми потоків даних; 2) навчитись встановлювати внутрішні сутності; 3) навчитись проводити деталізацію діаграми DFD.

План

1. Функціональна методика потоків даних:побудова моделі системи у вигляді діаграми потоків даних (Data Flow Diagram - DFD),

### Завдання

1. Створити діаграму DFD.
2. Додати зовнішні сутності (зовнішні посилання).
3. Створити внутрішні сутності.
4. Дати відповіді на контрольні питання.
5. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту.

2. Продемонструвати виконане завдання викладачу.

### Контрольні запитання:

1. Що собою являє DFD – діаграма?
2. Для чого використовують діаграми потоків даних?
3. В чому полягає різниця між DFD та IDEF0?
4. Для чого призначені зовнішні сутності, потоки даних та сховища даних?

### Аудиторна робота

**Приклад. 5.1**. Діаграми потоків даних (Data flow diagramming, DFD) можна використовувати як доповнення до моделі IDEF0 для більш наочного відображення поточних операцій документообігу в системах обробки інформації. Їх використовують для опису документообігу та обробки інформації. DFD зображують собою модель системи як мережу пов'язаних між собою робіт.

Діаграми потоків даних (DFD) показують зовнішні джерела та приймачі даних, потоки даних і сховища (накопичувачі) даних, до яких здійснюється доступ. DFD описує: 1) функції обробки інформації (роботи); 2) документи (стрілки, arrows), об'єкти, співробітників або відділи, які беруть участь в обробці інформації; 3) зовнішні посилання (external references), які забезпечують інтерфейс з зовнішніми об'єктами, що знаходяться за межами модельованої системи; 4) таблиці для зберігання документів (сховища даних, data store). Для зображення діаграм потоків даних використовується нотація Гейна-Сарсона (рис. 5.1.).

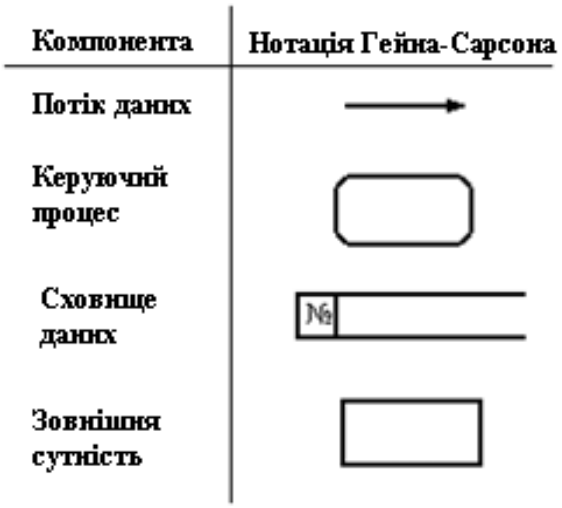



*Рис. 5.1.* Нотація Гейна-Сарсона

Етапи побудови моделі:

1. Виділення зовнішніх об'єктів, з якими система повинна бути пов'язана.
2. Формування DFD діаграми першого рівня.
3. Функціональна декомпозиція кожного підпроцесу за допомогою діаграм нижнього рівня.
4. Складання словника даних.
5. Побудова специфікацій процесу, якщо його не можна виразити комбінацією підпроцесів.

При доповненні моделі **IDEF0** діаграмою **DFD**, в палітрі інструментів на новій діаграмі DFD з'являються нові кнопки:

 - Додати в діаграму зовнішнє посилання (External Reference). Зовнішнє посилання є джерелом або приймачем даних ззовні моделі.

 - Додати в діаграму сховище даних (Data store). Сховище даних дозволяє описати дані, які необхідно зберегти в пам'яті перш, ніж використовувати в роботах.

На відміну від IDEF0, де система розглядається як взаємопов'язані роботи, DFD розглядає систему як сукупність предметів.

*Роботи* в DFD являють собою функції системи, що перетворюють входи у виходи. Хоча роботи зображуються прямокутниками з округленими кутами, сенс їх збігається зі змістом робіт в IDEF0, вони мають входи і виходи, але не підтримують управління та механізми, як IDEF0 (рис. 5.2).

*Зовнішні сутності* зображують входи в систему і / або виходи з системи. Зовнішні сутності зображуються у вигляді прямокутника з тінню і зазвичай розташовуються по краях діаграми (рис. 5.3).

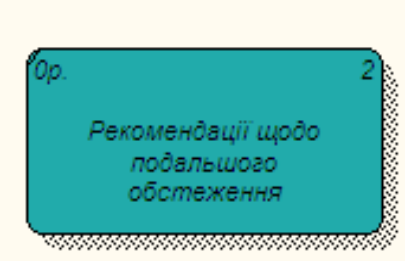



*Рис.* 5.2. *Зображення роботи*

Одна зовнішня сутність може бути використана багаторазово на одній або декількох діаграмах. Зазвичай такий прийом застосовують, щоб не малювати занадто довгих і заплутаних стрілок.


*Рис.* 5.3. Зовнішня сутність

*Стрілки (потоки даних)*описують рух об'єктів з однієї частини системи в іншу. Оскільки в DFD кожна сторона роботи не має чіткого призначення, як в IDEF0, стрілки можуть підходити і виходити з будь-якої грані прямокутника роботи. У DFD також використовуються двонапрямлені стрілки для опису діалогів типу команди-відповіді між роботами, між роботою і зовнішньої сутністю і між зовнішніми сутностями (рис. 5.4).

*Сховище даних* зображують об'єкти в спокої. У матеріальних системах сховища даних зображуються там, де об'єкти очікують обробки, наприклад в черзі. У системах обробки інформації сховища даних є механізмом, який дозволяє зберегти дані для подальших процесів (рис. 5.5).

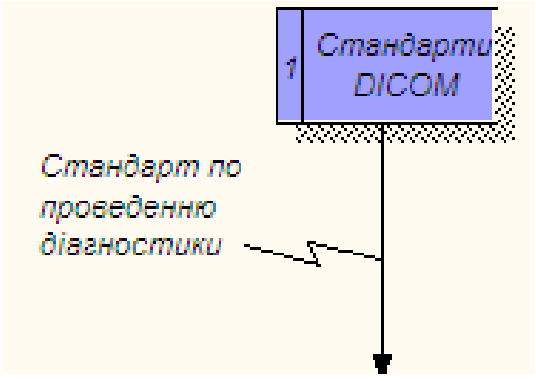



*Рис. 5.4.* Потоки даних (стрілки)


*Рис. 5.5.* Сховище даних

На відміну від стрілок IDEF0, які являють собою жорсткі взаємозв'язки, стрілки DFD показують, як об'єкти (включаючи дані) рухаються від однієї роботи до іншої. Це подання потоків спільно з сховищами даних і зовнішніми

сутностями робить моделі DFD більш схожими на фізичні характеристики системи - рух об'єктів (data flow), зберігання об'єктів (data stores), постачання і розповсюдження об'єктів (external entities) (рис. 5.6).

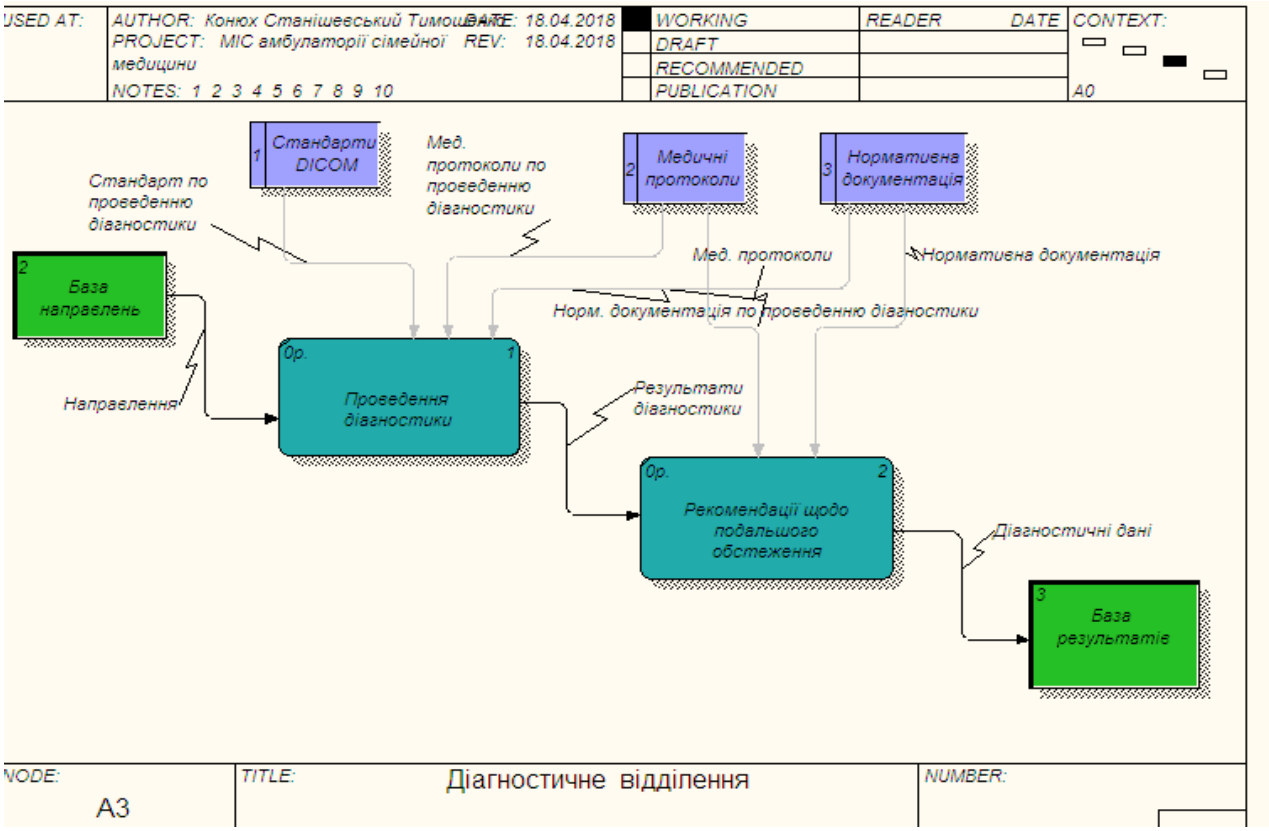



*Рис. 5.6.* Приклад діаграми DFD

Побудову моделі розглянемо на прикладі бізнес-процесу «Діагностичне відділення». *Процес створення діаграми* DFD:

1. Виберіть інструмент ▼ і в процесі декомпозиції роботи «Діагностичне відділення» в діалозі Activity Box Count «клікніть» по радіо-кнопці DFD.

2. У діалозі **Activity Box Count** виберіть кількість робіт - 2 (рис. 5.7).

Клацніть по кнопці ОК і внесіть в нову діаграму імена робіт: «Проведення діагностики», «Рекомендації щодо подальшого обстеження».

**Зображення зовнішніх сутностей.** Виконайте наступні дії:

1. Використовуючи кнопку □, внесіть зовнішні сутності (зовнішні посилання).

2. У діалоговому вікні введіть назву сутностей: «База направлень», «База результатів» (рис. 5.8) і натисніть ОК.Після виконання завдання у нас повинна вийти діаграма,зобрбажена на рис. 5.9.

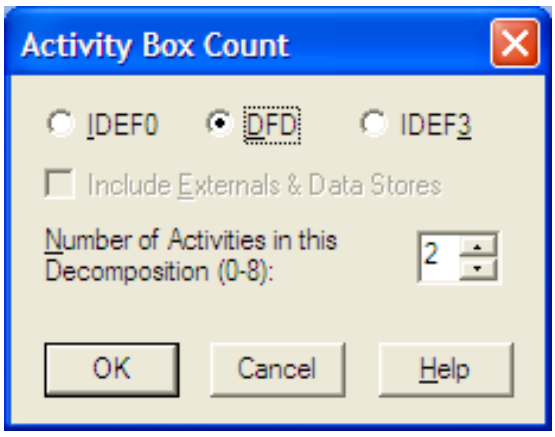



*Рис.* 5.7. Діалог Activity Box Count

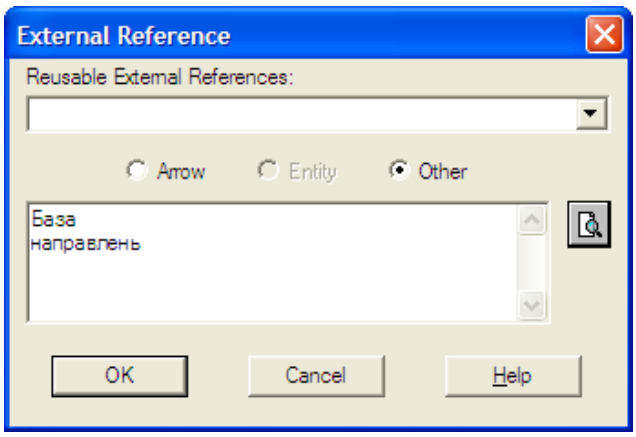



*Рис.* 5.8. Діалогове вікно зовнішньої сутності

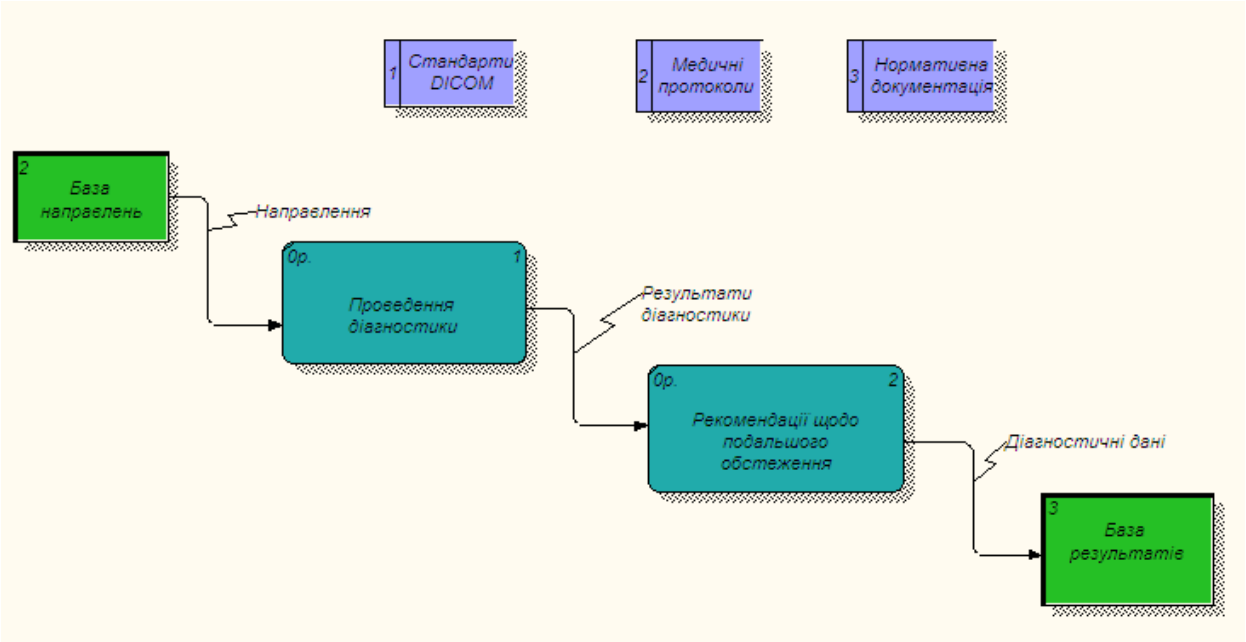



*Рис.* 5.9. Зображення зовнішніх сутностей

**Побудова сховищ**

Використовуючи кнопку  на палітрі інструментів, внесіть сховища даних: «Стандарти DICOM», «Медичні протоколи», «Нормативна документація» (рис. 5.10).

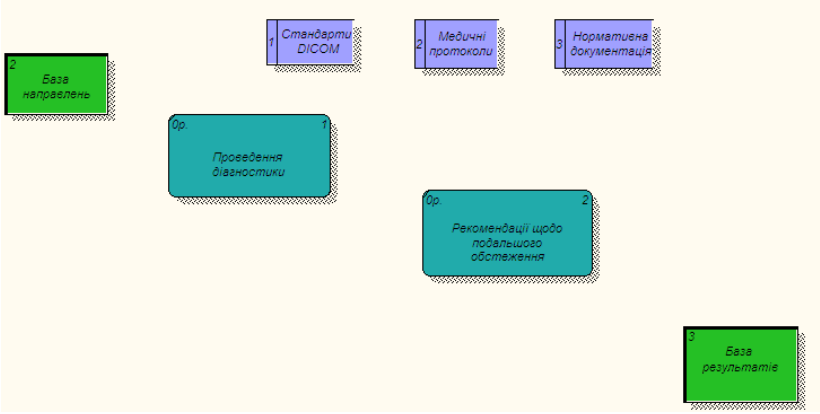



*Рис.* 5.10. Діаграма DFD

Створення внутрішніх посилань

1. Використовуючи інструмент  , створіть внутрішні посилання: «Направлення», «Результати діагностики», «Діагностичні дані».
2. Якщо стрілку необхідно зробити двобічною, для цього клацніть правою кнопкою по стрілці, виберіть у контекстному меню пункт **Style**.
3. Виберіть у діалоговому вікні **Arrow Properties** опцію **Bidirectional** (рис. 5.11.).

Повинна вийти діаграма, зображена на рис. 5.6.

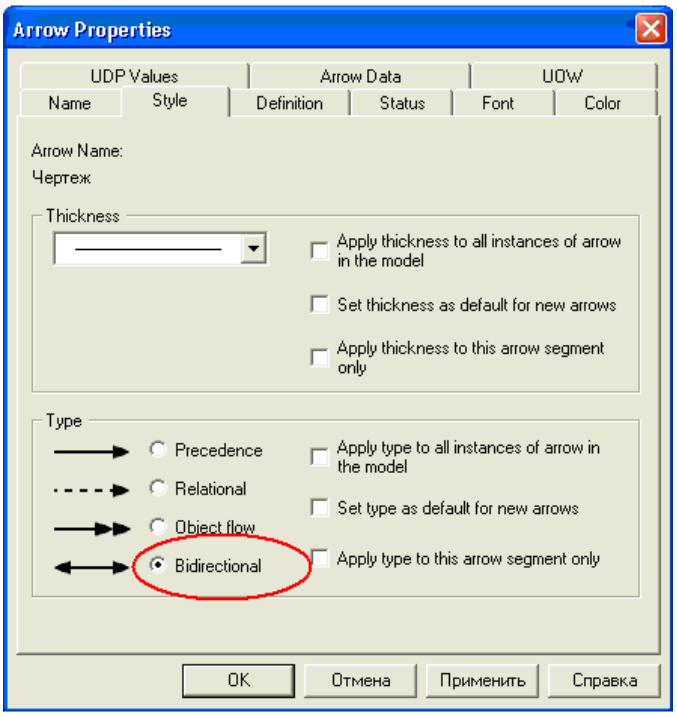



*Рис.* 5.11. Діалогове вікно вибору стилю стрілки

## Теоретичні відомості

**1. Функціональна методика потоків даних.** Метою методики є побудова моделі системи у вигляді діаграми потоків даних (DataFlowDiagram - DFD), що забезпечує правильний опис виходів (відгуку системи у вигляді даних) при заданому впливі на вхід системи (подачі сигналів через зовнішні інтерфейси). Діаграми потоків даних є основним засобом моделювання функціональних вимог до проєктованої ІС.

При створенні діаграми потоків даних використовуються чотири основні поняття: 1) потоки даних, 2) процеси (роботи) перетворення вхідних потоків даних у вихідні, 3) зовнішні сутності, 4) накопичувачі даних (сховища).

1. **Потоки даних** є абстракціями, які використовують для моделювання передачі інформації (або фізичних компонент) з однієї частини системи в іншу. Потоки на діаграмах зображуються іменованими стрілками, орієнтація яких вказує напрямок руху інформації.

2. Призначення **процесу** (роботи) полягає в продукуванні вихідних потоків із вхідних відповідно до дії, яка задається ім'ям процесу. Ім'я процесу повинно містити дієслово в невизначеній формі з подальшим доповненням (наприклад, «отримати документи з відвантаження продукції»). Кожний процес має унікальний номер для посилань на нього всередині діаграми, який може використовуватися спільно з номером діаграми для отримання унікального індексу процесу в усій моделі.

3. **Сховище (накопичувач) даних** дозволяє на зазначених ділянках визначати дані, які будуть зберігатися в пам'яті між процесами. Воно подає «зрізи» потоків даних у часі і містить інформацію, яку можна використовувати в будь-який час після її отримання, при цьому дані можуть вибиратися в будь-якому порядку. Ім'я сховища має визначати його вміст і бути іменником.

4. **Зовнішня сутність** є матеріальним об'єктом поза контексту системи, що є джерелом або приймачем системних даних. Її ім'я повинно містити іменник, наприклад, «склад товарів». Передбачається, що об'єкти, які подано як зовнішні сутності, не повинні брати участь ні в якій обробці.

Крім основних елементів, до складу DFD входять словники даних і мініспецифікації.

1. *Словники даних* є каталогами всіх елементів даних, присутніх в DFD, включаючи групові та індивідуальні потоки даних, сховища і процеси, а також всі їх атрибути.

2. *Мініспецифікації обробки* описують DFD-процеси нижнього рівня. *Мініспецифікації* мають вигляд алгоритмів опису завдань, які виконуються процесами: множина всіх мініспеціфікацій є повною специфікацією системи.

Процес побудови DFD розпочинається зі створення *основної діаграми типу «зірка»*, на якій зображено процес, що моделюється, і всі зовнішні

сутності, з якими він взаємодіє. У разі складного основного процесу він відразу подається у вигляді декомпозиції на ряд взаємодіючих процесів. Критеріями складності в даному випадку є: 1) наявність великої кількості зовнішніх сутностей, 2) багатофункціональність системи, 3) її розподілений характер.

*Зовнішні сутності* виділяються по відношенню до основного процесу. Для їх визначення необхідно виділити постачальників і споживачів основного процесу (тобто всі об'єкти, які взаємодіють з основним процесом). На цьому етапі опис взаємодії полягає у виборі дієслова, що надає уявлення про те, як зовнішня сутність використовує основний процес або використовується ним. Наприклад, основний процес - «облік звернень громадян», зовнішня сутність - «громадяни», опис взаємодії - «подає заяви та отримує відповіді». Цей етап є важливим, оскільки саме він визначає межі системи, яку моделюють.

Для всіх зовнішніх сутностей будується **таблиця подій**, яка описує їх взаємодію із основним потоком. Таблиця подій включає в себе найменування зовнішньої сутності, подію, її тип і реакцію системи.

На наступному кроці відбувається декомпозиція основного процесу на набір взаємозв'язаних процесів, які обмінюються потоками даних. Самі потоки не конкретизуються, визначається лише характер взаємодії. Декомпозиція завершується, коли процес стає простим, тобто: процес

1) має два-три вхідних і вихідних потоки;

2) можна описати у вигляді перетворення вхідних даних у вихідні;

3) можна описати у вигляді послідовного алгоритму.

Для простих процесів будується мініспеціфікація – формальний опис алгоритму перетворення вхідних даних у вихідні. Мініспецифікація задовольняє таким вимогам: для кожного процесу будується одна специфікація, яка

1) однозначно визначає вхідні та вихідні потоки для заданого процесу;

2) не визначає спосіб перетворення вхідних потоків у вихідні;

3) посилається на наявні елементи, не вводячи нові;

4) використовує, по можливості, стандартні підходи та операції.

Після декомпозиції основного процесу для кожного підпроцесу будується аналогічна **таблиця внутрішніх подій**.

Наступним кроком після визначення повної таблиці подій виділяються потоки даних, якими обмінюються процеси і зовнішні сутності. Найпростіший спосіб їх виділення полягає в аналізі таблиць подій. Події перетворюються в потоки даних від ініціатора події до запитуваного процесу, а реакції - в зворотний потік подій.

Після побудови вхідних і вихідних потоків аналогічним чином будуються *внутрішні потоки*. Для їх виділення для кожного із внутрішніх процесів виділяються постачальники і споживачі інформації. Якщо постачальник або споживач інформації подає процес збереження або запиту інформації, то вводиться сховище даних, для якого цей процес є інтерфейсом.

Після побудови потоків даних діаграма повинна бути перевірена на повноту і несуперечність. *Повнота діаграми* забезпечується, якщо в системі немає «повислих» процесів, які не використовуються в процесі перетворення вхідних потоків у вихідні. *Несуперечливість системи* забезпечується виконанням наборів формальних правил про можливі типи процесів: на діаграмі не може бути потоку, який зв'язує дві зовнішні сутності - ця взаємодія видаляється з розгляду; жодна сутність не може безпосередньо отримувати або віддавати інформацію у сховище даних (сховище даних є пасивним елементом, керованим за допомогою інтерфейсного процесу); два сховища даних не можуть безпосередньо обмінюватися інформацією: ці сховища повинні бути об'єднані.

До *переваг методики DFD* відносяться:

1) можливість однозначно визначити зовнішні сутності, аналізуючи потоки інформації всередині і поза системою;

2) можливість проєктування зверху вниз, що полегшує побудову моделі «як повинно бути»;

3) наявність специфікацій процесів нижнього рівня, що дозволяє подолати логічну незавершеність функціональної моделі і побудувати повну функціональну специфікацію розроблюваної системи.

До *недоліків методики DFD* відносять: необхідність штучного введення керуючих процесів, оскільки дії, що управляють (потоки) і керуючі процеси з точки зору DFD нічим не відрізняються від звичайних; відсутність поняття часу, тобто відсутність аналізу часових проміжків при перетворенні даних (всі обмеження по часу повинні бути введені в специфікаціях процесів).

## 6. Створення діаграми IDEF3

*Мета та завдання практикуму:* освоїти принципи побудови діаграми IDEF3; навчитись встановлювати зв'язки між роботами; освоїти правила створення перехрещень.

План

1. Діаграма IDEF3.

**Завдання:**

1. Створити діаграму IDEF3.
2. Додати посилання в діаграму IDEF3.
3. Зв'язати роботи за допомогою стрілок.
4. Встановити перехрещення.
5. Дати відповіді на контрольні питання.
6. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання кожного пункту.
2. Продемонструвати виконане завдання викладачу.

**Контрольні запитання:**

1. Для чого потрібна діаграма IDEF3?
2. Чим відрізняється діаграма IDEF3 від IDEF0?
3. Для чого встановлюють перехрещення та які існують їхні типи?

### Аудиторна робота

**Приклад 6.1.** Діаграма IDEF3

Наявність в діаграмах DFD елементів для опису джерел, приймачів і сховищ даних дозволяє найбільш ефективно і наочно описати процес документообігу. Однак для **опису логіки взаємодії інформаційних потоків** найбільше підходить **IDEF3**- методологія моделювання, яка використовує графічний опис інформаційних потоків, взаємин між процесами обробки інформації та об'єктів, що є частиною цих процесів.

Діаграми IDEF3 можна використовувати при 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, основною метою якого є надати можливість аналітикам описати:

1) ситуацію, коли процеси виконуються в певній послідовності;
2) об'єкти, що беруть участь спільно в одному процесі.

Кожна робота в IDEF3 описує певний 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 іменним словосполученням, що містить таке іменник.

Точка зору на модель повинна бути задокументована. Зазвичай це точка зору людини, відповідальної за роботу в цілому. Також необхідно задокументувати мету моделі - ті питання, на які покликана відповісти модель.

Діаграма є основною одиницею опису в IDEF3.

**Одиниці роботи - Unit of Work (UOW),** також названі роботами (або діями - activity), є центральними компонентами моделі. В IDEF3 роботи зображуються прямокутниками з прямими кутами (рис. 6.1) і мають **ім'я,** виражене віддієслівним іменником, що позначає процес діїі **номер** (ідентифікатор).

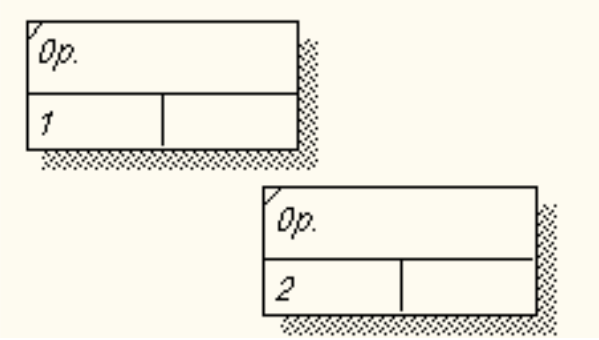



*Рис. 6.1.* Позначення роботи в діаграмі IDEF3

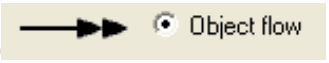
**Зв'язки** показують взаємини робіт. Всі зв'язки в IDEF3 односпрямовані зазвичай зліва направо. В IDEF3 розрізняють три типи стрілок, що зображують зв'язки, стиль яких встановлюється у вкладці **Style** (рис. 6.2) діалогу **Arrow Properties** (пункт контекстного меню **Style**).

**Старша (Precedence) стрілка** 

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

**Стрілка відношення (Relational)** 

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

**Потоки 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.

**Старший зв'язок** показує, що робота-джерело закінчується раніше, ніж починається робота-мета. Зазвичай результатом роботи-джерела стає об'єкт, необхідний для запуску роботи-мети. У цьому випадку стрілку, що позначає об'єкт, зображують з подвійним накінечником.

Ім'я стрілки повинно ясно ідентифікувати відображуваний об'єкт. Потік об'єктів має ту ж семантику, що і старша стрілка.

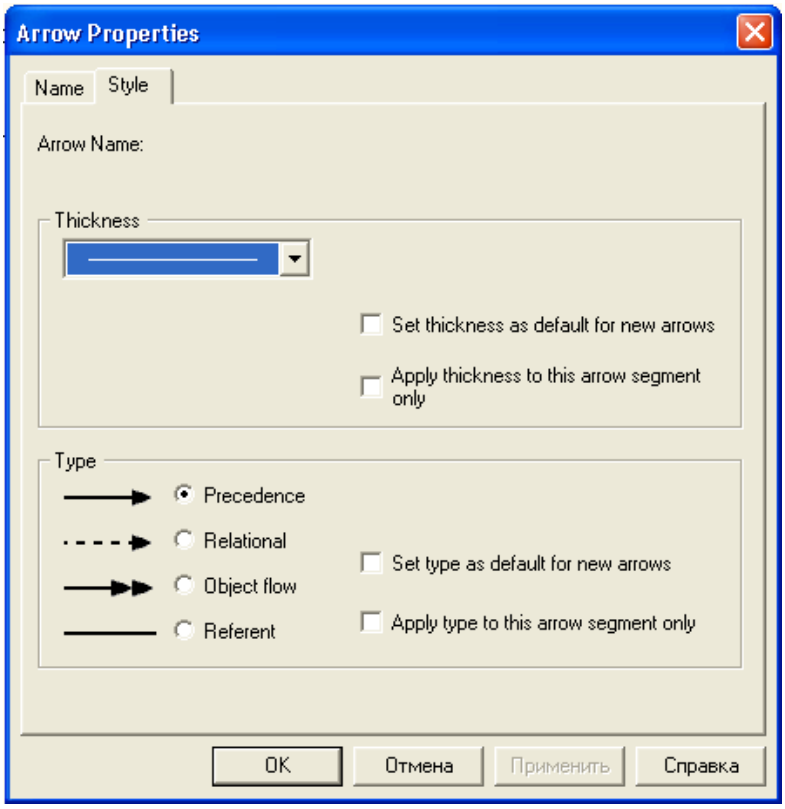



*Рис. 6.2.* Вкладка Style діалогу Arrow Properties

**Зв'язок** показує, що стрілка є альтернативою старшої стрілкою або потоку об'єктів у сенсі завдання послідовності виконання робіт - робота-джерело не обов'язково повинна закінчитися раніше, ніж робота-мета почнеться. Більше того, робота-мета може закінчитися раніше, ніж закінчиться робота-джерело (рис. 6.3.).

**Перехрещення (Junction).** Закінчення однієї роботи може служити сигналом до початку декількох робіт, або ж одна робота для свого запуску може очікувати закінчення декількох робіт. Перехрещення використовуються для відображення логіки взаємодії стрілок при злитті і розгалуженні або для відображення безлічі подій, які можуть або повинні бути завершені перед початком наступної роботи.

Розрізняють перехрещення для злиття (**Fan-in Junction**) і розгалуження (**Fan-in Junction**) стрілок. Перехрещення не може використовуватися одночасно для злиття і розгалуження. Для внесення перехрещень служить кнопка в палітрі інструментів. У діалозі **Junction Type Editor** потрібно вказати тип перехрещення (рис. 6.4). Сутність кожного типу наведено у табл.6.1.Всі перехрещення на діаграмі нумеруються, кожен номер має префікс **J** (рис. 6.5).

Можна редагувати властивості перехрещення (рис 6.6) За допомогою діалогу **Junction Properties**, який викликається з контекстного меню.

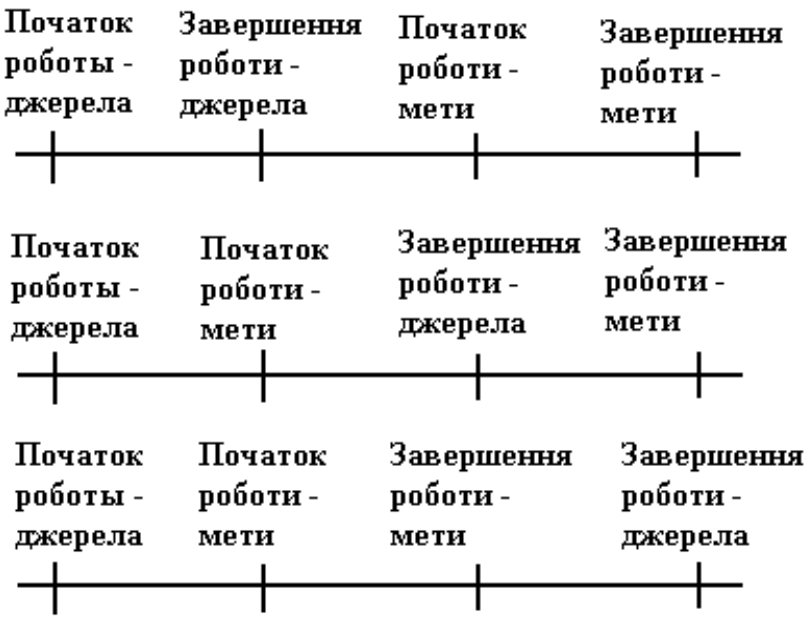



*Рис. 6.3.* Часова діаграма виконання робіт

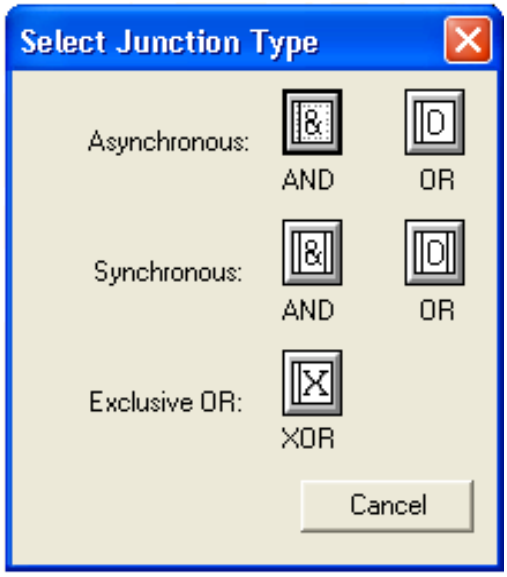



*Рис. 6.4.*Типи перехрещень

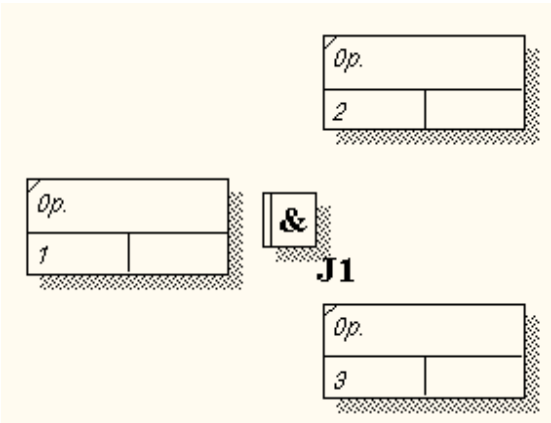



*Рис. 6.5.* Позначення нумерації перехрещення

Таблиця 6.1

Типи перехрещень

| Позначення | Найменування | Суть у разі злиття стрілок Fan-in Junction | Суть у разі розгалуження стрілок Fan-in Junction |
|---|---|---|---|
| Asynchronous: AND | Асинхроне «І» (Asynchronous AND) | Усі попередні процеси повинні бути завершені | Всі наступні процеси повинні бути запущені |
| Synchronous: AND | Синхроне «І» (Synchronous AND) | Усі попередні процеси завершені одночасно | Всі наступні процеси запускаються одночасно |
| Asynchronous: OR | Асинхроне «АБО» (Asynchronous OR) | Один або декілька попередніх процесів повинні бути завершені | Один або декілька таких процесів повинні бути запущені |
| Synchronous: OR | Синхроне «АБО» (Synchronous OR) | Один або декілька попередніх процесів завершені одночасно | Один або декілька таких процесів запускаються одночасно |
| Exclusive OR: XOR | Виключаюче «АБО»XOR (Exclusive OR) | Тільки один попередній процес завершено | Тільки один наступний процес запускається |

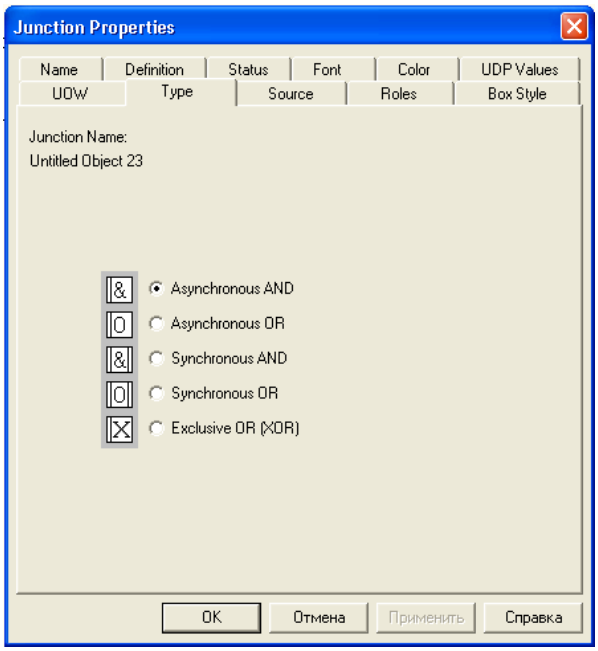



*Рис. 6.6.* Діалогове вікно властивостей перехрещень

На відміну від IDEF0 і DFD в IDEF3 стрілки можуть зливатися і розгалужуватися тільки через перехрещення.

**Правила створення перехрещень.** На одній діаграмі IDEF3 може бути створено кілька перехрещень різних типів. Певні поєднання перехрещень для

злиття і розгалуження можуть призводити до логічних невідповідностей. Щоб уникнути конфліктів, необхідно дотримуватися таких правил:

1. Кожному перехрещенню для злиття повинен передувати перехрестя для розгалуження.

2. Перехрещення для злиття «І» не може слідувати за перехрещенням для розгалуження типу синхронного або асинхронного «АБО». Дійсно, після роботи 1 може запускатися тільки одна робота - 2 або 3, а для запуску роботи 4 вимагається закінчення обох робіт - 2 і 3. Такий сценарій не може реалізуватися (рис. 6.7).

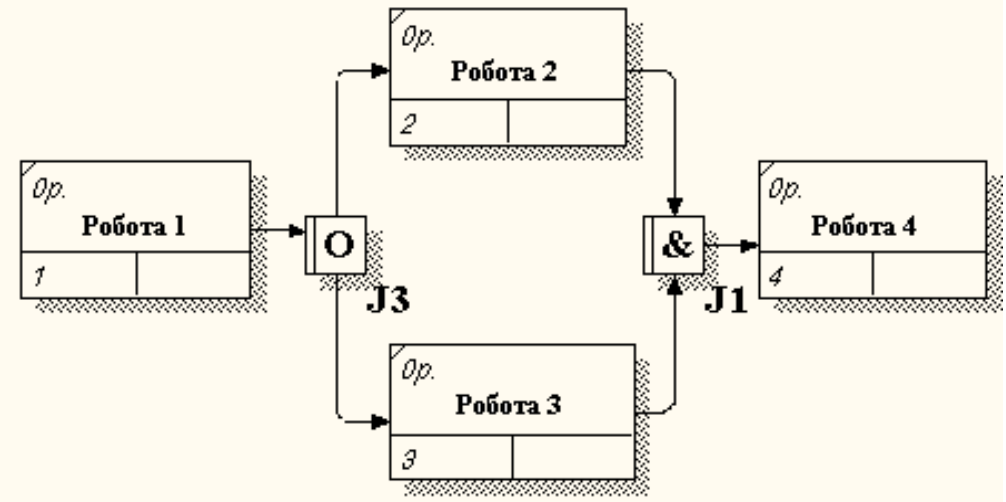



*Рис. 6.7.* Неправильне розміщення перехрещень. Перехрещення для злиття «І» не може слідувати за перехрещенням для розгалуження «АБО»

3. Перехрещення для злиття «І» не може слідувати за перехрещенням для розгалуження типу виключає «АБО» (рис. 6.8).

4. Перехрещення для злиття типу виключає «АБО» не може слідувати за перехрещенням для розгалуження типу «І» (рис. 6.9). Тут після завершення роботи 1 запускаються обидві роботи - 2 і 3, а для запуску роботи 4 потрібно, щоб завершилася одна і тільки одна робота: або 2, або 3.

5. Перехрещення, що має одну стрілку на одній стороні, повинне мати більше однієї стрілки на інший.

Побудову моделі розглянемо на прикладі бізнес-процесу «Огляд сімейного лікаря».

**Створення діаграми IDEF3**

1. Необхідно перейти на діаграму А3 і, вибравши інструмент ▼, декомпонувати роботу «Збірка вироби».

2. У діалозі Activity Box Count встановіть кількість робіт 4 і нотацію IDEF3 (рис. 6.10).

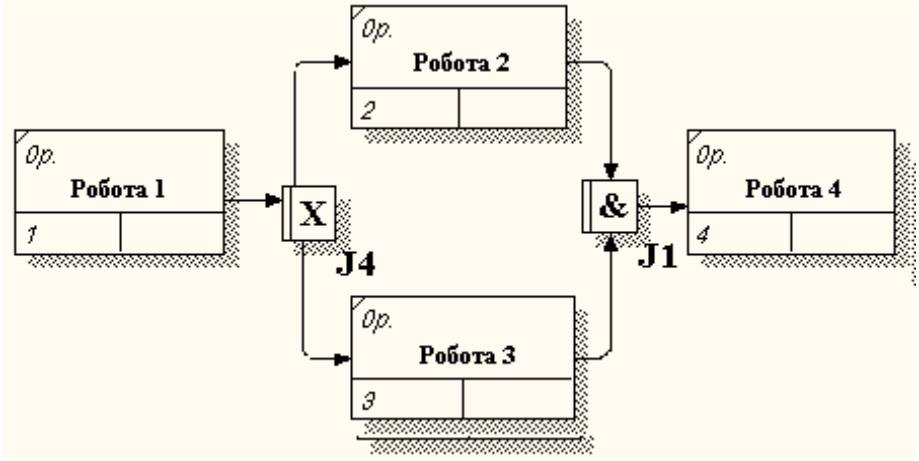



*Рис. 6.8.* Неправильне розміщення перехрещень. Перехрещення для злиття «І» не може слідувати за перехрещенням для розгалуження типу виключає «АБО»

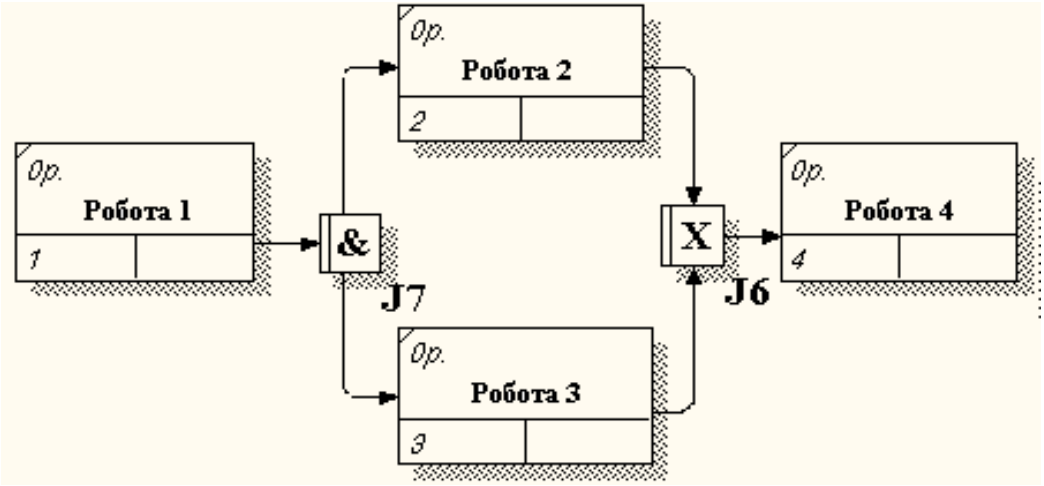



*Рис. 6.9.* Неправильне розміщення перехрещень. Перехрещення для злиття типу виключає «АБО» не може слідувати за перехрещенням для розгалуження типу «І»

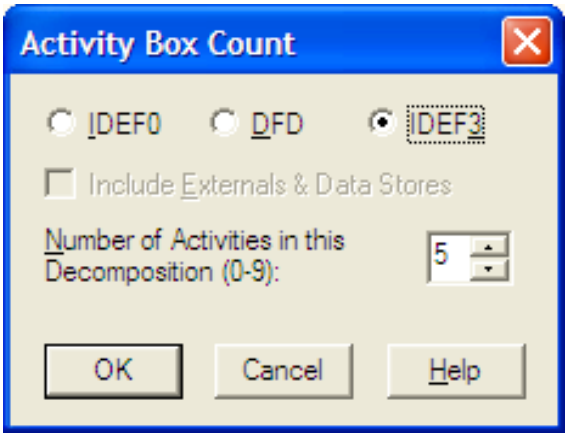



*Рис. 6.10.* Вибір нотації IDEF3 в діалозі Activity Box Count

З'являється діаграма IDEF3, що містить роботи (**UOW**).

3. ПКМ по роботі, виберіть у контекстному меню Name і внесіть ім'я роботи «Огляд сімейного лікаря».

4. У вкладці **Definition** внесіть визначення «Проводиться огляд сімейним лікарем» (рис. 6.11.).
5. Під вкладку UOW, внесіть властивості роботи (табл.6.2.).
6. Внесемо в діаграму ще 2 роботи (кнопка  ).
7. Внесемо імена наступних робіт: «Надходження пацієнта на прийом», «Діалог з пацієнтом», «Огляд лікарем», «Запис у медичну карту», «Постановка діагнозу», «Направлення на дообстеження», «Призначення лікування» (рис. 6.12).

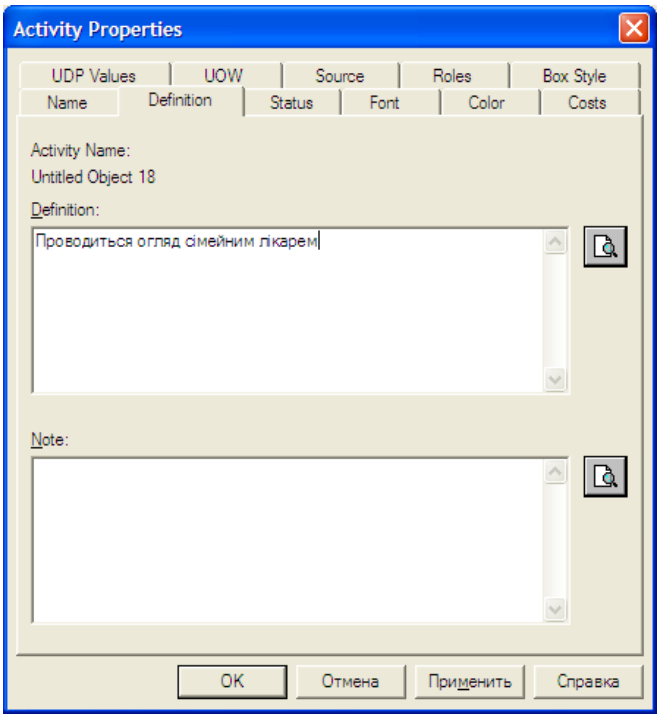



*Рис. 6.11.* Діалогове вікно властивостей роботи

Таблиця 6.2

Властивості UOW

| *Тип* | *Використання* |
|---|---|
| Name | Огляд сімейного лікаря |
| Definition | Проводиться огляд сімейним лікарем |
| Objects | Надходження пацієнта на прийом, Діалог з пацієнтом, Огляд лікарем, Запис у медичну карту, Постановка діагнозу, Направлення на дообстеження, Призначення лікування |
| Constrains | Проведення огляду можливе лише за наявності пацієнта та лікаря |

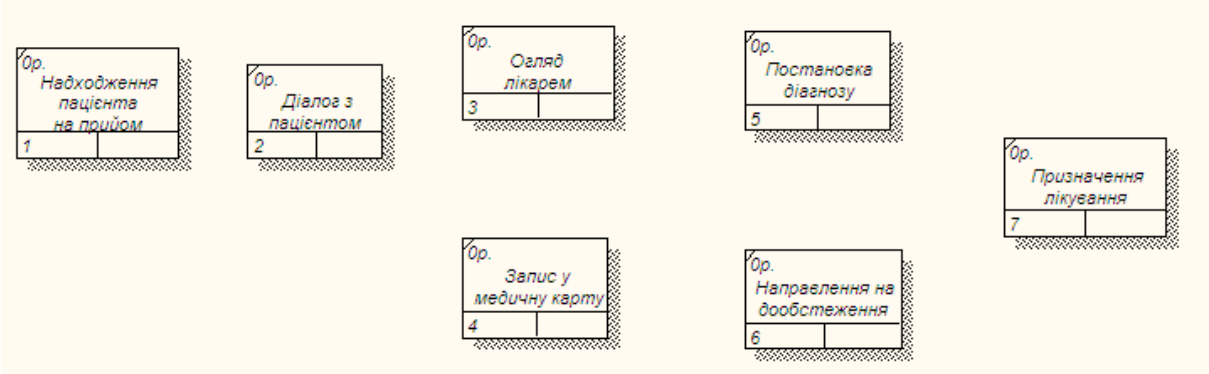
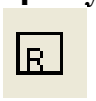



*Рис. 6.12.* Фрагмент діаграми IDEF3

**Додавання в діаграму IDEF3 об'єкта посилання**

1. За допомогою кнопки  (додати в діаграму об'єкт посилання - Referent), розташованої в палітрі інструментів, створіть об'єкт посилання.
2. Внесемо ім'я об'єкта зовнішнього посилання «Зареєстрований пацієнт».
3. Замінимо стиль стрілки на Referent (рис. 6.13.).

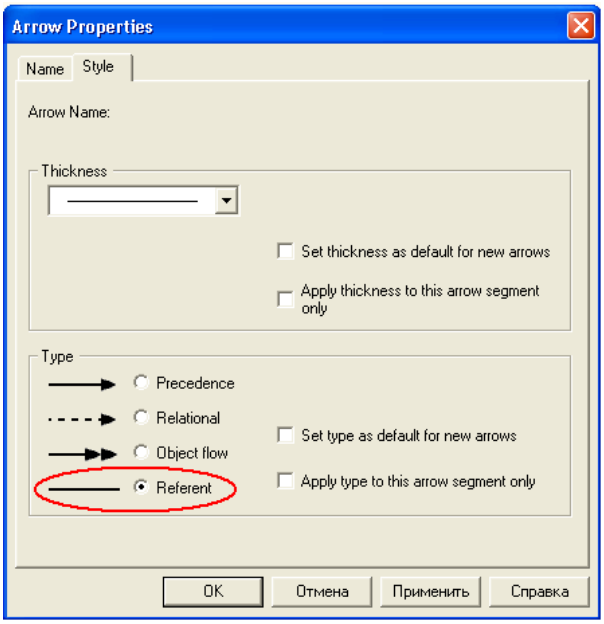
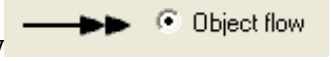



*Рис. 6.13.* Діалогове вікно визначення стилю стрілок

Об'єкт посилання в IDEF3 виражає певну ідею, концепцію або дані, які не можна пов'язати зі стрілкою, перехрестям або роботою.

**Зв'язування робіт за допомогою стрілок.**

1. Зв'яжіть стрілкою роботи «Надходження пацієнта на прийом» (вихід) і «Діалог з пацієнтом».
2. Змініть стиль стрілки на **Object Flow**  (рис. 6.14.).

В IDEF3 ім'я стрілки може бути відсутнім, хоча програма показує відсутність імені як помилку.

3. Враховуючи той факт, що роботи «Огляд лікарем», «Запис у медичну карту» повинні слідувати тільки один за одним, з'єднайте їх стрілкою **Precedence** 

. Вона показує, що робота-джерело повинна закінчитися раніше, ніж робота-мета почнеться (рис. 6.15).

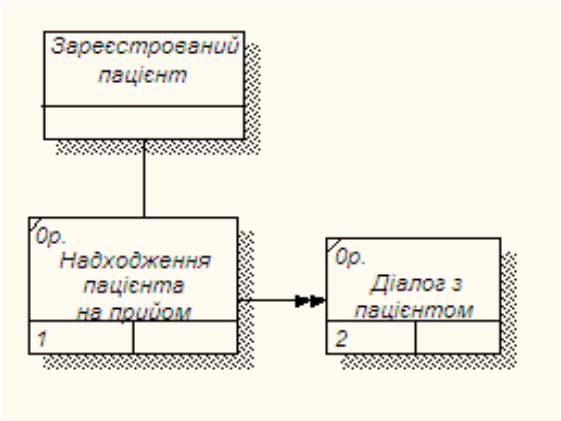



*Рис. 6.14.* Результат створення UOW і об'єкта посилання

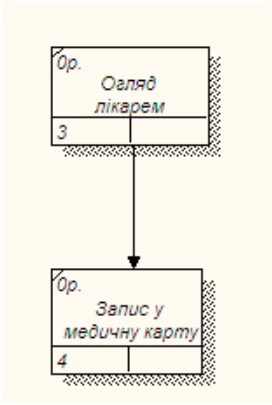



*Рис. 6.15.* Результат зв'язування послідовно виконуваних робіт

**Встановлення перехрещень**

1. За допомогою кнопки  на палітрі інструментів внесіть одне перехрещення типу синхронного «I» (один або декілька наступних процесів повинні бути запущені) і зв'яжіть роботи з перехрещенням, як показано на рис. 6.16 (це випадок розгалуження стрілок **Fan-in Junction**).

2. Встановимо перехрещення для злиття стрілок.Маємо отримати (рис. 6.17).

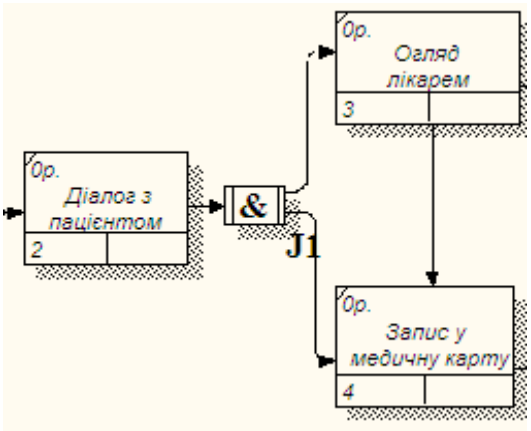



*Рис. 6.16.* Результат створення перехрестя

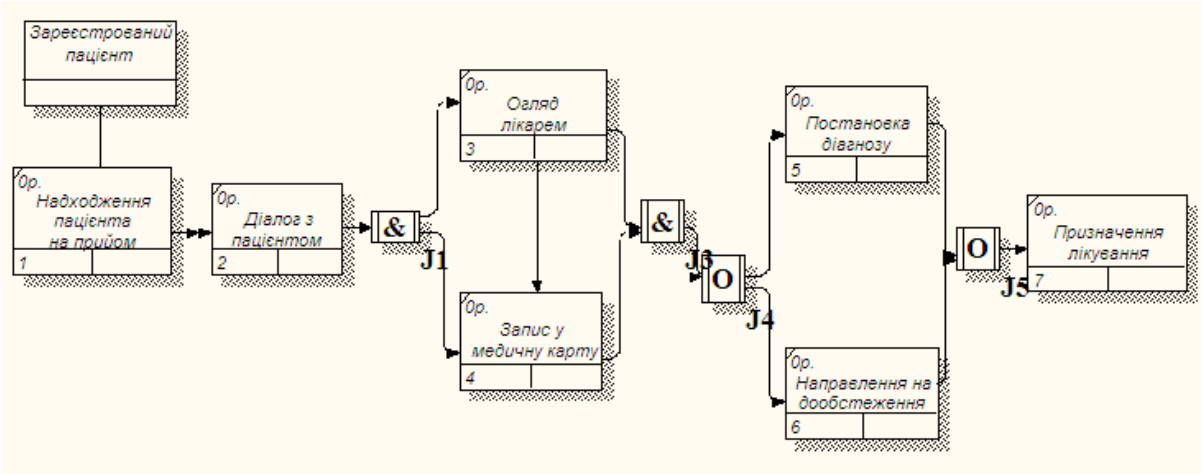
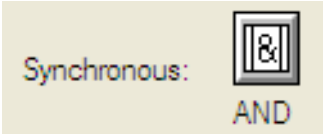



*Рис. 6.17.* Діаграма IDEF3

*Примітка.* Так як роботи «Постановка діагнозу» та «Направлення на дообстеження» повинні проводитися після закінчення всіх попередніх робіт, то обраний тип перехрестя - синхронне «I».




.

## 7. Вартісний аналіз бізнес – процесу

*Мета та завдання практикуму:*1) освоїти послідовність і правила визначення параметрів вартісного аналізу; 2) навчитись складати звіт по вартості бізнес – процесу.

**Завдання**

1. Налаштувати параметри вартісного аналізу.
2. Внести вартість витрат.
3. Скласти звіт.
4. Дати відповіді на контрольні питання.
5. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання до кожного пункту.

2.Продемонструвати виконане завдання викладачу.

**Контрольні запитання**

1. Дайте визначення вартісного аналізу та його ролі в моделюванні бізнес-процесів?
2. Які визначення включає в себе вартісний аналіз?
3. На чому оснований вартісний аналіз?
4. Як виглядає скорочена назва вартісного аналізу?

**Аудиторна робота**

**Приклад 7.1.** Вартісний аналіз (ABC)

Для того щоб визначити якість створеної моделі з точки зору ефективності бізнес-процесів, необхідна система метрики. Якість слід оцінювати кількісно.

Програма надає аналітику і два інструменти для оцінки моделі - вартісний аналіз, заснований на роботах (**Activity Based Costing, ABC**), і властивості, що визначаються користувачем (User Defined Properties, UDP).

**ABC** (вартісний аналіз) є широко поширеною методикою, яка використовується міжнародними корпораціями та державними організаціями (у тому числі Департаментом оборони США) для ідентифікації справжніх рушіїв витрат в організації. Саме цю методику ми розглянемо в лабораторній роботі.

*Вартісний аналіз* має вигляд угоди про облік, що використовується для збору витрат, пов'язаних з роботами, з метою визначити загальну вартість процесу.

Вартісний аналіз заснований на моделі процесів, бо кількісна оцінка неможлива без детального розуміння функціональності підприємства.

Зазвичай **ABC** застосовується для того, щоб зрозуміти походження вихідних витрат і полегшити вибір потрібної моделі робіт при реорганізації діяльності підприємства. За допомогою вартісного аналізу можна вирішити такі завдання, як:

1) визначення дійсної вартості виробництва продукту,

2) визначення дійсної вартості підтримки клієнта,

3) ідентифікація робіт, які коштують більше всього (ті, які повинні бути поліпшені в першу чергу),

4) забезпечення менеджерів фінансовою підтримкою для впровадження змін та ін.

**ABC** може проводитися тільки тоді, коли модель роботи послідовна (слідує синтаксичним правилам **IDEF0**), коректна (відображає бізнес-процес), повна (охоплює всю розглянуту область) і стабільна (проходить цикл експериментів без змін), іншими словами, коли створення моделі роботи закінчено.

**ABC** включає такі основні поняття:

***Об'єкт витрат*** - причина, по якій робота виконується.

| |
|---|
| Вартість робіт є сумарною вартістю об'єктів витрат (MIC, рис. 7.1) |

***рушій витрат*** - характеристики входів та керування роботою («Пацієнт, що надходить до амбулаторії сімейної медицини», «МКХ-10», «Довідник препаратів» - рис. 7.1.), які впливають на те, як виконується і як довго триває робота;

***центри витрат***, які можна трактувати як статті витрат.

При проведенні вартісного аналізу спочатку задаються одиниці виміру часу і грошей. Загальні витрати по роботі розраховуються як сума за всіма центрами витрат. При обчисленні витрат батьківської роботи спочатку обчислюється добуток витрат дочірньої роботи на частоту роботи (число разів, яке робота виконується в рамках проведення батьківської роботи), потім результати складаються. Якщо у всіх роботах моделі включений режим **Compute from Decompositions**, подібні обчислення автоматично проводяться по всій ієрархії робіт знизу вгору (рис. 7.2)

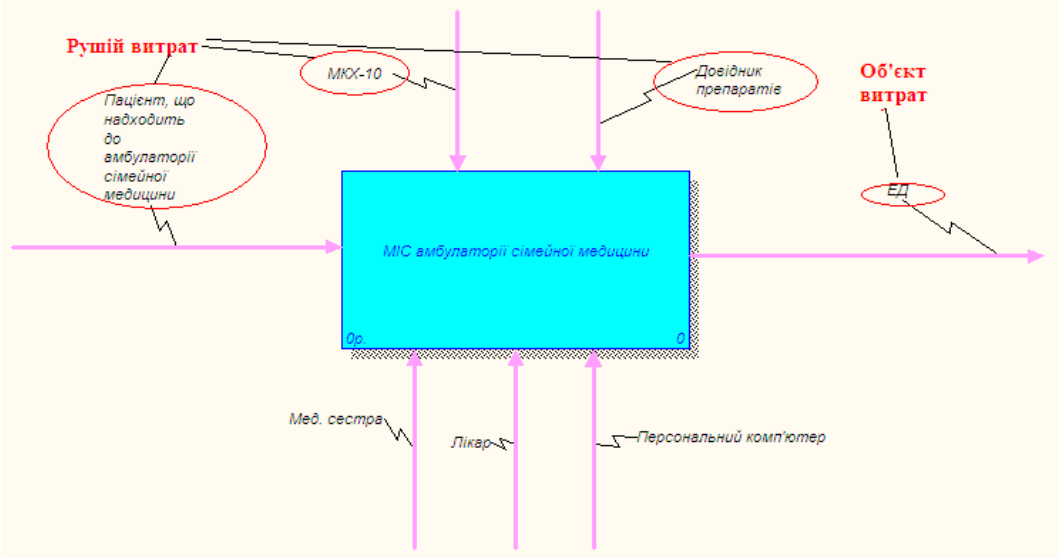



*Рис. 7.1.* Ілюстрація термінів ABC

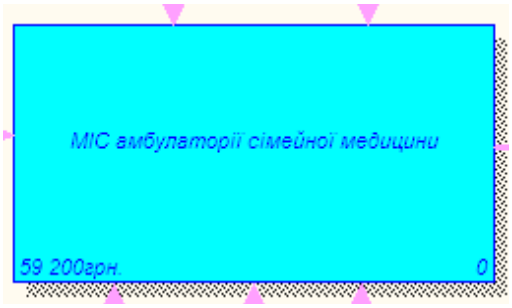



*Рис. 7.2.* Загальні витрати на роботу «МІС амбулаторії сімейної медицини»

Цей досить спрощений принцип підрахунку справедливий, якщо роботи виконуються послідовно. Вбудовані можливості BPwin дозволяють розробляти спрощені моделі вартості, які проте виявляються надзвичайно корисними для попередньої оцінки витрат. Якщо схема виконання більш складна (наприклад, роботи проводяться альтернативно), можна відмовитися від підрахунку і задати підсумкові суми для кожної роботи вручну.

Налаштування параметрів вартісного аналізу

1. У діалоговому **вікні Model Properties** (Model - Model Properties) у вкладці **ABC** встановіть одиниці виміру грошей і часу - гривні і години (рис.7.3).

2. Перейдіть в Dictionary - Cost Center і в діалозі Cost Center Dictionary внесіть назву і визначення центрів витрат (табл. 7.1).

3. Для відображення вартості кожної роботи в нижньому лівому кутку прямокутника перейдіть в меню **Model - Model Properties** і у вкладку **Display** діалогу **Model Properties** і увімкніть опцію **ABC Data** (рис. 7.4).

4. Для зображення вартості, частоти або тривалості роботи перемкніть радіо-кнопки в групі ABC Units (рис. 7.5).

5. Для призначення вартості роботі «Огляд сімейного лікаря» клацніть по ній правою кнопкою миші і виберіть в контекстному меню Cost.

6. Відкриється діалогове вікно для внесення вартості витрат (рис. 7.6).

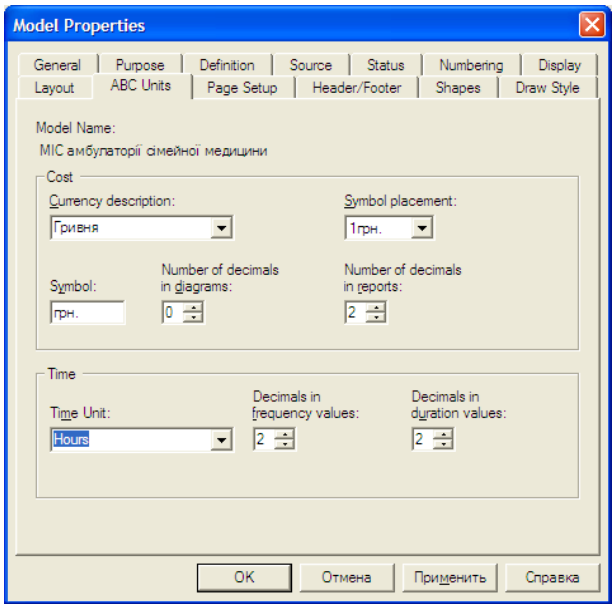



*Рис. 7.3.* Вкладка ABC Unit діалогу Model Properties

Таблиця 7.1

Центри витрат ABC

| *Центр витрат* | *Визначення* |
|---|---|
| Control | Витрати на управління, пов'язані зі складанням графіка робіт, контролем і перевіркою роботи персоналу |
| Workers | Витрати на оплату робітників, зайнятих у MIC |
| Materials | Витрати на закупівлю компонентів |

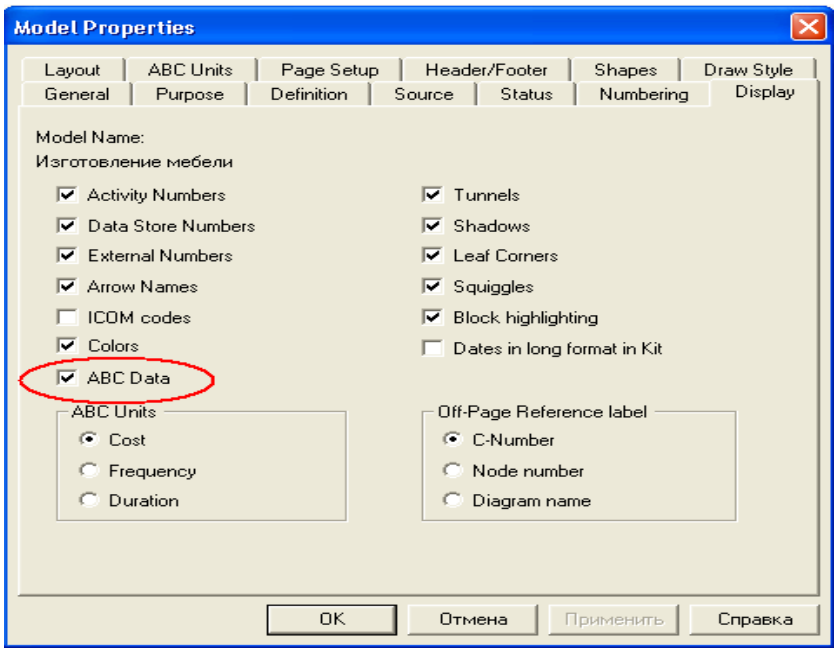



*Рис. 7.4.* Вкладка Display діалогу Model Properties

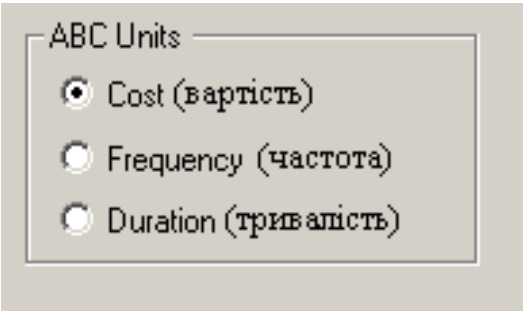



*Рис. 7.5.* Призначення радіо-кнопок в групі ABC Units

7. У діалоговому вікні Cost Center Editor додайте центри витрат «Матеріали», «Робоча сила», «Управління» (рис. 7.7).

8. У вкладці Costs діалогу Activity Properties вкажіть частоту проведення даної роботи в рамках загального процесу (Frequency) і тривалість (Duration).

9. Виберіть у списку один з центрів витрат і у вікні Cost задайте його вартість (рис. 7.8).

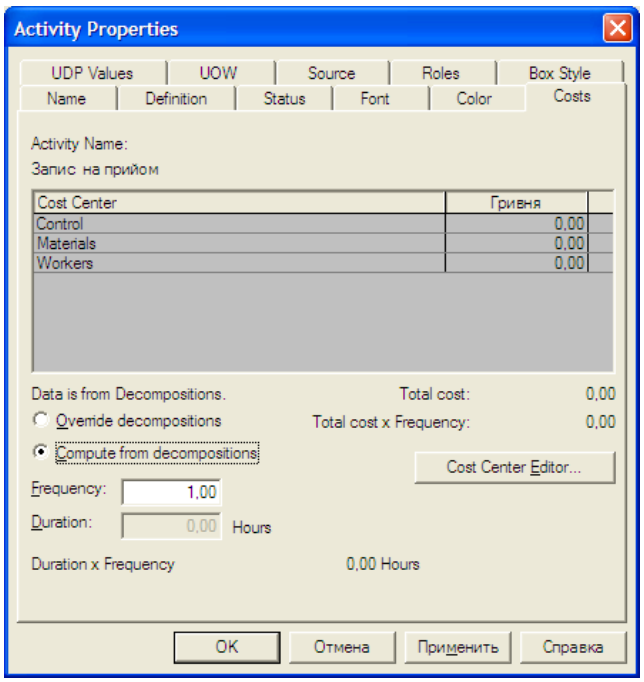



*Рис. 7.6.* Діалог Activity Properties

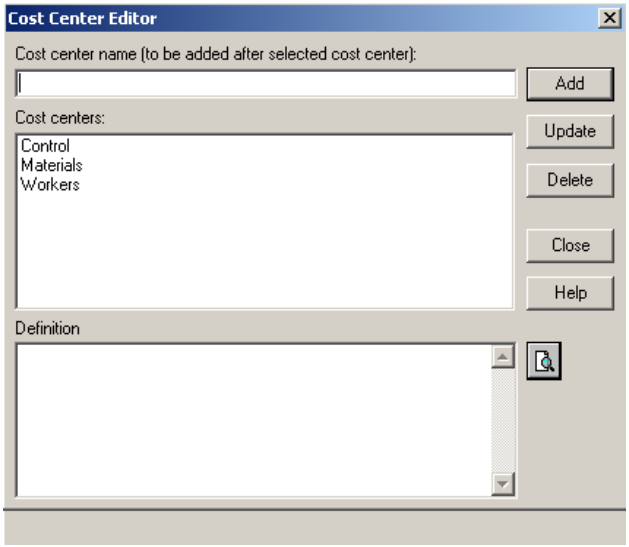



*Рис. 7.7.* Діалог Cost Center Editor

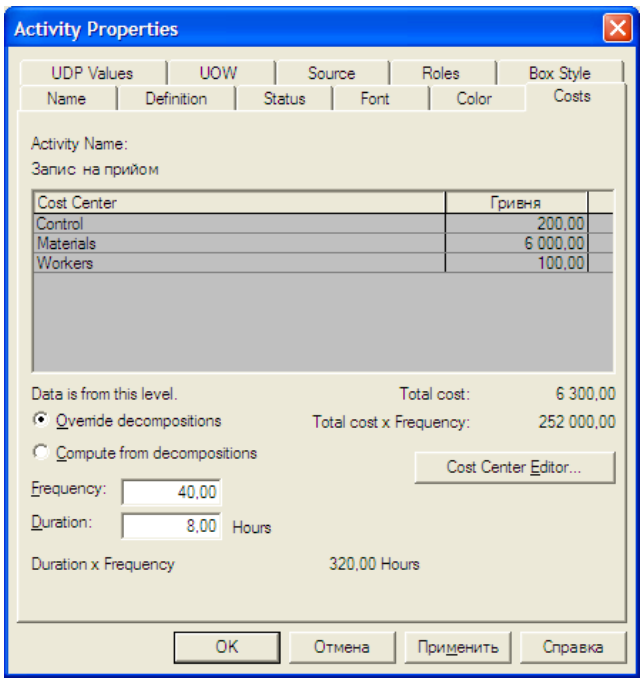



*Рис. 7.8.* Внесення вартості в центри витрат

Внесення вартості витрат:

1. Для етапів «Аналіз вхідних даних», «Проєктування», «Тестування», «Впровадження» та «Супровід» на діаграмі **A1** слід внести параметри **ABC** на власний розсуд.

2. Після внесення вартостей маємо отримати результат, зображений на рис. 7.9.

**Складання звіту.**Щоб згенерувати звіт, слід виконати такі дії:

1. Виберіть пункт меню **Activity Cost Report (Tools - Reports - Activity Cost Report)**.

2. Вкажіть пункти в діалозі **Activity Based Costing Report** (рис. 7.10), За якими хочете отримати інформацію. Призначення пунктів можна подивитися в табл. 7.3. Маємо отримати звіт, зображений на рис. 7.11.

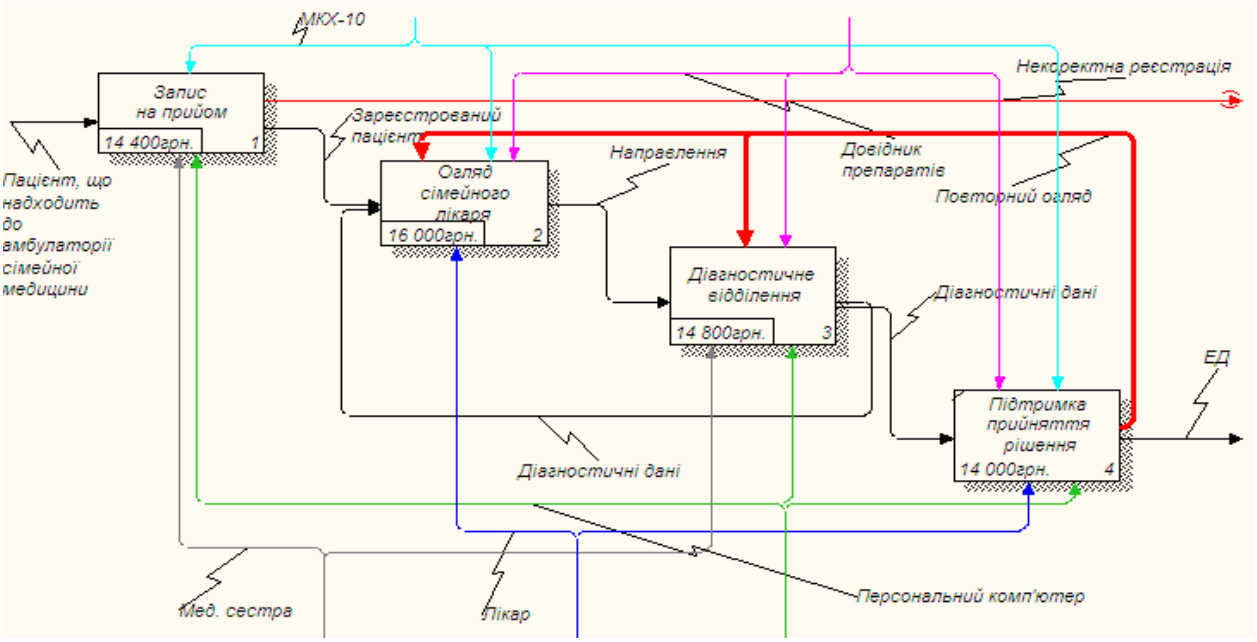



*Рис. 7.9.* Результат вартості робіт на діаграмі A1

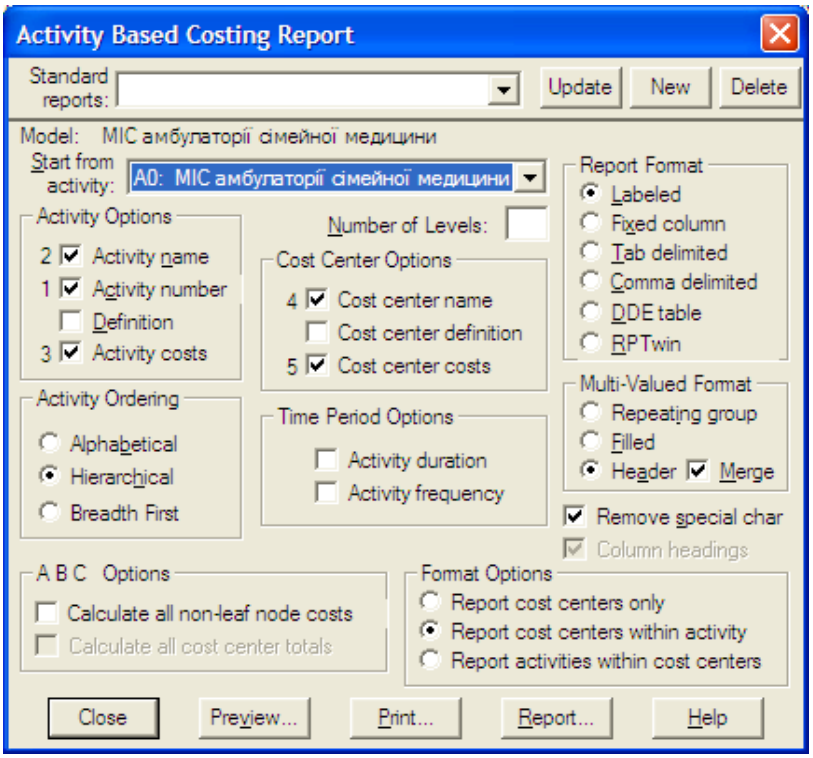
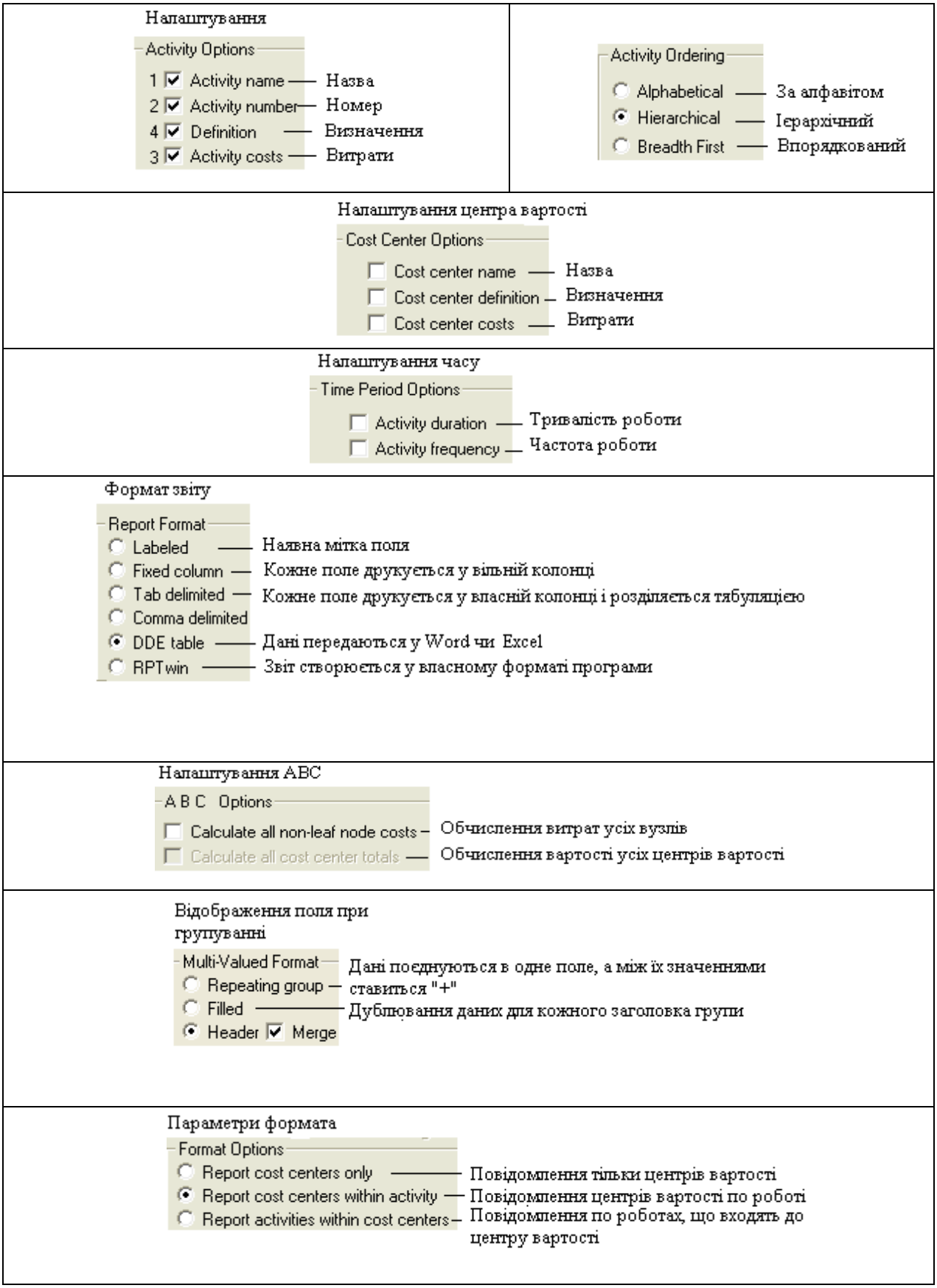



*Рис. 7.10.* Діалог Activity Based Costing Report

Таблиця 7.3

Пункти Activity Based Costing Report та їх значення



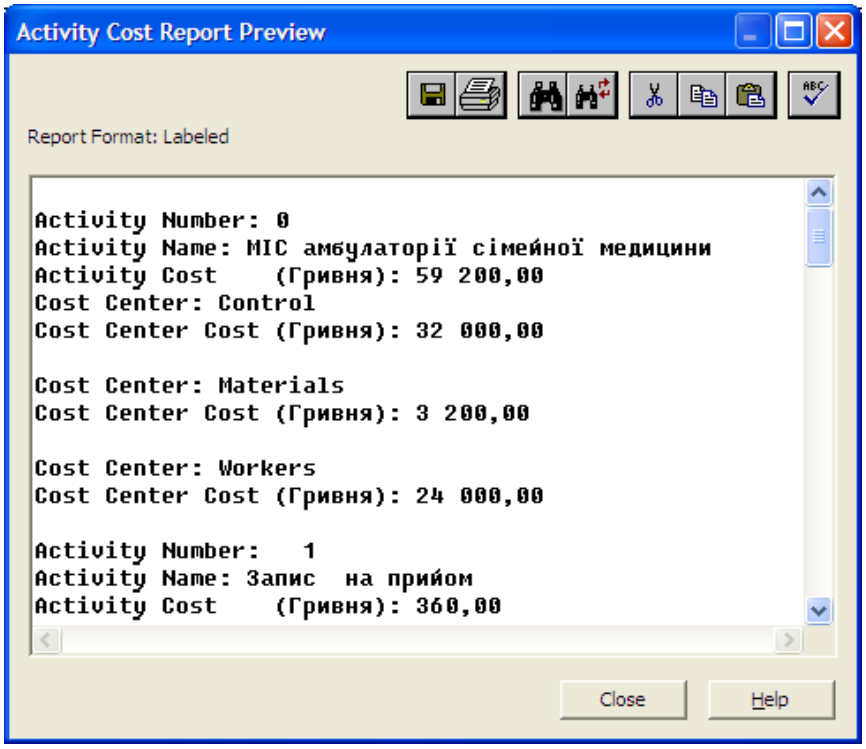



*Рис. 7.11.* Згенерований звіт

## Питання до розділу 1

1. Що визначає контекстна діаграма?
2. Які основні поняття використовуються при створенні функціональної діаграми IDEF0?
3. Вкажіть, з якою метою будуються діаграми для експозиції (FEO).
4. Що означає поява «тунелів» на діаграмі?
5. Вкажіть, чому повинна відповідати точка зору.
6. Вкажіть, що показує діаграма дерева вузлів.
7. Які стрілки називаються граничними?
8. Які стрілки називаються стрілками механізму (Mechanism)?
9. Вкажіть, що входить у визначення контексту моделі.
10. Вкажіть основні поняття ABC-аналізу.
11. Які основні поняття використовуються при створенні функціональної діаграми IDEF0?
12. Вкажіть, яка діаграма розглядає систему як сукупність предметів.
13. Яке призначення має вартісний аналіз?
14. Вкажіть основні компоненти діаграми потоків даних.
15. Вкажіть основні елементи імітаційного моделювання.
16. Який варіант правильно описує цифрами послідовність етапів ABC-аналізу?
17. Вкажіть, для чого в діаграмах IDEF3 використовуються перехрестя.
18. Вкажіть моделі, які враховують час виконання функцій.

# РОЗДІЛ 2. МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ЗАСОБАМИ ERWIN

## 8. Відображення моделі даних в середовищі засобу Erwin. Створення логічної моделі даних

*Мета та завдання практикуму:*1) ознайомитись із засобом моделювання даних і проєктування баз даних AllFusion ERwin Data Modeler (ERwin); 2) ознайомитись із механізмом розробки моделі бази даних на логічному та фізичному рівнях в середовищі ERwin.

План

1. Моделювання інформаційного забезпечення ІС

**Завдання**

1. Обрати приклад ІС та описати відповідну модель бази даних на логічному та фізичному рівнях в середовищі ERwin:

2. Запустити програму та створити нову модель бази даних в середовищі ERwin.

3. Зберегти модель.

4. Відповісти на контрольні питання.

5. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи.**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання до кожного пункту.

2. Продемонструвати виконане завдання викладачу.

### Аудиторна робота

### 1. Відображення моделі даних в середовищі засобу ERwin

ERwin має два рівня зображення моделі – логічний та фізичний.

*Логічний рівень* – це абстрактне відображення даних, коли дані подають таким чином, як вони виглядають у реальному світі, і можуть називатися так, як вони називаються в реальному світі. Об’єкти моделі, що тут задіяні називаються сутностями та атрибутами. Логічна модель даних (МД) розробляється на основі існуючих моделей даних (наприклад, реляційної), але ніяк не пов’язана з конкретною реалізацією системи управління бази даних (СУБД) та інших фізичних умов реалізації. Вона може бути побудована на основі іншої логічної моделі, наприклад, на основі моделі потоків даних або процесів.

Логічна модель даних є джерелом інформації для етапу фізичного проєктування. Вона надає розробнику фізичної моделі даних засоби проведення всебічного аналізу різних аспектів роботи с даними, що має виключне значення для вибору дійсно ефективного проєктного рішення.

*Фізична МД*, навпаки, залежить від конкретної СУБД і фактично є відображенням системного каталогу. Вона містить інформацію про всі об’єкти БД. Оскільки стандартів для об’єктів БД не існує (наприклад, не існує стандарту для типів даних), фізична модель залежить від конкретної реалізації

СУБД. Таким чином, одній і тій же моделі можуть відповідати декілька різних фізичних моделей.

На фізичному рівні об'єкти БД повинні називатися таким чином, як вимагає обмеження СУБД. На *логічному рівні* цим об'єктам можна дати синоніми – назви, які більш зрозумілі неспеціалістам, в тому числі на кирилиці і з використанням спеціальних символів. Така відповідність дозволить краще документувати модель і дає можливість обговорювати структуру даних з експертами прикладної області.

Після опису логічної моделі проєктувальник може обрати необхідну СУБД, і ERwin автоматично створить відповідну фізичну модель. На основі фізичної моделі ERwin може згенерувати системний каталог СУБД або відповідний SQL-скрипт. Цей процес називають **прямим проєктуванням** (Forward Engineering). Тим самим досягається масштабованість – створивши одну логічну модель даних, можна згенерувати фізичні моделі для будь-якої СУБД, що підтримує ERwin. З іншої сторони, ERwin має змогу за змістом системного каталогу або SQL-скрипту відтворити фізичну і логічну моделі даних (**зворотне проєктування,** Reverse Engineering). На основі отриманої логічної моделі даних можна згенерувати фізичну модель для іншої СУБД, а потім створити її системний каталог. Таким чином, ERwin дозволяє вирішити завдання щодо перенесення структури даних з одного серверу на інший. Якщо в логічній моделі не має значення, який конкретно тип даних має атрибут, то у фізичній моделі важливо описати всю інформацію про конкретні фізичні об'єкти – таблиці, стовпчики, індекси, процедури і т.д.

Інтерфейс ERwin виконаний у стилі Windows-додатків, достатньо простий і виконаний інтуїтивно. (рис. 8.1). Значення кнопок панелі інструментів наведено в табл. 8.1, розглянемо їх.

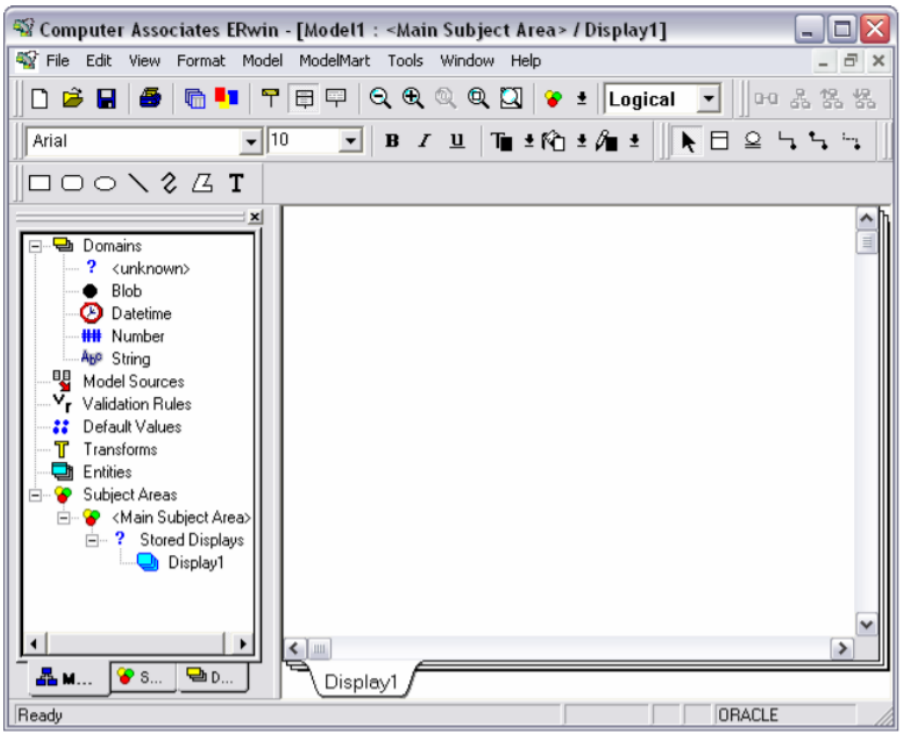
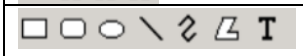



*Рис. 8.1.* Вікно відображення логічної моделі

Таблиця 8.1

*Базова панель інструментів*

| Кнопки | Призначення кнопок |
|---|---|
|  | Створення, відкриття, зберігання і друк моделі |
|  | Зміна рівня перегляду моделі: рівень сутностей, рівень атрибутів і рівень визначень |
|  | Зміна масштабу перегляду моделі |
|  | Перемикання між областями моделі – SubjectArea |
|  | Діалог для генерації звітів по моделі |
|  | Палітра інструментів |
|  | Панель інструментів FontandColor Toolbar |
|  | Панель Суперклас – підклас |
|  | Панель для малювання графічних об’єктів |

Для створення типів сутностей моделі і зв’язування їх між собою використовується палітра інструментів, зображена на рис. 8.2. Залежно від рівня відображення моделі палітра інструментів має різний вигляд. На логічному рівні палітра інструментів має певні значення кнопок (табл. 8.2).

На фізичному рівні палітра інструментів має:

1. Замість кнопки категорій – кнопку внесення зображень (view).

2. Замість кнопки зв’язку «багато–до–багатьох» – кнопку зв’язків зображень;

3. Запустіть Computer Associates ERwin. Якщо з’явиться:

- ModelMart Connection Manager, то закрийте його і натисніть на кнопку Cancel;

- діалогове вікно Computer Associates ERwin, то оберіть пункт Create a new model і натисніть OK (рис. 8.3).

4. У діалоговому вікні «Create Model – Select Template», що з’явилося, оберіть пункт Logical/Physical і натисніть OK (рис. 8.4).

Якщо не зрозуміло, як виконати конкретну дію, можна викликати допомогу: клавіша F1 або меню Help.


*Рис. 8.2.* Палітра інструментів

Таблиця 8.2.

*Палітра інструментів*

| Кнопки | Призначення кнопок | Опис |
|---|---|---|
| | Курсор | Кнопка курсора (режим миші) – в цьому режимі можна встановити фокус на будь-якому об’єкті моделі |
| | Сутність | Кнопка внесення сутності – для внесення сутності треба клацнути лівою кнопкою миші по кнопці внесення сутності і один раз по вільному місці на моделі. Для редагування сутностей або інших об’єктів моделі необхідно перейти в режим курсора |
| | Категорія | Категорія, або категоріальний зв’язок, - спеціальний тип зв’язку між сутностями, який буде розглянуто далі. Для встановлення категоріального зв’язку треба клацнути лівою кнопкою миші по кнопці категорії, потім один раз клацнути по сутності – родовому предку, і за цим – по сутності – нащадку. |
| | Ідентифікуючий зв’язок | Зв’язок між незалежними і залежними сутностями. |
| | Зв’язок «Багато-до-багатьох» | Екземпляр однієї сутності може бути пов’язаний з багатьма екземплярами іншої сутності і навпаки (можлива тільки на рівні логічної моделі). |
| | Неідентифікуючий зв’язок | Зв’язок між незалежними сутностями. |

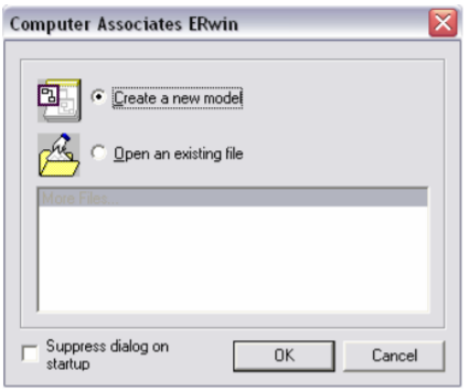



*Рис. 8.3.* Діалогове вікно Computer Associates Erwin

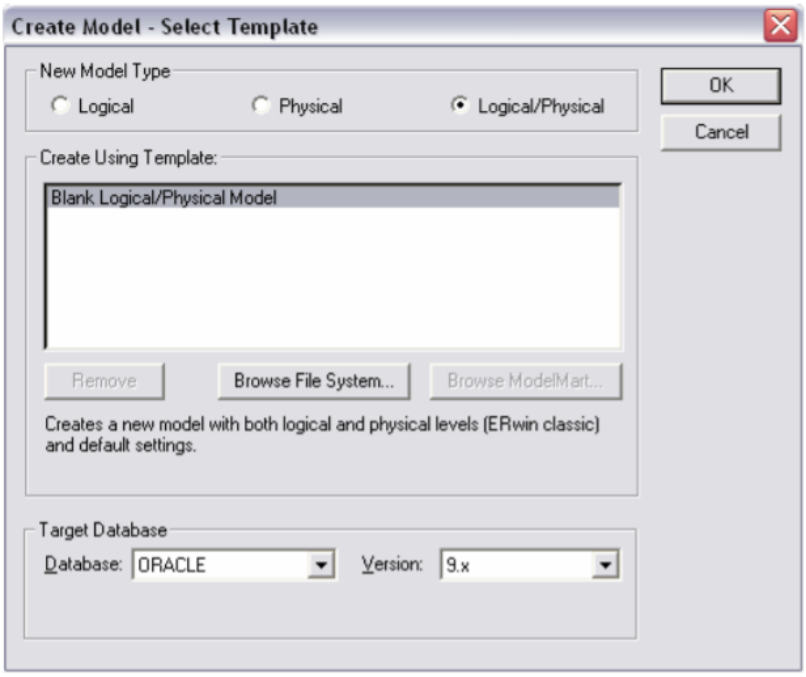



*Рис. 8.4.* Діалогове вікно Create Model – Select Template

## 2. Створення логічної моделі даних

**1. Рівні логічної моделі.** Розрізняють три рівні логічної моделі, що відрізняються глибиною зображення інформації про дані:

1. Діаграма сутність-зв'язок (Entity Relationship Diagram, ERD);
2. Модель даних, заснована на ключах (Key Based model, KB);
3. Повна атрибутна модель (Fully Attributed model, FA).

*Діаграма сутність–зв'язок* подає модель даних верхнього рівня. Вона включає сутності і взаємозв'язки, які відображають базові бізнес-правила прикладної області. Така діаграма не є дуже деталізованою, в ній містяться основні сутності та зв'язки між ними, які задовольняють основні вимоги ІС. Діаграма сутність–зв'язок може включати в себе зв'язки «багато – до – багатьох» і не включати опис ключів. Зазвичай, ERD використовують для презентацій і обговорень структури даних з експертами прикладної області.

*Модель даних, заснована на ключах*, є більш детальним поданням даних. Вона включає опис всіх сутностей і первинних ключів, призначена для подання структури даних і ключів, які відповідають прикладній області.

Повна атрибутна модель є найбільш детальним зображенням структури даних: вона відображає дані в третій нормальній формі, включає всі сутності, атрибути і зв'язки.

ERwin має декілька рівнів відображення діаграми: рівень 1)сутностей, 2) атрибутів, 3) визначень, 4) первинних ключів, 5) іконок.

Переключитися між першими трьома рівнями можна із використанням кнопок панелі інструментів. Переключитися на інші рівні відображення можна за допомогою контекстного меню, яке з'являється, якщо натиснути праву кнопку миші на будь-якому місці діаграми, де не містяться об'єкти моделі. В

контекстному меню треба обрати пункт Display Level і потім необхідний рівень відображення.

**2. Сутності та атрибути**. Основними компонентами діаграми ERwin є сутності, атрибути та зв'язки. Сутність можна визначити як об'єкт, подія або концепція, інформація про яку повинна зберігатися. Сутності повинні мати назву з чітким смисловим значенням, фактично це ім'я її екземпляра. Кожен екземпляр індивідуальний і повинен відрізнятися від всіх інших екземплярів. Атрибут відображає певну властивість об'єкта. З точки зору БД (фізична модель) сутності відповідає таблиця, екземпляру сутності – рядок в таблиці, а атрибуту – стовпчик таблиці.

Entity Editor в контекстному меню для сутності дозволяє визначити ім'я, опис, коментарі, іконку. Для опису атрибутів сутності обирається пункт Attribute Editor. Тут можна вказати ім'я нового атрибуту та домен, який буде використовуватися при визначенні типу стовпчика на рівні фізичної моделі.

Атрибути повинні іменуватися в однині, мати чітке смислове значення і бути досить важливими для того, щоб їх застосовувати в моделі. Назва сутності в однині полегшує читання моделі в майбутньому. Кожен атрибут повинен бути визначеним (закладка - Definition), при цьому бажано уникати циклічних визначень і похідних атрибутів. Для внесення додаткових коментарів і визначень до сутності існують властивості, визначені користувачем (UDP). Дотримання цього правила дозволяє частково вирішити проблему нормалізації даних вже на етапі визначення атрибутів.

Кожен атрибут зберігає інформацію про конкретну властивість сутності, кожен екземпляр сутності повинен бути унікальним. Атрибут або група атрибутів, які однозначно ідентифікують екземпляр сутність, називається первинним ключом. Атрибути первинного ключа на діаграмі не вимагають спеціального позначення – це ті атрибути, які знаходяться в списку атрибутів вище горизонтальної лінії.


*Рис. 8.5.* Загальна діаграма діяльності амбулаторії сімейної медицини (модель бізнес-прецедентів)

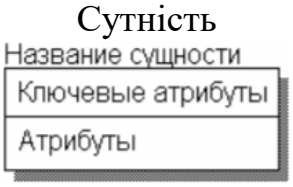
Продовжимо розгляд практичного моделювання на прикладі діяльності амбулаторії сімейної медицини, головною задачею якої є обслуговування пацієнтів (а саме надання медичної допомоги).

**Приклад 8.1**. Серед основних *процедур* амбулаторії сімейної медицини виділимо такі:

1. Лікарі, технічний персонал, зовнішній медичний заклад надає медичну допомогу пацієнтам.
2. Створення страхових полісів пацієнтам, за допомогою страхових компаній та юристів.
3. Виконання постанов від урядових установ страховими компаніями та медичним закладом.
4. Обслуговування пацієнтів.

Для початку зобразимо модель бази даних (БД) для амбулаторії сімейної медицини. Модель повинна містити десять таких сутностей: лікар, пацієнт, діагноз, технічний персонал, зовнішня медична установа, юрист, страхова компанія, страховий поліс, урядова установа, постанова (розпорядження, рішення, ухвала).

Клацніть на кнопку  на робочому полі програми. З’явиться віконце сутності з назвою E/1. Віконце сутності розділено на дві частини. У верхню частину входять атрибути, які є ключами сутності, а в нижню – не ключові атрибути. Подвійне клацання по сутності відкриває діалогове вікно Attributes (рис. 8.6).



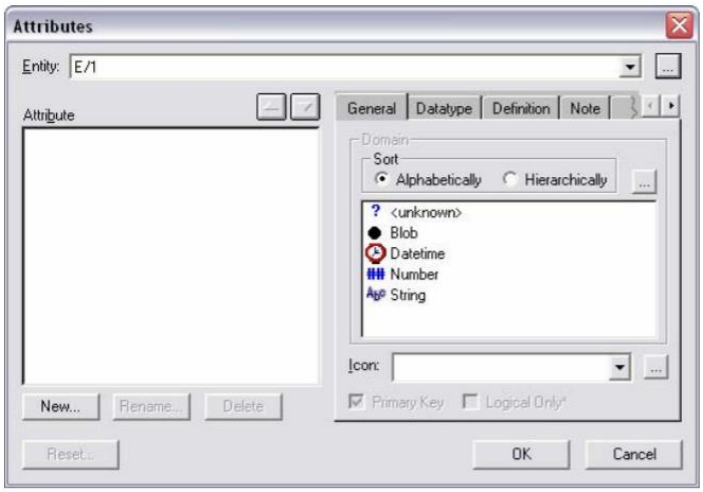




*Рис. 8.6.* Діалогове вікно Attributes

У верхньому правому кутку вікна знаходиться кнопка, яка відкриває діалогове вікно Entities (рис. 8.7), де в полі Name є можливість змінити ім’я сутності. Змініть ім’я сутності на «Лікар». Після закінчення процесу введення натисніть кнопку OK. Натисканням на кнопці New... у лівому нижньому кутку вікна відкривається діалогове вікно New Attribute створення нового атрибуту сутності (рис. 8.8).

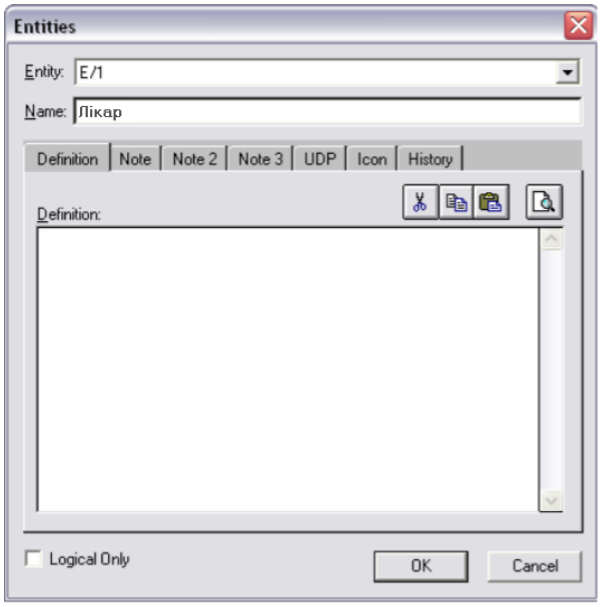



*Рис. 8.7.* Діалогове вікно Entities

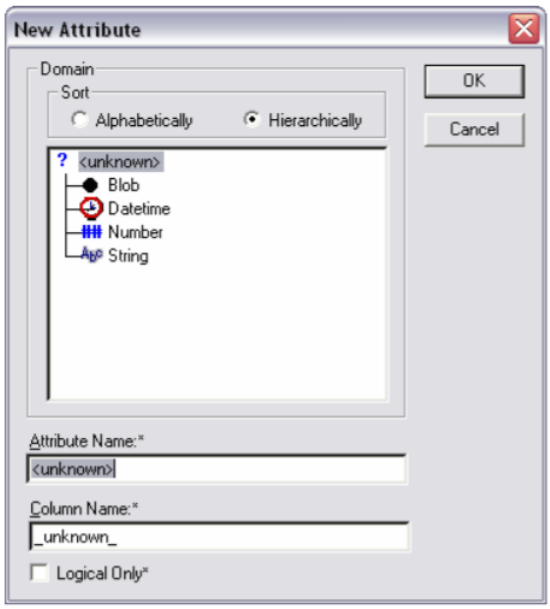



*Рис. 8.8.* Діалогове вікно New Attribute

В полі Attribute Name:* введіть назву атрибута, наприклад для сутності «Лікар» - назва атрибута – «ПІБ_Лікар»(можливо ідентифікувати по іншому). У вікні Domain оберіть тип введених даних. Для звершення процедури введення натисніть OK. Для задання первинного ключа у вікні Attributes для обраного атрибуту «ПІБ_Лікар» встановіть галочку Primary Key.

Можливий також й інший спосіб введення атрибутів в поля сутності. Щоб заповнити віконце сутності атрибутами, треба обрати сутність лівою кнопкою миші і натиснути клавішу «Tab». Після введення назви атрибута треба натиснути «Enter», якщо необхідно залишитися у верхній частині і продовжити заповнювати ключові атрибути, або «Tab», якщо необхідно приступити до заповнення інших атрибутів. Клавіша «Tab» переміщує курсор між трьома полями введення: назвою сутності, ключовими атрибутами і не ключовими атрибутами. Задайте ще декілька атрибутів в сутності «Лікар»: «Спеціалізація», «Стаж_роботи», «Кількість_робочів_днів» тощо. Результат зображено на рис. 8.9.

Одним із запропонованих способів створіть і заповніть всі необхідні сутності моделі амбулаторії сімейної медицини, описані в табл. 8.3.

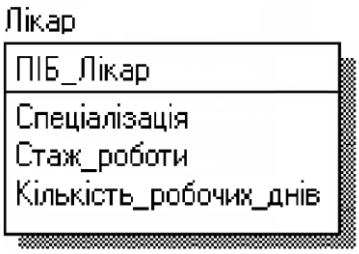



*Рис. 8.9.* Сутність «Лікар»

Таблиця 8.3.

*Сутності і атрибути амбулаторії сімейної медицини*

| *Сутність* | *Атрибути* |
|---|---|
| Лікар | ПІБ_Лікар (РК)<br>Спеціалізація<br>Стаж_роботи<br>Кількість_робочих_днів |
| Пацієнт | ПІБ_Пацієнт (РК)<br>Вік<br>Стать<br>Кількість_днів_у_лікарні<br>Процедури<br>Та ін. |
| Діагноз | Назва_діагноз(РК)<br>Дата_встановленя |
| Технічний персонал | ПІБ_правцівника (РК) |
| Зовнішня медична установа | Назва_мед_установи (РК) |
| Юрист | ПІБ_Юриста (РК) |
| Страхова компанія | Назва_СК (РК) |
| Страховий поліс | Номер_СП(РК) |
| Урядова установа | Назва_урядової_установи (РК) |
| Постанова | Номер_постанови(РК) |

**8.2.3. Зв'язки.** Зв'язок є логічним співвідношенням між сутностями. Кожен зв'язок повинен іменуватися дієсловом або дієслівною фразою (Relationship Verb Phrases). Ім'я зв'язку полегшує сприйняття діаграми (наприклад, діаграма на рис. 8.10). За замовчуванням ім'я зв'язку на діаграмі не зображується. Для відображення назви треба у контекстному меню на вільному місті діаграми обрати пункт Display Option/relationship та увімкнути опцію Verb Phrase.

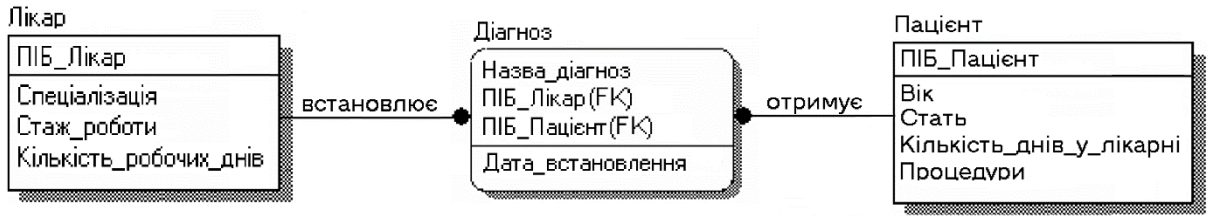



*Рис. 8.10.* Приклад зв'язків

На логічному рівні можна встановити ідентифікуючий зв'язок «один – до багатьох», зв'язок «багато – до– багатьох» і неідентифікуючий зв'язок «один – до багатьох». Тип сутності визначається її зв'язком з іншими сутностями.

Розрізняють залежні і незалежні сутності.

Ідентифікуючий зв'язок встановлюється між незалежною (батьківський кінець зв'язку) і залежною (дочірній кінець зв'язку) сутностями. Коли малюють

ідентифікуючий зв’язок ERwin автоматично перетворює дочірню сутність в залежну. Залежна сутність зображується прямокутником з округленими кутами.

Розрізняють декілька типів залежних сутностей.

1. *Характеристична* – залежна дочірня сутність, яка пов’язана тільки з одною батьківською і за змістом зберігає інформацію про характеристики батьківської сутності.

2. *Асоціативна* – сутність, пов’язана з декількома батьківськими сутностями. Така сутність містить інформацію про зв’язки сутностей.

Іменована – це окремий випадок асоціативної сутності, що не має власних атрибутів (тільки атрибути батьківських сутностей, які мігрували в якості зовнішнього ключа).

3. *Категоріальна* – дочірня сутність в ієрархії успадкування.

При встановленні неідентифікуючого зв’язку атрибути первинного ключа батьківської сутності автоматично переміщуються у зміст первинного ключа дочірньої сутності і зображується в дочірній сутності як зовнішній ключ (FK). Ця операція називається міграцією атрибутів. В подальшому, при генерації схеми БД, атрибути первинного ключа отримують признаки NOT NULL, що значить неможливість внесення запису до таблиці «Діагноз» без інформації про «ПІБ_Лікар».

При встановленні неідентифікуючого зв’язку дочірня сутність залишається незалежною, а атрибути первинного ключа батьківської сутності мігрують у зміст неключових компонентів батьківської сутності.

Ідентифікуючий зв’язок зображується на діаграмі суцільною лінією з товстою точкою на дочірнім кінці зв’язку, неідентифікуючий зв’язок – пунктирною.

У вкладці General меню Relationship Editor можна задати потужність, ім’я і тип зв’язку.

*Потужність зв’язків* (Cardinality) застосовують для позначення відношення кількості екземплярів батьківської сутності до кількості екземплярів дочірньої.

Розрізняють чотири типи сутності:

- загальний випадок, коли одному екземпляру батьківської сутності відповідає 0,1 або багато екземплярів дочірньої сутності (не позначається символом);

- одному екземпляру батьківської сутності відповідає 1 або багато екземплярів дочірньої сутності (позначається символом P);

- одному екземпляру батьківської сутності відповідає 0 або 1 екземпляр дочірньої сутності(позначається символом Z);

- одному екземпляру батьківської сутності відповідає заздалегідь вказана кількість екземплярів дочірньої сутності (позначається цифрою точної відповідності).

За замовчуванням символ, що позначає потужність зв’язку, не зображується на діаграмі. Для відображення імені треба в контекстному меню

діаграми (на вільному місці) обрати пункт Display Options/Relationship, а потім увімкнути опцію Cardinality.

*Ім'я зв'язку* (Verb Phrase) – фраза, що характеризує відношення між батьківською та дочірніми сутностями. Для зв'язку «один – до – багатьох», неідентифікуючого або ідентифікуючого, достатньо вказати ім'я, що характеризує відношення від батьківської до дочірньої сутності (Parent-to-Child). Для зв'язку «багато – до – багатьох» треба вказати імена як Parent-to-Child, так і Child-to-Parent.

Зв'язок «багато – до – багатьох» є можливим тільки на логічному рівні. При переході на фізичний рівень ERwin автоматично перетворить зв'язок «багато – до – багатьох», додаючи нову таблицю і встановлюючи два нових зв'язки «один – до – багатьох» від попередніх до нової таблиці.

*Ієрархія успадкування* – особливий тип об'єднання сутностей, які розділяють загальні характеристики. Зазвичай ієрархію успадкування створюють, коли декілька сутностей мають загальну за змістом атрибути, або коли сутності мають загальні за змістом зв'язки, або коли це диктується бізнес-правилами.

Для кожної категорії можна вказати дискримінатор – атрибут родового предка, що показує, як відрізнити одну категоріальну сутність від іншої.

Ієрархії категорій діляться на два типи – повні і неповні. В повній категорії одному екземпляру родового предка обов'язково відповідає екземпляр в будь-якому нащадку. Якщо категорія ще не побудована повністю і в родовому предку можуть існувати екземпляри, які не мають відповідних екземплярів у нащадків, то така категорія буде неповною.

У нашому прикладі, один лікар може встановити декілька діагнозів та декілька діагнозів може бути у одного пацієнта (наприклад, діагноз «Порок серця» та діагноз «ГРВІ», що можуть одночасно бути у одного пацієнта), тобто треба застосувати зв'язки «один – до - багатьох»(товста точка на кінці зв'язку означає «багато»). Аналогічно треба розмістить зв'язки між всіма сутностями моделі.

1. Оберіть кнопку  , що означає зв'язок «один – до багатьох» і з'єднайте дві необхідні сутності. Щоб з'єднати, необхідно один раз клацнути на сутність «один», а потім на сутність «багато».

2. Оберіть кнопку  , що означає зв'язок «багато – до – багатьох» і з'єднайте дві необхідні сутності.

3. Оберіть кнопку  , що означає не ідентифікуючий зв'язок, і з'єднайте дві необхідні сутності.

4. Аналогічно з'єднайте всі необхідні сутності, відображаючи взаємозв'язки між ними. Якщо під час встановлення зв'язку у вас з'явилося діалогове вікно ERwin (рис. 8.11), це означає, що назва атрибуту, що мігрує від батьківської сутності, вже існує. Оберіть відповідну опцію міграції і натисніть ОК. В результаті отримаємо логічну модель бази даних амбулаторії сімейної медицини (рис. 8.12).

**4. Ключі.** Кожен екземпляр сутності повинен бути унікальним і відрізнятися від інших атрибутів. *Первинний ключ* (primary key) – це атрибут або група атрибутів, що однозначно ідентифікує екземпляр сутності. Атрибути первинного ключа на діаграмі не потребують спеціального позначення – це ті атрибути, які знаходяться в списку атрибутів вище горизонтальної лінії (рис. 8.5). В одній сутності можуть опинитися декілька атрибутів або наборів атрибутів, що претендують на роль первинного ключа. Такі претенденти називаються *потенціальними ключами* (candidate key). Ключі можуть бути складними, тобто містити декілька атрибутів. Складні первинні ключі не потребують спеціального позначення – це список атрибутів, що розташовані вище горизонтальної лінії.

Щоб стати первинним, потенціальний ключ повинен задовольняти такі вимоги:

1. *Унікальність*. Два екземпляри не повинні мати одинакові значення можливого ключа.

2. *Компактність*. При виборі первинного ключа перевага повинна надаватися більш простим ключам, тобто ключам, що містять меншу кількість атрибутів. Атрибути ключа не повинні містити нульових значень. Значення атрибутів ключа не повинно змінюватися на протязі всього часу існування екземпляра сутності.

Кожна сутність повинна мати хоча б один *потенційний ключ*. Багато сутностей мають лише один потенційний ключ. Такий ключ стає первинним. Деякі сутності можуть мати більше одного можливого ключа. Тоді один з них стає первинним, а інші – альтернативними ключами.

*Альтернативний ключ* (Alternate Key) – це потенційний ключ, що не став первинним.

**5. Нормалізація даних.** *Нормалізація даних* - це процес перевірки і реорганізації сутностей і атрибутів з метою задоволення потреб до реляційної моделі даних.

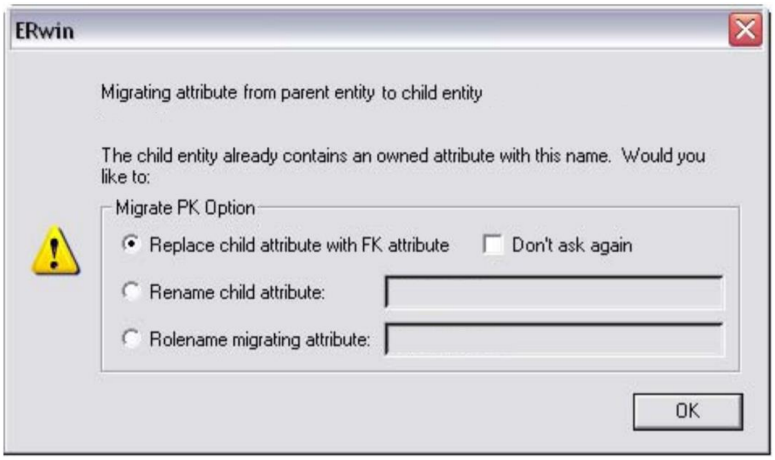



*Рис. 8.11*. Діалогове вікно ERwin при встановленні зв’язку

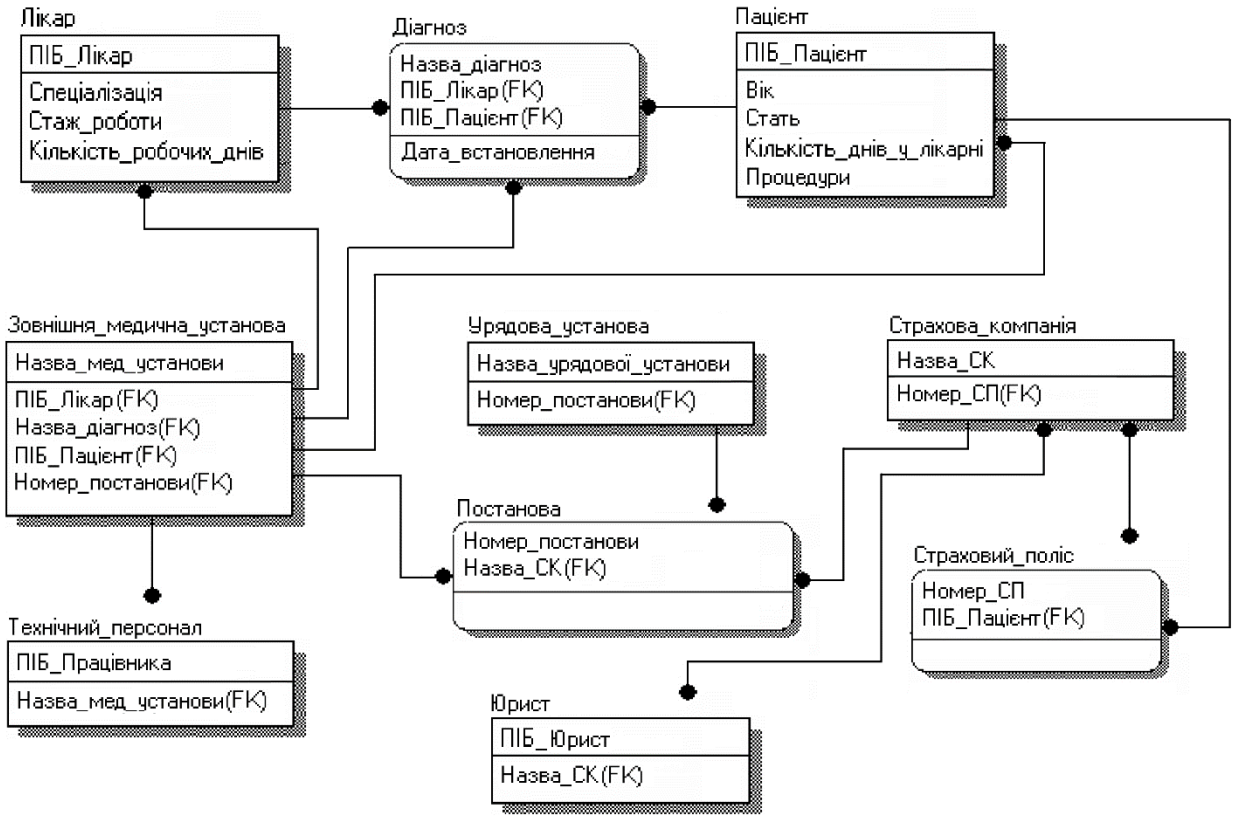



*Рис. 8.12*. Логічна модель бази даних амбулаторії

Нормалізація дозволяє бути впевненим, що кожен атрибут визначений для своєї сутності, а також значно скоротити обсяг пам'яті для зберігання інформації і усунути аномалії в організації зберігання даних. В результаті проведення нормалізації повинна бути створена структура даних, при якій інформація про кожний факт зберігається тільки в одному місці. Процес нормалізації зводиться до послідовного приведення структури даних до нормальних форм - формалізованим вимогам до організації даних. Відомо шість нормальних форм.

*Перша нормальна форма* (1FN). Сутність знаходиться в першій нормальній формі в тому випадку, якщо всі атрибути містять атомарні значення.

Для приведення сутності до першої нормальної формі слід розділити складні атрибути на атомарні. Для цього необхідно:

- створити нову сутність;
- перенести в неї всі «повторювані» атрибути;
- вибрати можливий атрибут для нового PK (або створити новий PK);
- встановити ідентифікуючий зв'язок від колишньої сутності до нової.

*Друга нормальна форма* (2NF). Сутність знаходиться в другій нормальній формі, якщо вона знаходиться в першій нормальній формі і кожен неключовий атрибут повністю залежить від первинного ключа. Друга нормальна форма має сенс тільки для сутностей, що мають складний первинний ключ. Для приведення сутності до другої нормальної форми слід виділити атрибути, які

залежать тільки від частини первинного ключа в нову сутність і встановити ідентифікуючий зв'язок. Друга нормальна форма дозволяє уникнути аномалій при операціях оновлення, вставки і видалення записів.

*Третя нормальна форма* (3NF). Сутність знаходиться в третій нормальній формі, якщо вона знаходиться в другій нормальній формі і ніякий неключовий атрибут не залежить від іншого неключового атрибута. Для приведення сутності до третьої нормальної формі слід створити нову сутність і перенести в неї атрибути з однією і тією ж залежністю від неключового атрибута і встановити зв'язок.

ERwin не містить повного алгоритму нормалізації і не може проводити нормалізацію автоматично, проте його можливості полегшують створення нормалізованої моделі даних. В результаті нормалізації всі взаємозв'язки даних стають правильно визначеними, виключаються аномалії при операціях з даними, модель даних легше підтримувати. Однак часто нормалізація даних не веде до підвищення продуктивності ІС в цілому. Тому з метою підвищення продуктивності при переході на фізичний рівень доводиться свідомо відходити від нормальних форм (проводити операцію денормалізації) для того, щоб скористатися наявними можливостями конкретного сервера або ІС в цілому.

**6. Домени.** Домен можна визначити як сукупність значень, з яких беруться значення атрибутів. Кожен атрибут може бути визначений тільки на одному домені, але на кожному домені може бути визначено безліч атрибутів. У поняття домену входить не тільки тип даних, а й область значень даних. У ERwin домен може бути визначений тільки один раз і використовуватися як в логічній, так і в фізичній моделі. Домени дозволяють полегшити роботу з даними як розробникам на етапі проєктування, так і адміністраторам БД на етапі експлуатації системи. На логічному рівні домени можна описати без конкретних фізичних властивостей. На фізичному рівні вони автоматично отримують специфічні властивості, які можна змінити вручну.

Для створення домену логічної моделі служить діалог Domain Dictionary Editor. Його можна викликати за допомогою меню Edit / Domain Dictionary, розташованому у верхній лівій частині закладки General діалогу Attribute Editor. Кожен домен можна описати завдяки коментарю або властивостям, визначеним певним користувачем (UDP). ERwin має спеціальний інструмент, який полегшує створення нових атрибутів в моделі, використовуючи опис доменів Independent Attribute Browser: цей діалог викликають за допомогою «CTRL + B».

## Теоретичні відомості

## Моделювання даних

**Основні поняття моделі «сутність - зв'язок».** Мета моделювання даних полягає в забезпеченні розробника системи концептуальною схемою бази даних у формі однієї моделі або кількох локальних моделей, які відносно легко можна

відобразити убудь-яку систему баз даних. Найбільш поширеним засобом моделювання даних (предметної області) є модель «сутність-зв'язок» (ERM), яка була вперше введена Пітером Ченом в 1976 р.

Базовими поняттями ERM є сутність, зв'язок та атрибут.

1. *Сутність* (Entity) - реальний або уявний об'єкт, який має істотне значення для розглянутої предметної області. Кожна сутність повинна мати найменування, виражене іменником в однині. Прикладами сутностей можуть бути такі класи об'єктів, як «Постачальник», «Співробітник», «Замовлення». Кожна сутність в моделі зображується у вигляді прямокутника з найменуванням (рис. 8.23).

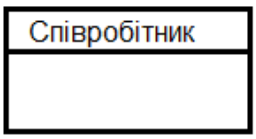



*Рис. 8.23*. Графічне подання сутності

Основний (неформальний) спосіб ідентифікації сутностей - це пошук абстракцій, які описують фізичні або матеріальні об'єкти, процеси і події, ролі людей, організації та інші поняття. Єдиним формальним способом ідентифікації сутностей є аналіз текстових описів предметної області, виділення з описів іменників і вибір їх в якості «кандидатів» на роль абстракцій.

*Екземпляр сутності* - це конкретний представник даної сутності. Наприклад, екземпляром сутності «Співробітник» може бути «Співробітник Іванов». Сутності повинні мати деякі властивості, унікальні для кожного екземпляра цієї сутності. Кожен екземпляр сутності повинен однозначно ідентифікуватися і відрізнятися від усіх інших екземплярів даного типу сутності. Кожна сутність повинна володіти деякими властивостями:

- мати унікальне ім'я; до одного і того ж імені необхідно завжди застосовувати одну інтерпретацію; одна і та ж інтерпретація не може застосовуватися до різних імен, якщо тільки вони не є псевдонімами;

- володіти 1) одним або декількома атрибутами, які або належать сутності, або успадковуються через зв'язок; 2) одним або декількома атрибутами, які однозначно ідентифікують кожен екземпляр сутності.

2. *Атрибут* – це будь-яка характеристика сутності, значима для розглянутої предметної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або вираження стану сутності. Атрибут зображує тип характеристик або властивостей, асоційованих із множиною реальних або абстрактних об'єктів (людей, місць, подій, станів, ідей, предметів тощо). *Екземпляр атрибута* - це визначена характеристика окремого елемента множини. Він визначається типом характеристики та її значенням, названим значенням атрибута. У ERM атрибути асоціюються з конкретними сутностями. Таким чином, екземпляр сутності повинен володіти єдиним певним значенням для асоційованого атрибута.

Найменування атрибута повинно бути виражено іменником в однині (можливо, з прикметниками, які його характеризують). Прикладами атрибутів сутності «Співробітник» можуть бути такі атрибути, як «Табельний номер», «Прізвище», «Ім'я», «По батькові», «Посада», «Зарплата» тощо. Атрибути зображують у межах прямокутника, який визначає сутність (рис. 8.24).

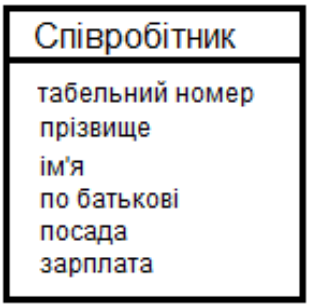



*Рис. 8.24*. Сутність з атрибутами

**Види атрибутів:** 1) простий - складається з одного елемента даних; 2) складений - складається з декількох елементів даних; 3) однозначний - містить одне значення для однієї сутності; 4) багатозначний - містить кілька значень для однієї сутності; 5) необов'язковий - може мати пусте (невизначений) значення; 6) похідний – зображує значення, похідне від значення пов'язаного з ним атрибута.

*Унікальним ідентифікатором* називається ненадлишковий набір атрибутів, значення яких в сукупності є унікальними для кожного екземпляра сутності.

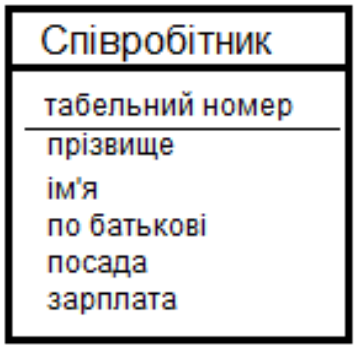



*Рис. 8.25.* Сутність з унікальним ідентифікатором

Ненадмірність полягає в тому, що видалення будь-якого атрибуту з унікального ідентифікатора порушує його унікальність. Сутність може мати кілька різних унікальних ідентифікаторів, які зображують на діаграмі підкресленням (рис. 8.25).

Кожна сутність може мати будь-яку кількість зв'язків з іншими сутностями моделі.

3. *Зв'язок* (Relationship) - пойменована асоціація між двома сутностями, значима для розглянутої предметної області. Зв'язок - це асоціація між сутностями, при якій кожен екземпляр однієї сутності асоційований з довільною кількістю (в тому числі нульовою) екземплярів іншої сутності, і навпаки.

*Ступенем зв'язку* називається кількість сутностей, які беруть участь у зв'язку. Зв'язок ступеня «2» називається бінарним, ступеня N - N-арним. Зв'язок, в якому одна і та ж сутність бере участь в різних ролях, називається рекурсивним (або унарним). Один із можливих варіантів графічного зображення зв'язку показано на рис. 8.26. Пари чисел на діаграмі відображають дві важливі характеристики зв'язку: потужність зв'язку (друге число) і клас належності (перше число).

*Потужністю зв'язку* називають максимальне число екземплярів сутності, яке може бути пов'язане з одним екземпляром даної сутності. Потужність зв'язку може дорівнювати «1», *N* (будь-яке число) і може бути конкретним числом. Потужності зв'язку на рис. 8.26 означають: кожен співробітник може працювати не більше ніж в одному відділі, а в кожному відділі може працювати будь-яка кількість співробітників.

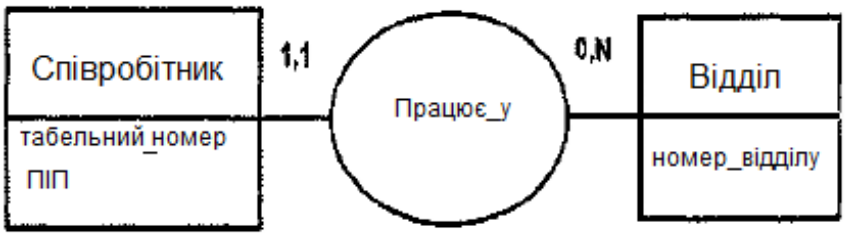



*Рис. 8.26.* Позначення сутностей і зв'язку

*Клас належності* характеризує обов'язковість участі екземпляра сутності у зв'язку. Клас належності може приймати значення «0» (необов'язкова участь - екземпляр однієї сутності може бути пов'язаний з одним або декількома екземплярами іншої сутності, а може бути і не пов'язаний з жодним екземпляром) або «1» (обов'язкова участь - екземпляр однієї сутності повинен бути пов'язаний не менше ніж з одним екземпляром іншої сутності). Класи належності на рис. 8.26 означають: кожен співробітник обов’язково працює в будь-якому відділі, а в деяких відділах може і не бути співробітників.

Зв'язок може мати один з таких трьох типів (в залежності від значення потужності):

1. Один-до-одного (позначається 1: 1), рис. 8.27.
2. Один-до-багатьох (позначається 1: n), рис. 8.26.
3. Багато-до-багатьох (позначається m: n), рис. 8.28.

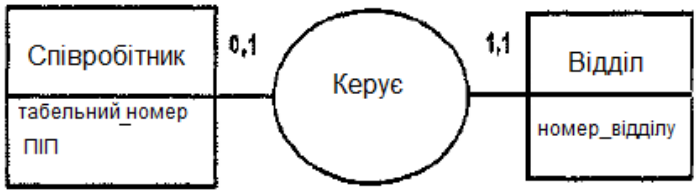



*Рис. 8.27.* Зв'язок типу 1:1


*Рис. 8.28.* Зв'язок типу min

**Види ідентифікаторів**. Існують такі види ідентифікаторів:

1. *Первинний / альтернативний*: сутність може мати кілька ідентифікаторів (рис. 8.29). Один повинен бути основним (первинним), а інші - альтернативними. Первинний ідентифікатор на діаграмі підкреслюється. Альтернативні ідентифікатори випереджаються символами <1> для першого альтернативного ідентифікатора, <2> для другого і т. д.

У концептуальному моделюванні даних відмінність первинних та альтернативних ідентифікаторів зазвичай не використовують. У реляційній моделі, отриманій з концептуальної моделі даних, первинні ключі використовують як зовнішніх ключів. Альтернативні ідентифікатори не копіюють в якості зовнішніх ключів в інші таблиці;

2. *Простий / складений*: ідентифікатор, складений з одного атрибута, є простим, з кількох атрибутів - складеним (рис. 8.29).

3. *Абсолютний / відносний*: якщо всі атрибути, які складають ідентифікатор, належать сутності, то ідентифікатор є абсолютним. Якщо один або більше атрибутів ідентифікатора належать іншій сутності, то ідентифікатор є відносним. Коли первинний ідентифікатор є відносним, сутність визначається як залежна, оскільки її ідентифікатор залежить від іншої сутності. У прикладі на рис. 8.30 ідентифікатор сутності «Рядок-замовлення» є відносним: він включає ідентифікатор сутності «Замовлення», яке зображено підкресленням на рисунку у вигляді «1.1».

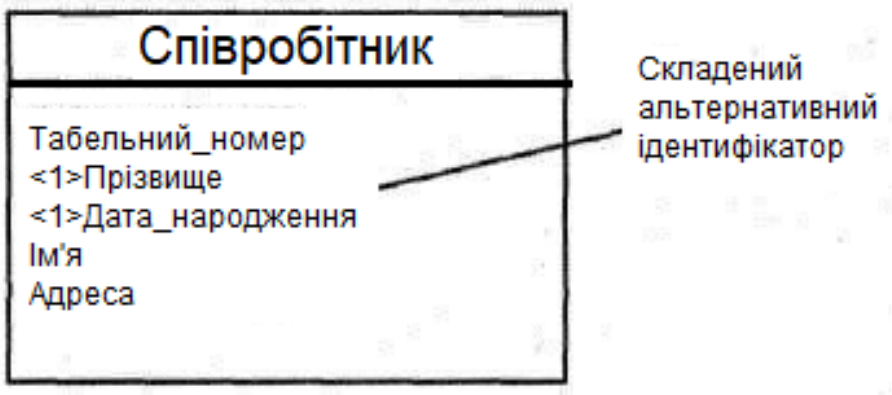



*Рис. 8.29*. Складений альтернативний ідентифікатор

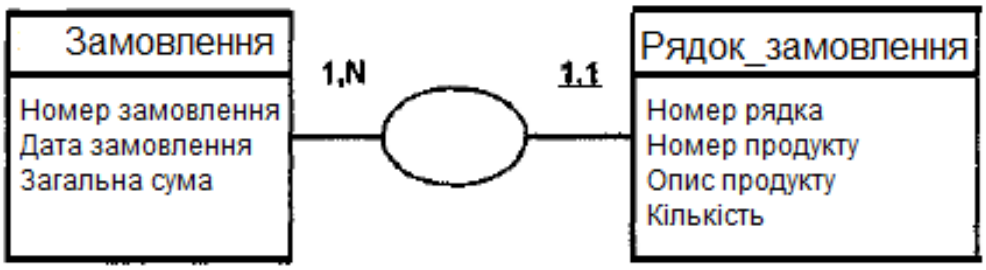



*Рис. 8.30*. Відносний ідентифікатор

**Зв'язки із атрибутами**. Як і сутності, зв'язки можуть мати атрибути. У прикладі на рис. 8.31, щоб знайти оцінку студента, потрібно знати не тільки ідентифікатор студента, а й номер курсу. Оцінка не є атрибутом студента або атрибутом курсу; вона є атрибутом обох цих сутностей. Це є атрибут зв'язку між студентом і курсом, який в прикладі називається «Реєстрація».

Зв'язок між сутностями в концептуальній моделі даних є типом, який зображує множину екземплярів зв'язку між екземплярами сутностей. Щоб ідентифікувати певний екземпляр сутності, використовується ідентифікатор сутності. Для визначення екземплярів зв'язку між сутностями потрібен ідентифікатор зв'язку. Наприклад, в прикладі на рис. 8.31 ідентифікатором зв'язку «Реєстрація» є ідентифікатор студента і номер курсу, оскільки разом вони визначають конкретний екземпляр зв'язку студентів і курсів.

*Зв'язки «супертип-підтип».* У зв'язку «супертип-підтип» (рис. 8.32) загальні атрибути типу визначаються за сутністю-супертипа, сутність-підтип успадковує всі атрибути супертипа. Екземпляр підтипу існує тільки за умови існування визначеного екземпляру супертипа. Підтип не може мати ідентифікатор (він імпортує його із супертипа).

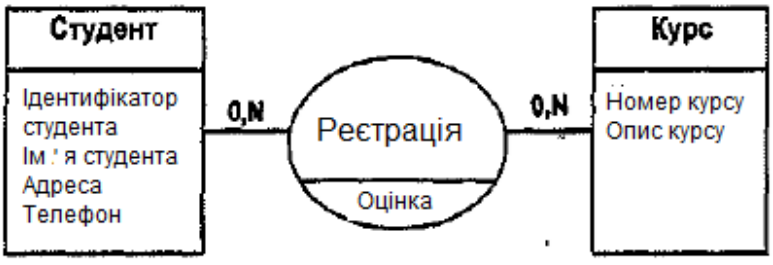



*Рис. 8.31*. Зв'язок із атрибутами

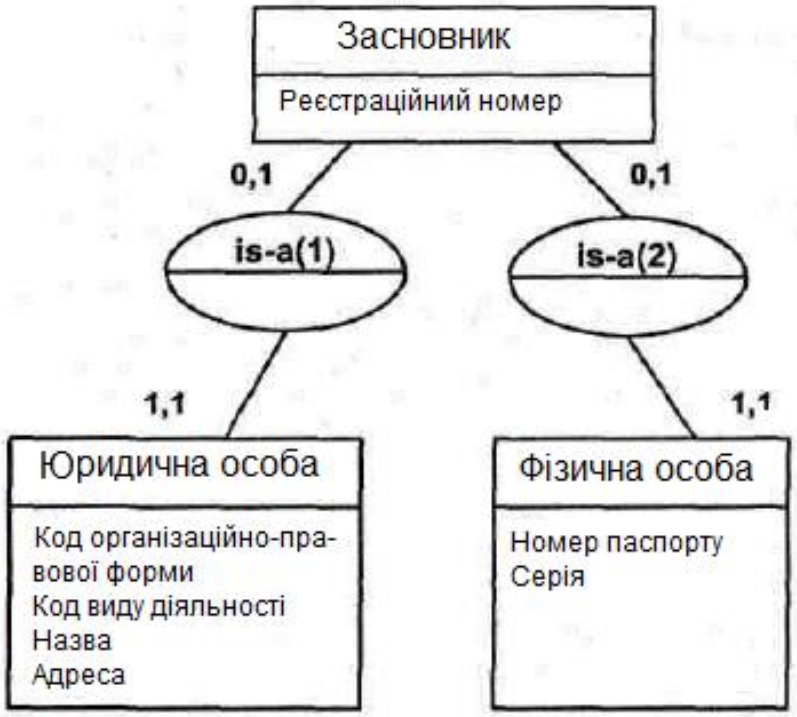



*Рис. 8.32*. Зв'язок «супертип-підтип»

**Приклад нотації моделі «сутність-зв'язок» - метод IDEF1X.** Метод IDEF1X, який входить до стандартів IDEF, використовує різновиди моделі «сутність-зв'язок» і реалізований в ряді поширених CASE-засобів (зокрема, ERwin). Сутність в методі IDEF1X є незалежною від ідентифікаторів, якщо кожен екземпляр сутності можна одназначно ідентифікувати без визначення його відносин з іншими сутностями.

*Сутність називається залежною* від ідентифікаторів, якщо однозначна ідентифікація екземпляра сутності залежить від його ставлення до іншої сутності (рис. 8.33). Кожній сутності присвоюють унікальне ім'я і номер, що розділяють косою рисою "/" і розміщують над блоком. Зв'язок може додатково визначатися за допомогою вказівки потужності (кількості екземплярів сутності-нащадка, який може існувати для кожного екземпляра сутності-батька). У IDEF1X можуть бути виражені такі потужності зв'язків: кожен екземпляр сутності-батька: 1) може мати нуль, один або більше одного пов'язаного з ним екземпляра сутності-нащадка; 2) повинен мати не менше одного пов'язаного з ним екземпляра сутності-нащадка; 3) повинен мати не більше одного

пов'язаного з ним екземпляра сутності-нащадка; 4) повинен бути пов'язаним з деяким фіксованим числом екземплярів сутності-нащадка.

Якщо екземпляр сутності-нащадка однозначно визначається своїм зв'язком із сутністю-батьком, то зв'язок називають ідентифікуючим, в іншому випадку - неідентифікуючим. Зв'язок зображують лінією, яка проводиться між сутністю-батьком і сутністю-нащадком з точкою на кінці лінії у сутності-нащадка (рис. 8.34).


*Рис. 8.33*. Незалежні (а) і залежні (б) від ідентифікатора сутності

Потужність зв'язку може набувати таких значень: N - нуль, один або більше, Z - нуль або один, P - один або більше. За замовчуванням потужність зв'язку приймається рівною N.

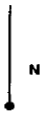



*Рис. 8.34*. Графічне зображення потужності зв'язку

Зв'язок, який ідентифікує, між сутністю-батьком і сутністю-нащадком зображується суцільною лінією (рис. 8.35). Сутність-нащадок у зв'язку, що ідентифікує, є залежним від ідентифікатора сутності. Сутність-батько утакому зв'язку може бути як незалежним, так і залежним від ідентифікатора сутності (це визначається його зв'язками з іншими сутностями). Пунктирна лінія зображує зв'язок, який не ідентифікує (рис. 8.36).

Сутність-нащадок у зв'язку, який не ідентифікує, буде незалежною від ідентифікатора, якщо вона не є також сутністю-нащадком у будь-якому зв'язку, який ідентифікує. Атрибути зображують у вигляді списку імен усередині блоку сутності. Атрибути, які визначають первинний ключ, розміщуються нагорі списку і відділені від інших атрибутів горизонтальній рискою. Сутності можуть мати також зовнішні ключі (Foreign Key), які можуть використовуватися в якості частини або цілого первинного ключа або неключового атрибуту. Зовнішній ключ зображують за допомогою розташування всередині блоку сутності імен атрибутів, після яких слідують букви FK у дужках.

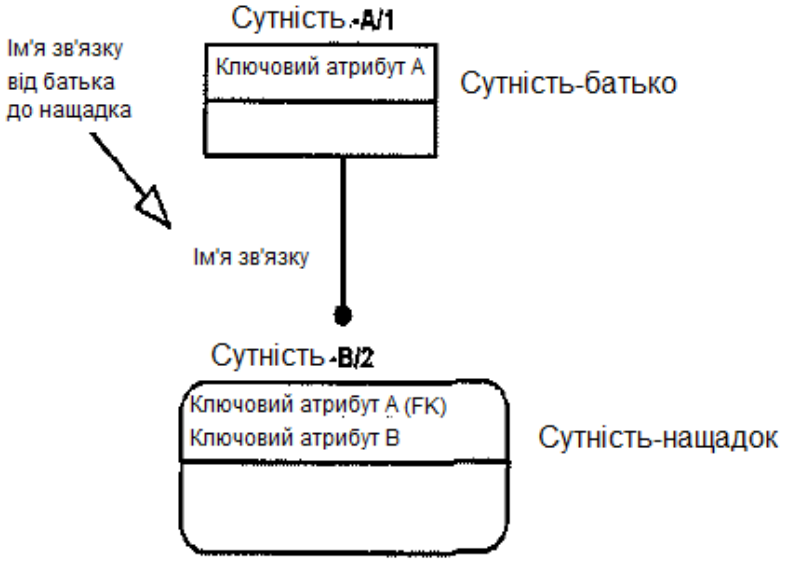



*Рис. 8.35.* Зв'язок, який ідентифікує

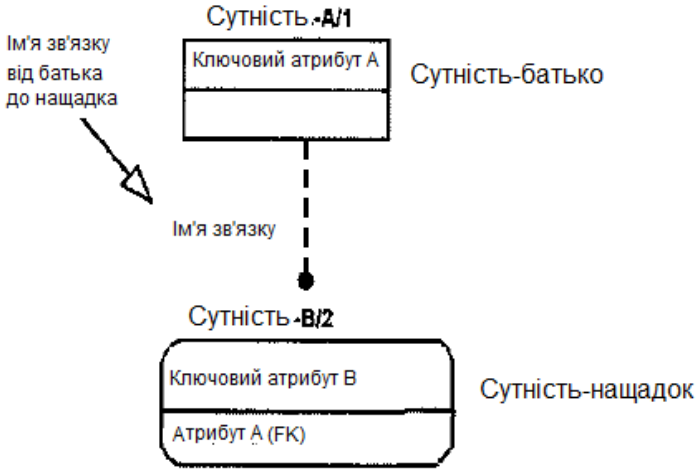



*Рис. 8.36.* Зв'язок, який не ідентифікує

## 9. Створення фізичної моделі даних

*Мета та завдання практикуму:* ознайомитись із механізмом розробки моделі бази даних на фізичному рівні в середовищі ERwin.

План

1. Моделювання інформаційного забезпечення ІС

**Завдання**

1. Обрати приклад ІС та описати відповідну модель бази даних на фізичному рівні в середовищі ERwin:

2. Запустити програму та створити нову модель бази даних в середовищі ERwin.

3. Зберегти модель.

4. Відповісти на контрольні питання.

5. Роздрукувати протокол роботи та показати викладачу.

**Порядок виконання роботи.**

1. Виконати роботу згідно із завданням, прикріпивши скріншоти виконання до кожного пункту.

2. Продемонструвати виконане завдання викладачу.

### Аудиторна робота

### 9.1. Створення фізичної моделі даних

Фізична модель містить всю інформацію, необхідну для реалізації конкретної БД. Розрізняють два рівні фізичної моделі:

- трансформаційна модель;
- модель СУБД.

*Трансформаційна модель* містить інформацію для реалізації окремого проєкту, який може бути частиною загальної ІС і описувати підмножину прикладної області. Ця модель дозволяє проєктувальникам та адміністраторам БД краще уявити, які об'єкти БД зберігаються в словнику даних, і перевірити, наскільки фізична модель задовольняє вимогам до ІС.

*Модель СУБД* автоматично генерується з трансформаційної моделі і є точним відображенням системного каталогу СУБД.

Фізичний рівень подання моделі залежить від обраного сервера. ERwin підтримує більше 20 реляційних і нереляційних БД. Для вибору СУБД слугує редактор Target Server (меню Server / Target Server ... доступний тільки на фізичному рівні).

За замовчуванням ERwin генерує імена таблиць та індексів за шаблоном на основі імен відповідних сутностей і ключів логічної моделі, які в подальшому можуть бути відкориговані вручну. Імена таблиць і стовпців буде згенеровано за замовчуванням на основі імен сутностей і атрибутів логічної моделі. Значення за замовчуванням можна при бажанні змінити шляхом зміни шаблону (Target Server / Table Name Macro) або вручну.

Діалог Target Server дозволяє задати тип даних та опцію NULL для нових стовпців. Тип даних можна обрати в списку Default Datatype, який автоматично

заповнюється типами даних, що підтримує обраний сервер. Група кнопок Default Non-Key Null Option дозволяє вирішити або заборонити значення NULL для неключових стовпців.

Фізична модель відрізняється від логічної тим, що кожному атрибуту сутності необхідно присвоювати тип даних, що вводяться та їх розмір. Так само, у фізичній моделі не припустимі зв'язки «багато– до– багатьох», тому ці зв'язки необхідно виключити або яким-небудь чином замінити на зв'язок «Один до багатьох». Атрибути і назви сутностей необхідно писати англійською або транслітом. Все це треба зробити для того, щоб була можливість імпорту бази даних в СУБД і коректної роботи з нею.

Внесення нових таблиць і зв'язків між ними на фізичному рівні створюються так само як і на логічному рівні. Викликати редактори Table Editor або Column Editor для задання властивостей таблиць і стовпців можна через контекстно-залежне меню для цих таблиць. Всі зміни, зроблені в Table Editor або Column Editor, не позначаються на іменах сутностей і атрибутів, оскільки інформація на логічному і фізичному рівнях в Erwin зберігається окремо. Редактор Table Editor дозволяє задати властивості будь-якої таблиці моделі, відмінні від значення за замовчуванням, в тому числі ім'я таблиці, синоніми, правила валідації, процедури тощо.

Зображення (view) або тимчасові або похідні таблиці, є об'єктами БД, дані в яких не зберігаються постійно, як в таблиці, а формуються динамічно при зверненні до зображення. Зображення не може існувати саме по собі, а визначається тільки в термінах однієї або декількох таблиць. Застосування зображень дозволяє розробнику БД забезпечити кожному користувачеві або групі користувачів свій погляд на дані, що вирішує проблеми простоти використання і безпеки даних. ERwin має спеціальні інструменти для створення і редагування зображень. Палітра інструментів на фізичному рівні містить кнопки внесення зображень і встановлення зв'язків між таблицями і зображеннями. За замовчуванням зображення отримує номер V_n, де n - унікальний порядковий номер представлення. Для редагування зображень необхідно вибрати в контекстно-залежному меню для подання пункт View Editor. Кожній таблиці можна задати необхідну інформацію, яка буде використовуватися в SQL команді для створення зображень.

**1. Побудова фізичної моделі.**

1. Для перемикання між логічною і фізичною моделями даних служить список вибору в центральній частині панелі інструментів ERwin (рис. 9.1). Перейдіть на відображення фізичної моделі. Якщо при перемиканні фізичної моделі ще не існує, вона буде створена автоматично.

2. Якщо існує необхідність розпочати новий проєкт по створенню фізичної моделі, необхідно вибрати File / New ... в головному меню програми (Ctrl + N). У діалоговому вікні Create Model - Select Template необхідно вибрати пункт Physical, а в полі Target Database вибрати СУБД з якою планується працювати і натиснути OK (рис. 9.1).

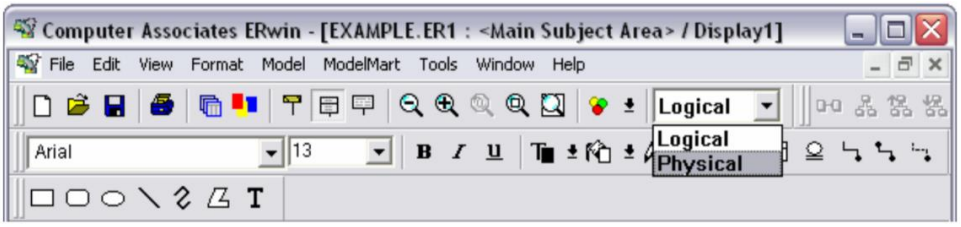



*Рис. 9.1.* Перемикання між логічною і фізичною моделлю

3. Принцип побудови фізичної моделі аналогічний побудові логічної з невеликою поправкою: для кожного атрибута сутності необхідно вказувати тип даних, які вводяться. Для цього у властивостях сутності виберіть пункт Columns ... і закладку ORACLE в діалоговому вікні, що з'явилося (рис. 9.2).

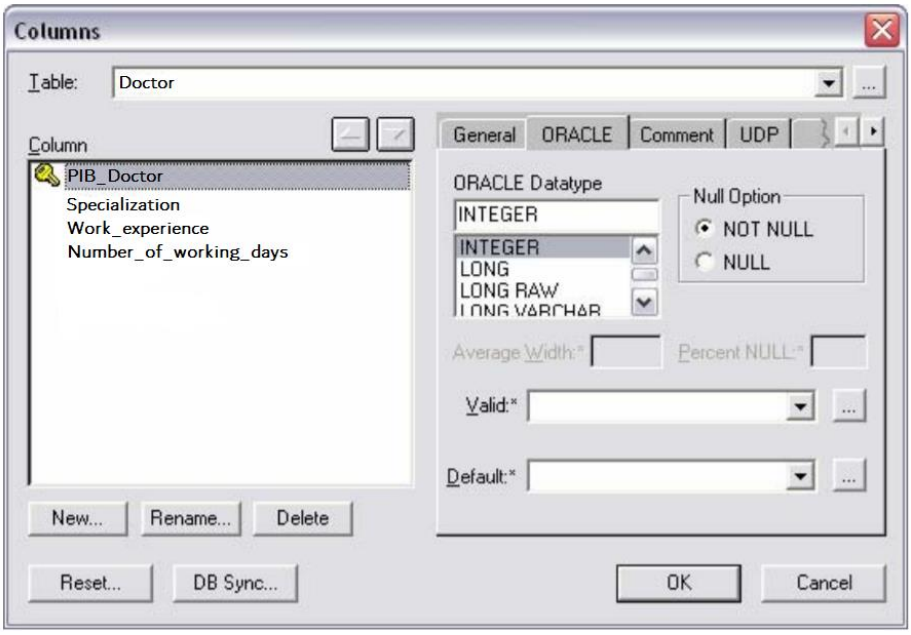



*Рис. 9.2.*Діалогове вікно Columns з закладкою ORACLE

4. Перевірити назви сутностей та атрибутів з урахуванням правила іменування об'єктів в Oracle, при яких ідентифікатор:

- може містити до 30 символів (латинські літери, цифри, символ підкреслення, $, #);
- починається з букви;
- нечутливий до регістру (eMp = EMP);
- унікальний в схемі користувача;
- відмінний від зарезервованих слів.

5. Сутність «Діагноз» фізичної моделі із зазначеними типами даних, які вводяться, повинна мати вигляд, зображений на рис. 9.3.

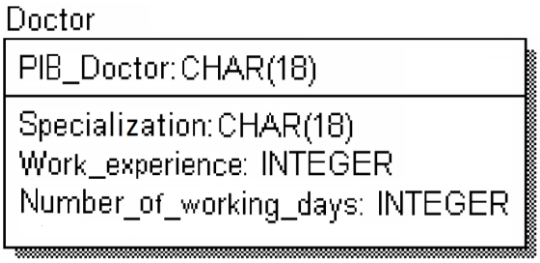



*Рис. 9.3.* Сутність фізичної моделі Доктор

**2. Режим прямого та зворотного проєктування.** Процес генерації фізичної схеми БД з логічної моделі даних називається прямим проєктуванням (Forward Engineering). При генерації фізичної схеми Erwin включає тригери посилальної цілісності, збережені процедури, індекси, обмеження та інші можливості, доступні при визначенні таблиць в обраній СУБД.

Процес генерації логічної моделі з фізичної БД називається зворотним проєктуванням (Reverse Engineering). Erwin дозволяє створити модель даних шляхом зворотного проєктування наявної БД. Після того як модель створена, можна переключитися на інший сервер (модель буде конвертована) і провести пряме проєктування структури БД для іншої СУБД.

**3. Правила валідації і значення за замовчуванням.** ERwin підтримує правила валідації для стовпчиків, а також значення, що привласнюється стовпцям за замовчуванням. Правило валідації задає список допустимих значень для конкретного стовпчика і / або правила перевірки допустимих значень. У список допустимих значень можна вносити нові значення. ERwin дозволяє згенерувати правила валідації відповідно синтаксису обраної СУБД з урахуванням обмежень діапазону або списку значень.

*Значення за замовчуванням* – це значення, яке потрібно ввести в стовпчик, якщо ніяке інше значення не задано явно під час введення даних. З кожним стовпчиком або доменом можна зв'язати значення за замовчуванням. Список значень можна редагувати.

Після створення правила валідації і значення за замовчуванням їх можна присвоїти одній або декільком стовпчикам (або доменам).

1. У вікні Model / Validation Rules задайте максимальне і мінімальне значення і тип валідації. Наприклад, значення, що вводиться в стовпчик «Кількість_робочих_днів» (мається на увазі в місяці), має бути більше 20 днів (за прийнятими правилами), але менше 30 (що є максимальною кількістю днів в місяці). Для опису цього правила можна створити правило валідації з ім'ям «Перевірка днів», яке містить вираз: Днів BETWEEN 20 AND 30 (рис. 9.4).

СУБД повідомить про помилку, якщо вводиться кількість днів, що знаходиться поза межами діапазону.

2. Оберіть пункт меню Model / Default Value.

3. У редакторі Default / Initial Editor задайте значення, які автоматично, за замовчуванням, будуть присвоюватися стовпчику. Наприклад, датою встановлення діагнозу може бути значення за замовчуванням «сьогоднішнє число», тобто автоматично задається, що всі діагнози визначаються в день введення інформації в БД.

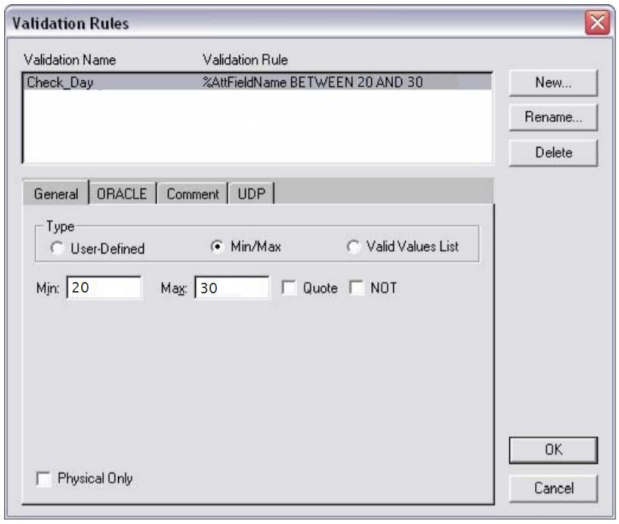



*Рис. 9.4.* Вікно Validation Rules

**4. Індекси.** В БД дані зазвичай зберігаються в тому порядку, в якому їх ввели в таблицю. Багато реляційних СУБД мають сторінкову організацію, при якій таблиця може зберігатися фрагментарно в різних областях диска, причому рядки таблиці розташовуються на сторінках невпорядковано. Такий спосіб дозволяє швидко вводити нові дані, але ускладнює пошук даних.

Щоб вирішити проблему пошуку, СУБД використовують об'єкти, названі індексами. Індекс містить відсортовану по одному стовпчику (або декільком) інформацію і вказує на рядки, в яких зберігається конкретне значення стовпчика. Оскільки значення в індексі зберігаються в певному порядку, при пошуку переглядати потрібно значно менший обсяг даних, що істотно зменшує час виконання запиту. Індекс рекомендується створювати для тих стовпчиків, за якими часто проводиться пошук.

При генерації схеми фізичної БД ERwin автоматично створює індекс на основі первинного ключа кожної таблиці, а також на основі всіх альтернативних ключів і зовнішніх ключів, оскільки ці колонки найбільш часто використовуються для пошуку даних. Можна відмовитися від генерації індексів за замовчуванням і створити власні індекси. Для збільшення ефективності пошуку адміністратор БД повинен аналізувати запити, що часто виконуються, і на основі аналізу створювати власні індекси.

1. Відкрийте редактор Indexes.

2. Змініть, імена індексу та їх визначення так, щоб вони приймали унікальні або дублюючі значення, або змініть порядок сортування даних.

**5. Тригери і збережені процедури.** Тригери і процедури - це іменовані блоки коду SQL, які заздалегідь скомпільовані і зберігаються на сервері для того, щоб швидко проводити обробку запитів, затвердження даних та інші часто виконувані функції. Зберігання та виконання коду на сервері дозволяє створювати код тільки один раз, а не в кожному додатку, що працює з БД. Це економить час при написанні і супроводі програм. При цьому гарантується, що цілісність даних і бізнес-правила підтримуються незалежно від того, який саме клієнтський додаток звертається до даних.

Тригери і процедури не потрібно пересилати по мережі з клієнтського додатку, що значно знижує мережевий трафік. Збереженою процедурою називається іменований набір попередньо скомпільованих команд SQL, який може викликатися з клієнтського додатку або з іншої збереженої процедури.

*Тригером* називають процедуру, яка виконується автоматично як реакція на подію.

Тригер посилальної цілісності - це вид тригера, який використовують для підтримки цілісності між двома таблицями, які пов'язані між собою. Якщо рядок в одній таблиці вставляється, змінюється або видаляється, то тригер посилальної цілісності повідомляє СУБД, що саме потрібно робити з тими рядками в інших таблицях, у яких значення зовнішнього ключа збігаються зі значенням первинного ключа вставленого рядка (зміненого або видаленого рядка).

Для генерації тригерів ERwin використовує механізм шаблонів - спеціальних скриптів, які використовують макрокоманди. При генерації коду тригера замість макрокоманд підставляються імена таблиць, стовпчиків, змінні та інші фрагменти коду, які відповідають синтаксису обраної СУБД. Шаблони тригерів посилальної цілісності, які генеруються ERwin за замовчуванням, можна змінювати.

Для створення і редагування збережених процедур ERwin має спеціальні редактори, які є аналогічними редакторам, що використовуються для створення тригерів. На відміну від тригера збережена процедура не виконується у відповідь на якусь подію, її викликають з іншої програми, яка передає на сервер ім'я процедури. Процедура більш гнучка, ніж тригер, оскільки може викликати інші процедури. Їй можна передавати параметри, і вона може повертати параметри, значення і повідомлення.

1. Для таблиці з контекстно-залежного меню виберіть пункт Triggers.

2. У вікні здійсніть операції по створенню і редагуванню тригерів і процедур. Остаточний вигляд фізичної моделі БД показано на рис. 9.5.

3. Для генерації звітів в ERwin існує інструмент Report Builder, створіть за його допомогою власний звіт.

Кожен звіт може бути налаштований індивідуально, дані в ньому можуть бути відсортовані і відфільтровані.

**6. Переведення фізичної моделі в SQL-скрипт для СУБД ORACLE.** ERwin підтримує синхронізацію між логічною моделлю і системним каталогом СУБД на протязі усього життєвого циклу створення БД.

1. Для генерації системного каталогу БД слід обрати пункт меню Tools / Forward Engineering / Schema Generation. У вікні Schema Generation в закладці Options можна задати опції генерації об'єктів БД - тригерів, таблиць, форм подання, стовпців, індексів тощо.

2. Кнопка Preview дозволяє відобразити SQL-скрипт, який створюється ERwin для генерації системного каталогу СУБД.

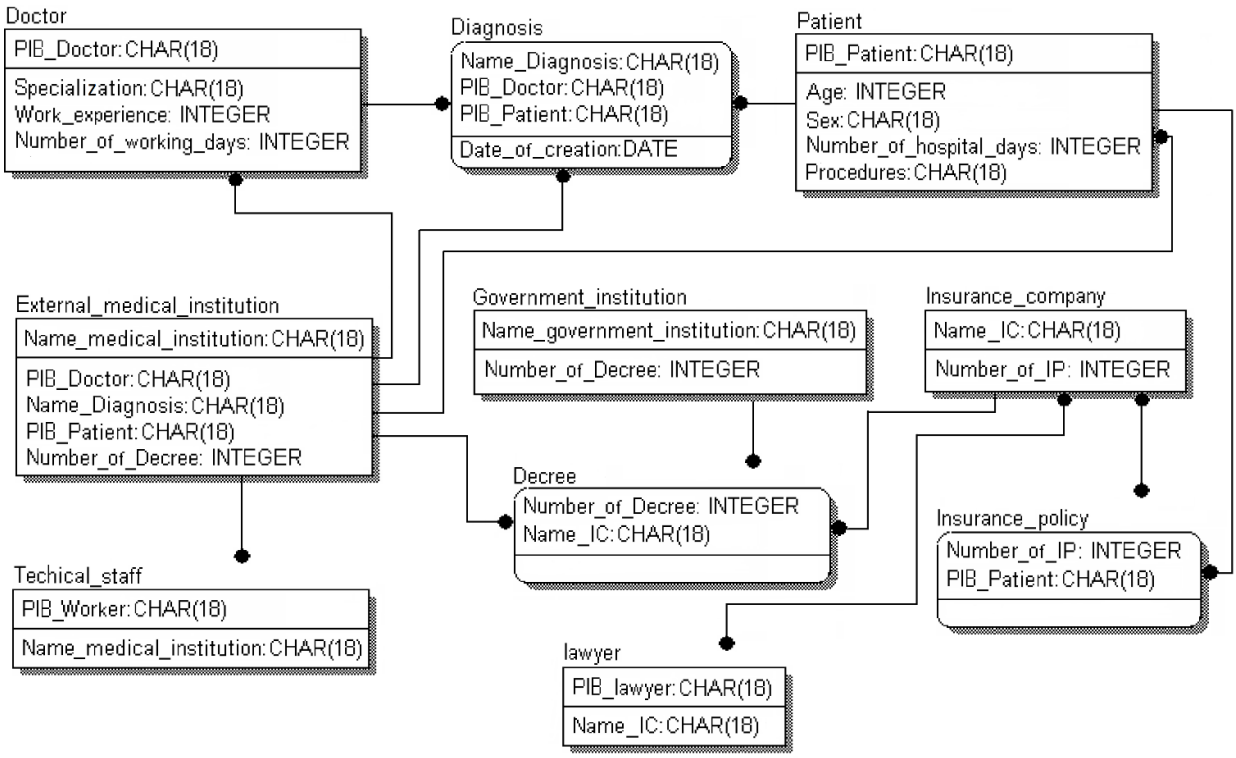



*Рис.* 9.5. Фізична модель бази даних амбулаторії

3. Натискання кнопки Generate призведе до запуску процесу генерації схеми. Виникає діалог зв'язку з БД, встановлюється сеанс зв'язку з сервером і починає виконуватися SQL-скрипт.

4. Кнопка Report зберігає той же скрипт в SQL текстовому файлі. Ці команди можна в подальшому редагувати будь-яким текстовим редактором і виконувати за допомогою відповідної утиліти серверу.

## Питання до розділу 2

1. Вкажіть, до якого рівня деталізації відноситься діаграма сутність-зв'язок, повна атрибутна модель.

3. Вкажіть, до якого рівня деталізації відноситься модель даних, заснована на ключах.

4. Вкажіть, що реалізують діаграми ERD.

5. Вкажіть, що задає правило валідації.

6. Вкажіть базові поняття ERD-діаграми.

7. Вкажіть, яка модель даних зображує дані в третій нормальній формі.

8. Вкажіть, які рівні відображення діаграми має ERwin.

9. Вкажіть, яка модель даних включає опис всіх сутностей і первинних ключів.

10. Назвіть основні етапи проєктування бази даних.

11. Назвіть  принципи побудови логічної моделі даних.

12. Які типи зв'язків використовуються при побудові моделі «сутність-зв'язок»?

13. Наведіть приклади зв'язків, які ідентифікують і не ідентифікують?

14. Що таке потужність зв'язку?

15. Які об'єкти БД генеруються при проєктуванні фізичної схеми?

16. Назвіть відмінності в ідентифікації об'єктів на логічному і фізичному рівні.

17. Як можна здійснити конвертацію БД з однієї СУБД в іншу?

18. Що таке тригер? Які елементи логічної моделі є основними для створення тригерів при прямому проєктуванні?

# РОЗДІЛ 3. ВІЗУАЛЬНЕ МОДЕЛЮВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ В СЕРЕДОВИЩІ RATIONAL ROSE

## 10. Загальна характеристика інструментарію Rational Rose. Початок роботи над проєктом: розробка діаграми варіантів використання

*Мета заняття* – ознайомитися із 1) загальною характеристикою інструментарію; 2) процедурою розробки діаграми варіантів використання.

*Об'єкт вивчення* – інструментарій Rational Rose; діаграма варіантів використання.

План

1. Основні відомості про роботу в середовищі Rational Rose: елементи екрану, різновиди зображення моделі Rose, параметри налаштування відображення.
2. Моделювання бізнес-процесів.
3. Постановка задачі: розробка автоматизованої системи - управління банкоматом,

### Завдання

I. Розглянути загальну характеристику та особливості інтерфейсу інструментарію Rational Rose.

II. Ознайомитися із процедурою розробки діаграми варіантів використання на основі проєкту, який розробляється, у вигляді моделі системи управління банкоматом.

Крок 0. Специфікація вимог до системи (уточнена постановка задачі для системи). Складання глосарія проєкту. Опис додаткових специфікацій

Крок 1. Створення початкової моделі варіантів використання

1. Створення дійових осіб
2. Побудова початкової діаграми варіантів використання.
3. Побудова модифікованої діаграми варіантів використання
4. Додавання описів до варіантів використання
5. Прикріплення файлу до варіанту використання

III. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

Процес роботи над проєктом складається з послідовної розробки канонічних діаграм, які в сукупності становлять інтегровану модель розроблюваної програмної системи. Хоча послідовність розробки діаграм на мові UML не визначена, для цієї мети можна скористатися рекомендаціями раціонального уніфікованого процесу RUP. Цей порядок розробки апробований на реальних проєктах.

Концептуально процес розробки канонічних діаграм полягає в розміщенні на діаграмах відповідних графічних елементів, встановленні відношень між цими елементами, їх специфікації і документування, відповідно

до правил і нотації мови UML. Після побудови моделі, перевірки правильності та узгодженості властивостей її елементів, можна згенерувати текст програмного коду однією з обраних мов програмування. В подальшому цей код слід допрацювати в тому чи іншому середовищі програмування з метою отримання виконуваних модулів програми, орієнтованих на роботу в певному операційному середовищі та в певній обчислювальній платформі.

Процес розробки графічних діаграм нагадує процес роботи в популярних середовищах візуального програмування. Розробник обирає необхідний графічний елемент за допомогою натискання відповідної кнопки на спеціальній панелі інструментів і розміщує цей елемент на робочому аркуші канонічної діаграми. Після цього редагується набір властивостей цього елемента відповідно до розглянутої нотації мови UML.

**1. Розробка діаграми варіантів використання:** в якості прикладу проєкту для розробки служить модель системи управління банкоматом.

Робота над проєктом починається із загального аналізу проблеми і побудови діаграми варіантів використання, яка відображає функціональне призначення проєктованої програмної системи. Як проєкт розглядається модель системи управління банкоматом, яка використовувалася в якості наскрізного прикладу при ілюстрації елементів нотації мови UML. Для новостворюваного проєкту можна скористатися майстром типових проєктів, якщо він встановлений в даній конфігурації.

У нашому випадку слід відмовитися від майстра, в результаті чого з'явиться робочий інтерфейс з чистим вікном активної діаграми класів та ім'ям проєкту untitled за замовчуванням. Для зміни імені проєкту слід зберегти модель у файлі на диску, наприклад, під ім'ям ATM_MOdel.mdl. В цьому випадку зміниться ім'я в рядку заголовка та ім'я папки проєкту в ієрархічному поданні моделі в браузері проєкту.

Засіб Rational Rose дозволяє налаштовувати глобальні параметри середовища, такі як вибір шрифтів і кольору для подання різних елементів моделі. Налаштування шрифтів, кольору ліній і графічних елементів відбувається через операцію головного меню: **Tools → Options** (Інструменти → Параметри).

*Примітка.* Характерною особливістю середовища є можливість роботи з символами кирилиці. Однак, слід зауважити, що при специфікації елементів моделі з подальшою генерацією тексту програмного коду потрібно відразу записувати імена і властивості класів, асоціацій, атрибутів, операцій і компонентів символами тієї мови, яка підтримується відповідною мовою програмування.

Для розробки діаграми варіантів використання моделі необхідно активувати відповідну діаграму у вікні діаграми. Це можна зробити різними способами.

1. Розкрити відображення варіантів використання в браузері (Use Case View) и двічі клацнути на піктограмі **Main** (Головна).

2. Через пункт меню **Browse → Use Case Diagram** (Огляд → Діаграма варіантів використання).

3. Натиснувши відповідну кнопку на стандартній панелі інструментів.

При цьому з'являється нове вікно з чистим робочим аркушем діаграми варіантів використання і спеціальна панель інструментів, що містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми варіантів використання. Призначення окремих кнопок панелі можна дізнатися зі спливаючих підказок. На спеціальній панелі інструментів за замовчуванням присутня тільки частина піктограм елементів, які можуть бути використані для побудови діаграми. Додати кнопки з піктограмами інших графічних елементів, наприклад, таких як бізнес-варіант використання (business use case), бізнес-актор (business actor), співробітник (business worker), або видалити непотрібні кнопки можна за допомогою налаштування спеціальної панелі інструментів. Діалогове вікно налаштування зручно викликати за допомогою пункту контекстного меню Customize (Налаштування) при позиціонуванні курсора на спеціальній панелі інструментів (рис. 10.1).

Для додавання необхідних кнопок на панель слід виділити їх в лівому вікні зі списком піктограм графічних елементів, після чого натиснути кнопку Додати (Добавить) в центрі діалогового вікна. Для видалення непотрібних кнопок з панелі інструментів слід виділити їх в правому вікні зі списком піктограм графічних елементів, після чого натиснути кнопку Видалити в центрі діалогового вікна. Для відновлення набору піктограм за замовчуванням можна натиснути кнопку Скидання (Сброс).

Після налаштування спеціальної панелі інструментів відповідне вікно слід закрити натисканням кнопки Закрити (Закрыть).

*Примітка*. Відкрити діалогове вікно налаштування спеціальних панелей інструментів діаграм можна за допомогою операції головного меню Tools → Options (Інструменти → Параметри), відкривши вкладку Toolbars (Панелі інструментів) і натиснувши відповідну кнопку (наприклад, Use Case diagram) в групі опцій Customize Toolbars (Налаштування панелей інструментів).

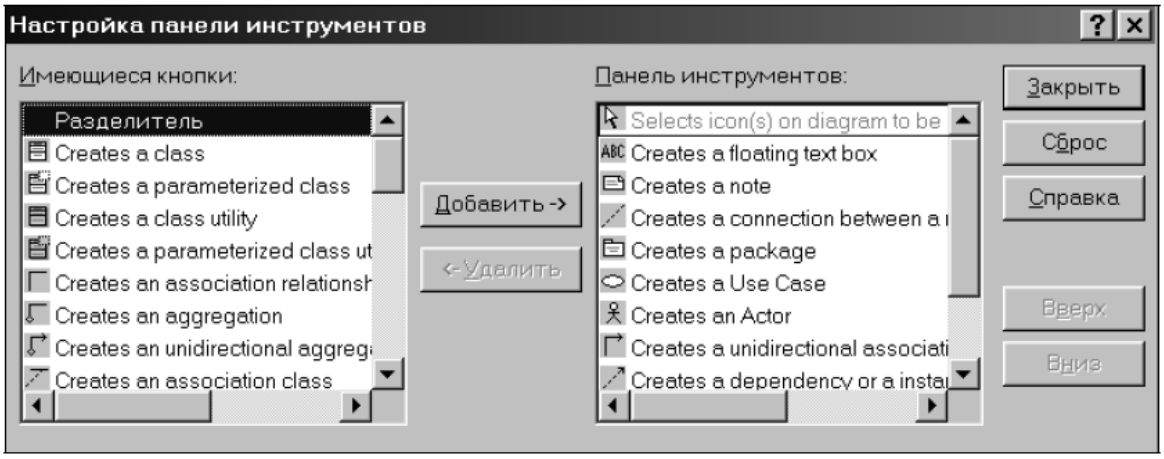



*Рис. 10.1.* Діалогове вікно налаштування спеціальної панелі інструментів діаграми варіантів використання

**Додавання актора на діаграму варіантів використання.** Для додавання актора на діаграму варіанту використання потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми актора на спеціальній панелі інструментів і клацнути лівою кнопкою миші на вільному місці робочого листа діаграми. На діаграмі з'явиться зображення варіанта використання з маркерами зміни його геометричних розмірів і запропонованим за замовчуванням ім'ям (рис. 10.2). Ім'я доданого елемента можна змінити або відразу після розміщення елемента на діаграмі, або при подальшій роботі над проєктом.

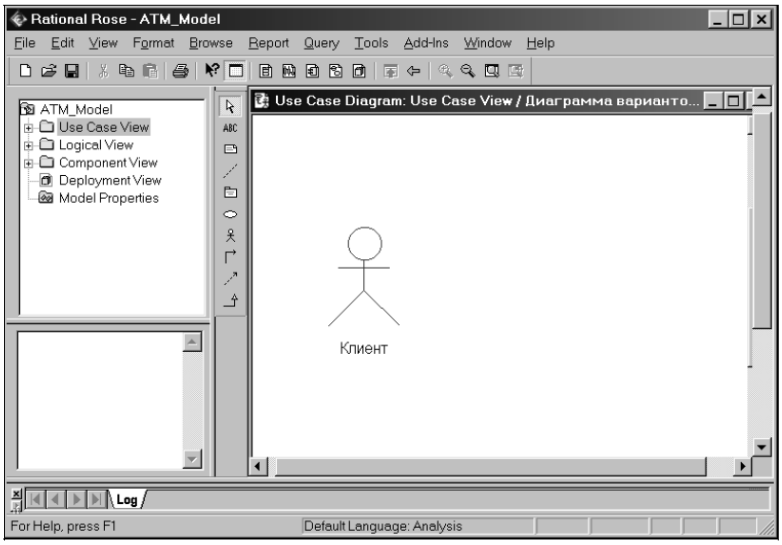



*Рис. 10.2.* Діаграма варіантів використання після додавання на неї актора

Після клацання правою кнопкою миші на вибраному елементі викликається контекстне меню елемента, серед опцій якого є пункт Open Specification (Відкрити специфікацію). В цьому випадку активується діалогове вікно зі спеціальними вкладками, в поля яких можна занести всю інформацію про актора (рис. 10.3).

*Примітка.* Відкрити діалогове вікно специфікації графічного елементу можна також подвійним клацанням лівою кнопкою миші на зображенні цього елемента на діаграмі. У середовищі Rational Rose актор є класом.

Для актора некоректно специфікувати атрибути та операції, оскільки він є зовнішньою по відношенню до розроблюваної системи сутністю.

**Додавання варіанту використання.** Для додавання варіанту використання на діаграму потрібно, за допомогою лівої кнопки миші, натиснути кнопку із зображенням варіанту використання на спеціальній панелі інструментів і клацнути лівою кнопкою миші на вільному місці діаграми. На діаграмі з'явиться зображення варіанту використання з маркерами зміни його геометричних розмірів і запропонованим за замовчуванням ім'ям (рис. 10.4).

Для варіанту використання можна також відкрити діалогове вікно специфікації подвійним клацанням лівою кнопкою миші на зображенні цього елемента.

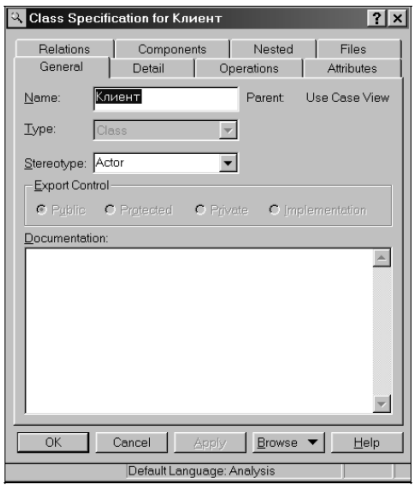



*Рис. 10.3.* Діалогове вікно налаштування властивостей актора

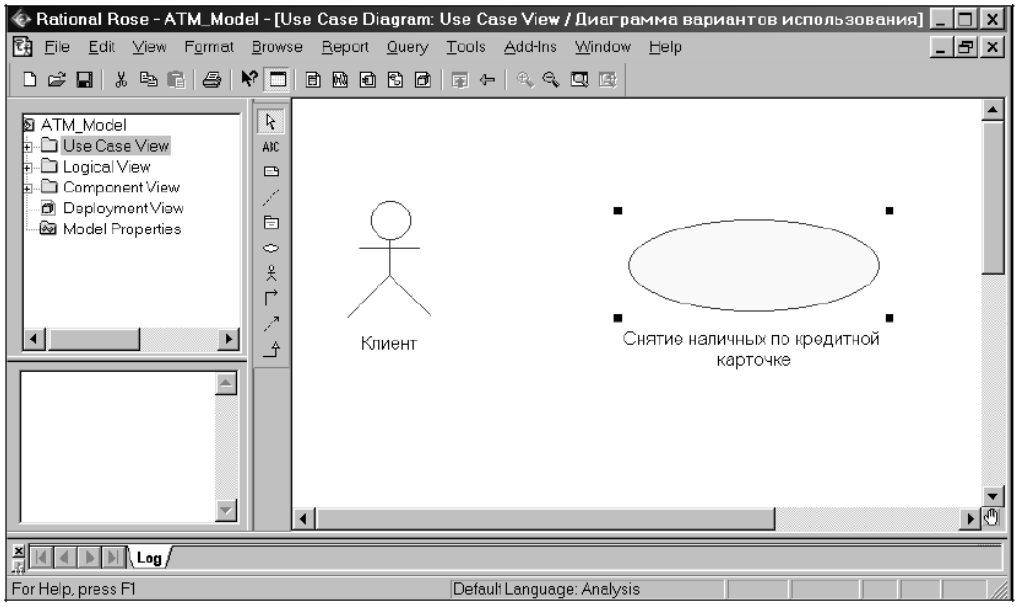



*Рис. 10.4.* Діаграма варіантів використання після додавання до неї варіанта використання

**Додавання асоціації.** Для додавання асоціації між актором і варіантом використання на діаграму потрібно, за допомогою лівої кнопки миші, натиснути на спеціальній панелі інструментів кнопку із зображенням піктограми спрямованої асоціації, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні актора на діаграмі і відпустити її на зображенні варіанту використання. В результаті цих дій на діаграмі з’явиться зображення асоціації, які з’єднують актора з варіантом використання (рис. 10.5).

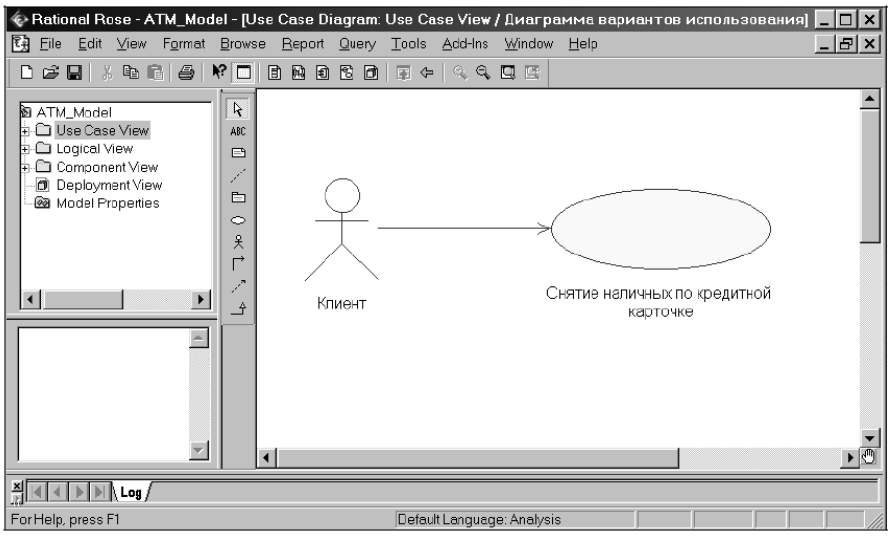



*Рис. 10.5.* Діаграма варіантів використання після додавання до неї асоціації

При необхідності можна зробити спрямовану асоціацію неспрямованою, для чого слід скористатися діалоговим вікном властивостей асоціації (рис. 10.6). Відкрити це вікно можна подвійним клацанням на зображенні лінії асоціації на діаграмі, після чого прибрати позначку рядка вибору Navigable (Навігація) на вкладці Role A Detail (детальні властивості кінцевої точки асоціації A).

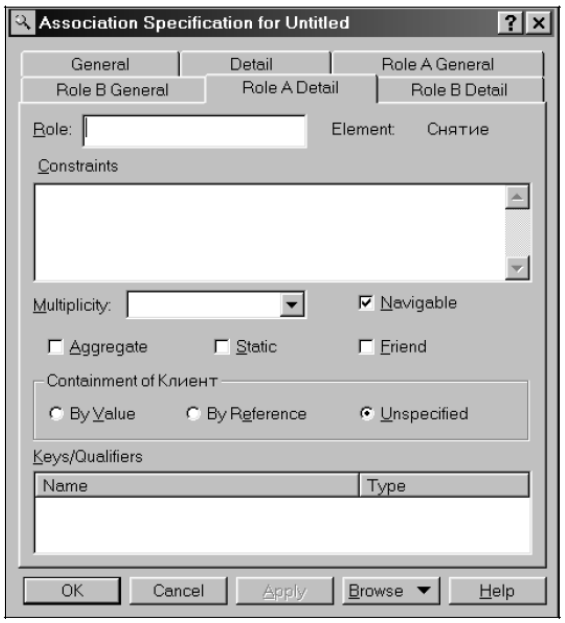



*Рис. 10.6.* Діалогове вікно налаштувань властивостей асоціації

**Додавання відношення залежності.** Для додавання відношення залежності між двома варіантами використання на діаграму необхідно, попередньо, додати варіант використання «Перевірка ПІН-коду». Після чого потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми залежності на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні варіанту використання «Зняття готівки кредитною карткою» на діаграмі і відпустити її

на зображенні варіанту використання «Перевірка ПІН-коду». В результаті цих дій на діаграмі з'явиться зображення відношення залежності, що з'єднує два обраних варіанти використання (рис. 10.7).

Для вказівки додаткового стереотипу «include» для доданого відношення залежності слід вже відомим способом відкрити діалогове вікно властивостей цього відношення (рис. 10.8) і вибрати потрібний стереотип із запропонованого списку, після чого натиснути кнопку Apply (Застосувати).

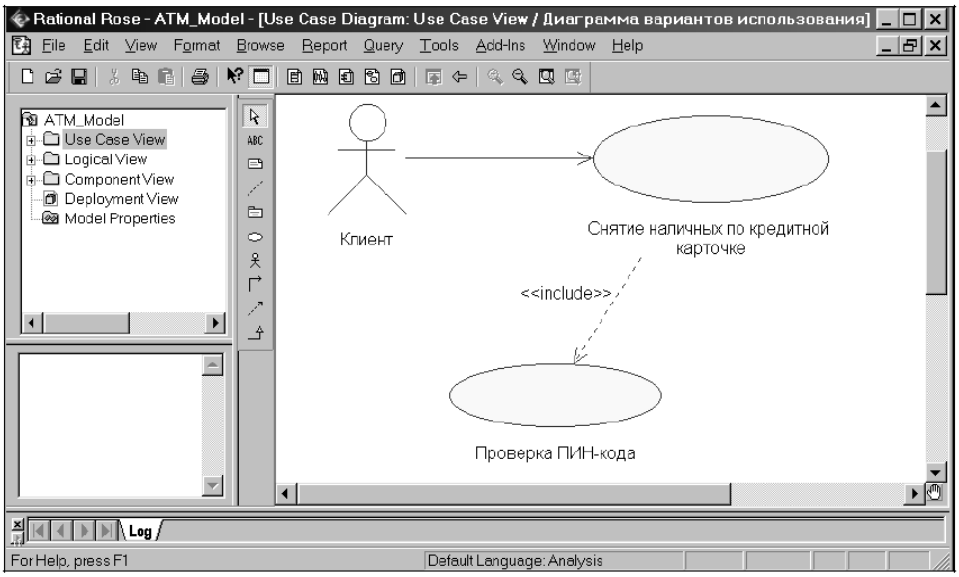



*Рис. 10.7.* Діаграма варіантів використання після додавання на неї відношення залежності

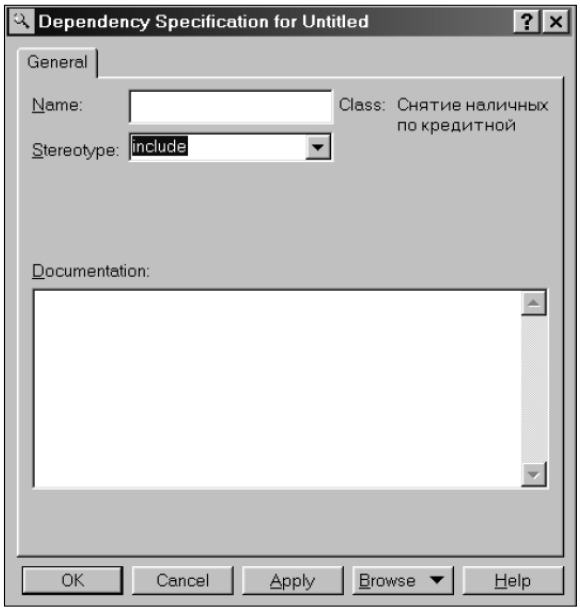



*Рис. 10.8.* Діаграма варіантів використання після додавання до неї відношення залежності

Аналогічним чином можна додати до діаграми варіантів використання відношення залежності зі стереотипом «extend», які застосовують для моделювання винятків варіанту використання.

**Додавання текстового файлу зі сценарієм варіанту використання.** Для додавання до варіанту використання текстового файлу (наприклад, до варіанту використання «Зняття готівки з кредитної картки» - файлу формату MS Word) з описом сценарію його виконання необхідно виділити цей варіант використання в браузері проєкту, за допомогою правої кнопки миші викликати контекстне меню і виконати операцію **New → File** (Новий → Файл). В результаті цього буде викликано стандартне вікно відкриття файлу, в якому необхідно задати ім'я попередньо створеного в середовищі MS Word файлу, що додається.

Після натискання кнопки «Відкрити» піктограма доданого файлу з'явиться в браузері проєкту нижче відповідного варіанту використання. В подальшому, можна повернутися до редагування цього файлу сценарію, виконавши подвійне клацання на цій піктограмі. При цьому файл сценарію буде відкритий у відповідній програмі (в даному випадку – в текстовому процесорі MS Word).

Для остаточної побудови діаграми варіанту використання розглянутого прикладу слід аналогічним чином додати актора з ім'ям «Банк», варіант використання «Отримання довідки про стан рахунку», а також відповідні асоціації та відношення залежності. Побудована таким чином діаграма варіантів використання матиме вигляд, зобрбажений на рис. 10.9.

Діаграма варіантів використання є високорівневим поданням моделі, тому вона не повинна містити занадто багато варіантів використання та акторів. В подальшому, побудована діаграма може бути змінена додаванням нових елементів, таких як варіанти використання та актори, або їх видаленням.

Для видалення графічного елементу з діаграми його слід виділити на діаграмі і натиснути клавішу «Delete» на клавіатурі. При цьому виділений елемент буде видалено з активної діаграми, але не з моделі. Для видалення елемента не тільки з діаграми, а й з моделі проєкту необхідно виділити елемент для видалення на діаграмі і скористатися пунктом меню Edit → Delete from Model (Редагування → Видалити з моделі). Для цієї ж мети служить комбінація клавіш швидкого доступу (Ctrl + D).

*Примітка.* При роботі зі зв'язками на діаграмі варіантів використання слід пам'ятати про призначення відповідних зв'язків в нотації мови UML. Йдеться про те, що якщо для двох елементів обраний вид зв'язку не є допустимим, то в більшості випадків середовище Rational Rose повідомить про це розробнику, і такий зв'язок не буде додано до діаграми.

Розробку моделі системи управління банкоматом продовжимо побудовою наступного типу канонічних діаграм – діаграми класів.

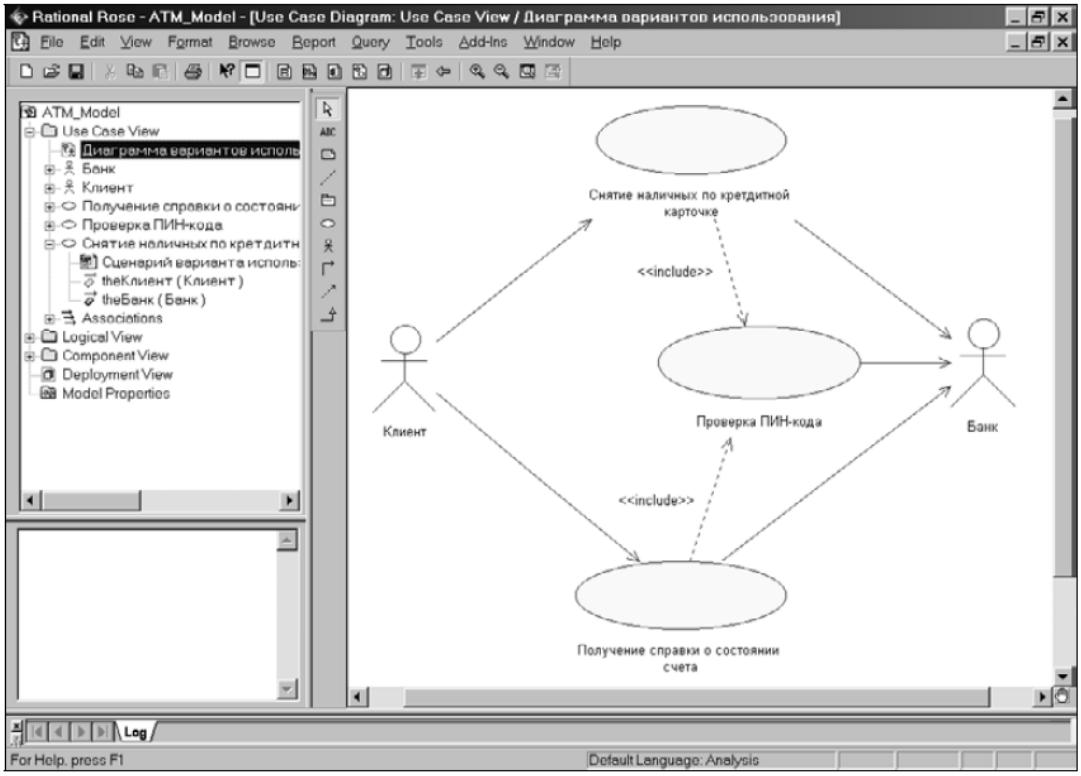



*Рис. 10.9***.** Остаточний вигляд діаграми варіантів використання розроблюваної моделі управління банкоматом

## Теоретичні відомості

### 1. Загальна характеристика інструментарія Rational Rose

Поява на ринку програмних продуктів перших CASE-засобів (Computer Aided Software Engineering) ознаменувало новий етап розвитку програмної інженерії, характерними особливостями якого є скорочення термінів реалізації програмних проєктів, підтримка методології групової розробки та орієнтація на візуальні засоби специфікації компонентів програмного забезпечення.

Первісною сферою застосування CASE-засобів були додатки баз даних, особливо ті з них, які вимагали серйозних зусиль при розробці своїх концептуальних схем. Реалізація можливості автоматичної генерації програмного коду на основі попередньо розробленої концептуальної схеми виявилася настільки конструктивної, що стимулювала появу десятків CASE-засобів різних фірм.

Початковий етап розвитку CASE-технологій характеризувався тим, що різні фірми пропонували свої власні засоби візуального подання концептуальних схем. Найчастіше вибір розробниками того чи іншого CASE-засобу визначався простотою нотації підтримуваного засобом мови подання схем і діаграм.

Поява стандартів в цій галузі стабілізувала ситуацію. Однак найгостріша конкуренція серед фірм-виробників програмного забезпечення вимагала від CASE-засобів реалізації об'єктно-орієнтованої технології розробки програм, підтримки широкого діапазону мов програмування і конкретних баз даних.

Слід згадати середовище MS Visual Studio.NET, яке підтримує розробку діаграм мови UML і подальшу генерацію програмного коду в нотації відповідних мов програмування, включаючи мову C#.

За універсальністю платформ реалізації, повнотою мов програмування і схем баз даних, продовжує лідирувати засіб Rational Rose. Саме тому даний засіб обрано в якості базового для ілюстрації можливостей інструментальної підтримки мови UML і процесу розробки візуальних моделей у відповідній нотації.

Серед фірм-виробників CASE-засобів саме компанія IBM Rational Software Corp. (до серпня 2003 року - Rational Software Corp.) одна з перших усвідомила перспективність розвитку об'єктно-орієнтованих технологій аналізу та проєктування програмних систем. Ця компанія виступила ініціатором уніфікації мови візуального моделювання в рамках консорціуму OMG, що призвело до появи перших версій мови UML, і першою розробила інструментальний об'єктно-орієнтований CASE-засіб, в якому було реалізовано мову UML як базову нотацію візуального моделювання.

Серед причин, що стримують застосування CASE-засобів і визначають контраст їх популярності серед західних і вітчизняних розробників програм, слід зазначити, в першу чергу, масштабність проєктів і відмінність в технологіях створення програм (необхідність автоматизації аналізу та проєктування програмних систем на базі CASE-технології починає усвідомлюватися, коли виконуваний проєкт є досить складним і масштабним). Реалізація масштабних проєктів під силу лише групі розробників високої кваліфікації, в яку, крім програмістів, повинні входити системні аналітики, бізнес-аналітики, системотехніки, тестувальники та інші категорії фахівців. При цьому неодмінною умовою успішної роботи є відповідна кваліфікація фахівців, в тому числі і знання ними базової нотації мови UML, а забезпечення групової роботи над проєктом вимагає додаткових коштів для реалізації сумісності його складових частин.

### 1.1. Загальна характеристика CASE-засобу Rational Rose

CASE-засіб IBM Rational Rose – це сучасний інтегрований інструментарій аналізу, моделювання і розробки програмних систем (де мова UML стала базовою технологією візуалізації і розробки програмних систем). В рамках IBM Rational Rose існують різні варіанти цього засобу, що відрізняються між собою діапазоном реалізованих можливостей. Базовим засобом в даний час є IBM Rational Rose Enterprise Edition, який володіє найбільш повними можливостями.

CASE-засіб Rational Rose акумулює такі можливості:

1) інтеграцію з MS Visual Studio 6 / .NET, яка включає в себе підтримку на рівні прямої і зворотньої генерації кодів і діаграм VB 6, Visual C++ 6, Visual J ++ 6 (ATL - Microsoft Active Template Library, Web- Classes, DHTML, Data Connections);

2) безпосередню роботу (інжиніринг і реінжиніринг) з виконуваними модулями і бібліотеками;

3) підтримку технологій MTS (Microsoft Transaction Server) і ADO (ActiveX Data Objects) на рівні шаблонів і вихідного коду, а також елементів технології COM + (DCOM);

4) повну підтримку CORBA, включаючи реалізацію технології компонентної розробки додатків CBD (Component-Based Development), мови визначення інтерфейсу IDL (Interface Definition Language) і мови визначення даних DDL (Data Definition Language);

5) повну підтримку середовища розробки Java-додатків, включаючи пряму і зворотню генерацію класів Java формату JAR, а також роботу з файлами формату CAB і ZIP.

### 1.2. Особливості інтерфейсу Rational Rose

У CASE-засобі IBM Rational Rose реалізовані загальноприйняті стандарти для робочого інтерфейсу програми, подібно відомим середовищам візуального програмування. Після установки IBM Rational Rose на комп'ютер користувача запуск цього засобу в операційних системах сімейств Microsoft (MS) Windows 9x / NT призводить до появи на екрані відповідного робочого інтерфейсу програми (рис. 10.10).

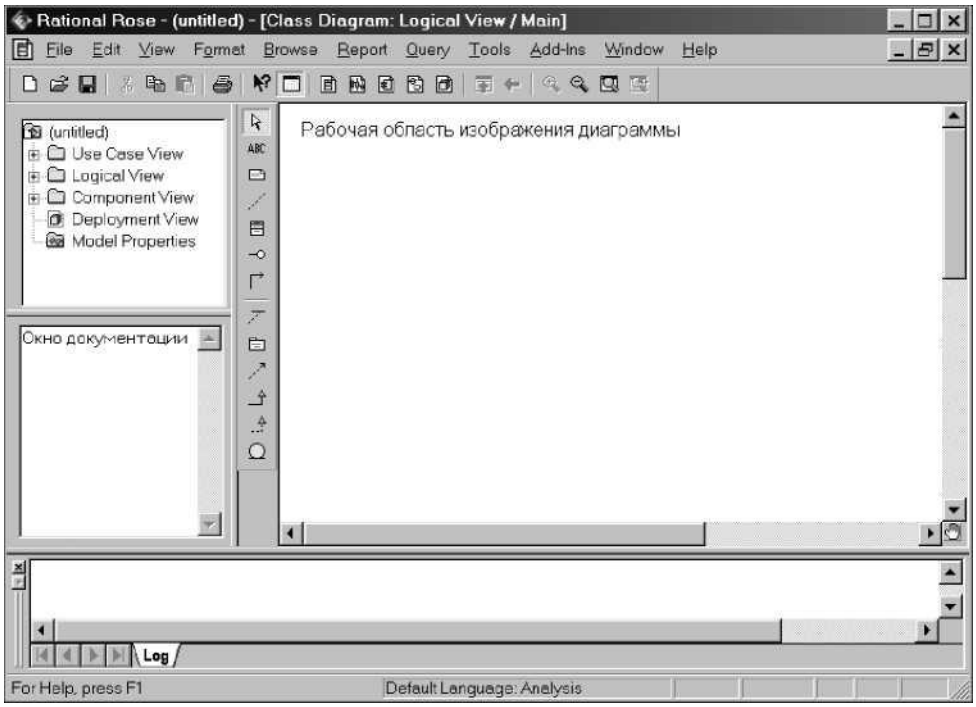



*Рис.* 10.10. Загальний вигляд робочого інтерфейсу середовища Rational Rose

Робочий інтерфейс складається з різних елементів, основними з яких є: головне меню; стандартна панель інструментів; вікно браузера; спеціальна панель інструментів; робоча область зображення діаграми (вікно діаграми); вікно документації; вікно журналу. Розглянемо коротко призначення і основні функції кожного з цих елементів.

**Головне меню.** Головне меню засобу Rational Rose виконано в загальноприйнятому стандарті і має вигляд, зображений на рис. 10.11.

File Edit View Farm at growse Report Query Joels Add-Ins Window Help

*Рис. 10.11.* Зовнішній вигляд головного меню програми

Окремі пункти меню об'єднують подібні операції, що відносяться до усього проєкту в цілому. Деякі з пунктів меню містять добре знайомі функції (відкриття проєкту, друк діаграм, копіювання в буфер і вставка з буфера різних елементів діаграм). Інші настільки специфічні, що можуть потребувати додаткових зусиль на вивчення (опції генерації програмного коду, перевірка узгодженості моделей, підключення додаткових модулів).

**Стандартна панель інструментів.** Стандартна панель інструментів розташовується нижче рядка головного меню і має вигляд, зображений на рис. 10.12. Деякі з інструментів недоступні для нового проєкту, який поки не має ніяких елементів. Стандартна панель інструментів забезпечує швидкий доступ до тих команд меню, які виконуються розробниками найбільш часто.

*Рис. 10.12.* Зовнішній вигляд стандартної панелі інструментів

Користувач може налаштувати зовнішній вигляд цієї панелі на свій розсуд. Для цього необхідно виконати операцію головного меню Tools → Options (Інструменти → Параметри), відкрити вкладку Toolbars (Панелі інструментів) діалогового вікна і натиснути кнопку Standard (Стандартна). У додатково відкритому вікні можна переносити необхідні кнопки з лівого списку в правий, а непотрібні кнопки - з правого списку в лівий. Цим способом можна показати або приховати різні кнопки інструментів, а також змінити їх розмір. Призначення окремих кнопок стандартної панелі інструментів наведені далі.

**Вікно браузера проєкту.** Вікно браузера проєкту за замовчуванням розташовується в лівій частині робочого інтерфейсу під стандартною панеллю інструментів і має вигляд, зображений на рис. 10.13.

Браузер проєкту організує зображення моделі у вигляді ієрархічної структури, яка спрощує навігацію і дозволяє знайти будь-який елемент моделі в проєкті. При цьому будь-який елемент, який розробник додає в модель, відразу відображається у вікні браузера. Відповідно, вибравши елемент у вікні браузера, ми можемо його візуалізувати у вікні діаграми або змінити його специфікацію. Браузер проєкту дозволяє також організовувати елементи моделі в пакети й переміщати елементи між різними зображеннями моделі. При бажанні вікно браузера можна розташувати в іншому місці робочого інтерфейсу або приховати, використовуючи для цього пункт меню View (Вигляд) (можна також змінити розміри браузера, перемістивши мишкою границю його зовнішньої рамки).

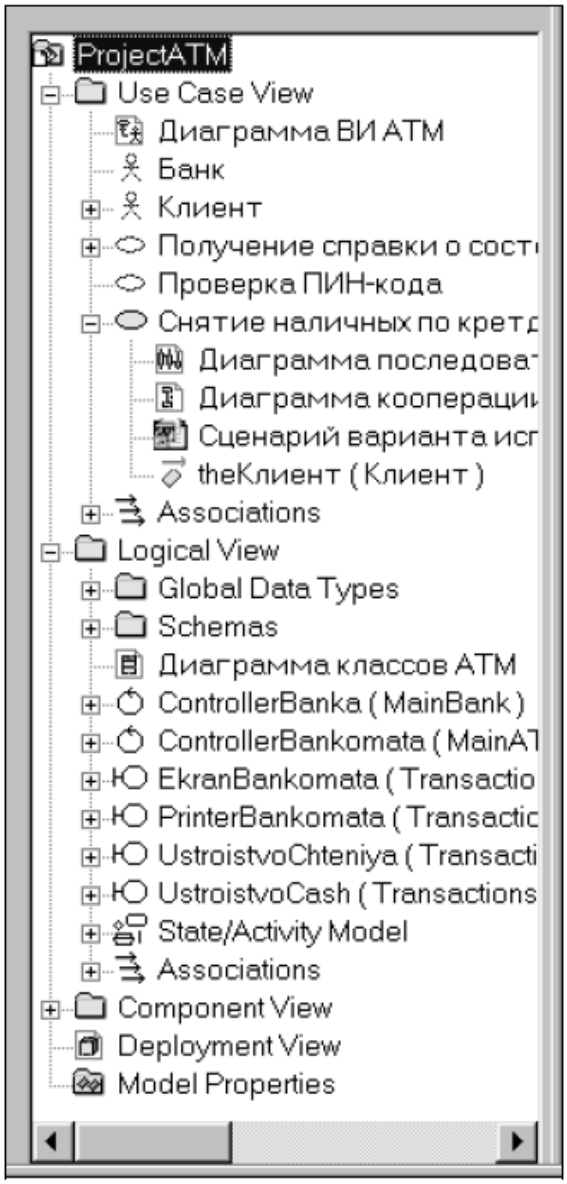



*Рис. 10.13.* Зовнішній вигляд браузера з ієрархічним поданням структури проєкту

**Спеціальна панель інструментів.** Спеціальна панель інструментів розташовується між вікном браузера і вікном діаграми в середній частині робочого інтерфейсу. За замовчуванням пропонується панель інструментів для побудови діаграми класів моделі (рис. 10.14).


*Рис. 10.14.* Зовнішній вигляд спеціальної панелі інструментів для діаграми класів

Розташування спеціальної панелі інструментів можна змінювати, перемістивши рамку панелі в потрібне місце. Можна налаштовувати і зміст панелі, додаючи або видаляючи окремі кнопки, що відповідають тим чи іншим

інструментам. Призначення кнопок можна дізнатися з спливаючих підказок, які з'являються після затримки покажчика миші над відповідною кнопкою.

Зовнішній вигляд спеціальної панелі інструментів визначається не тільки вибором виду діаграми, а й вибором графічної нотації для зображення самих елементів цих діаграм. В Rational Rose реалізовані три таких нотації: UML, OMT і Booch: одна і та ж діаграма може бути подана по-різному, для цього достатньо вибрати бажане уявлення через пункт меню View (Вигляд) (ми залишимо без уваги особливості двох останніх нотацій, які відображають еволюційний аспект засобу Rational Rose).

**Вікно діаграми.** Вікно діаграми є основною робочою областю її інтерфейсу, в якій візуалізуються різні уявлення моделі проєкту. За замовчуванням вікно діаграми розташовується в правій частині робочого інтерфейсу, проте його розташування і розміри також можна змінити. При розробці нового проєкту, якщо майстер проєктів не був використаний, вікно діаграми являє собою чисту область, яка не містить жодних елементів моделі (рис. 10.15). Назва діаграми, яка розташовується в цьому вікні, вказується у рядку заголовка програми або, якщо вікно не розгорнуто на весь екран, у рядку заголовка вікна діаграми. Одночасно у вікні діаграми можуть бути присутні кілька діаграм, проте активною може бути тільки одна з них (наприклад, на рис. 10.15 активною є діаграма послідовності, хоча є й інші діаграми).

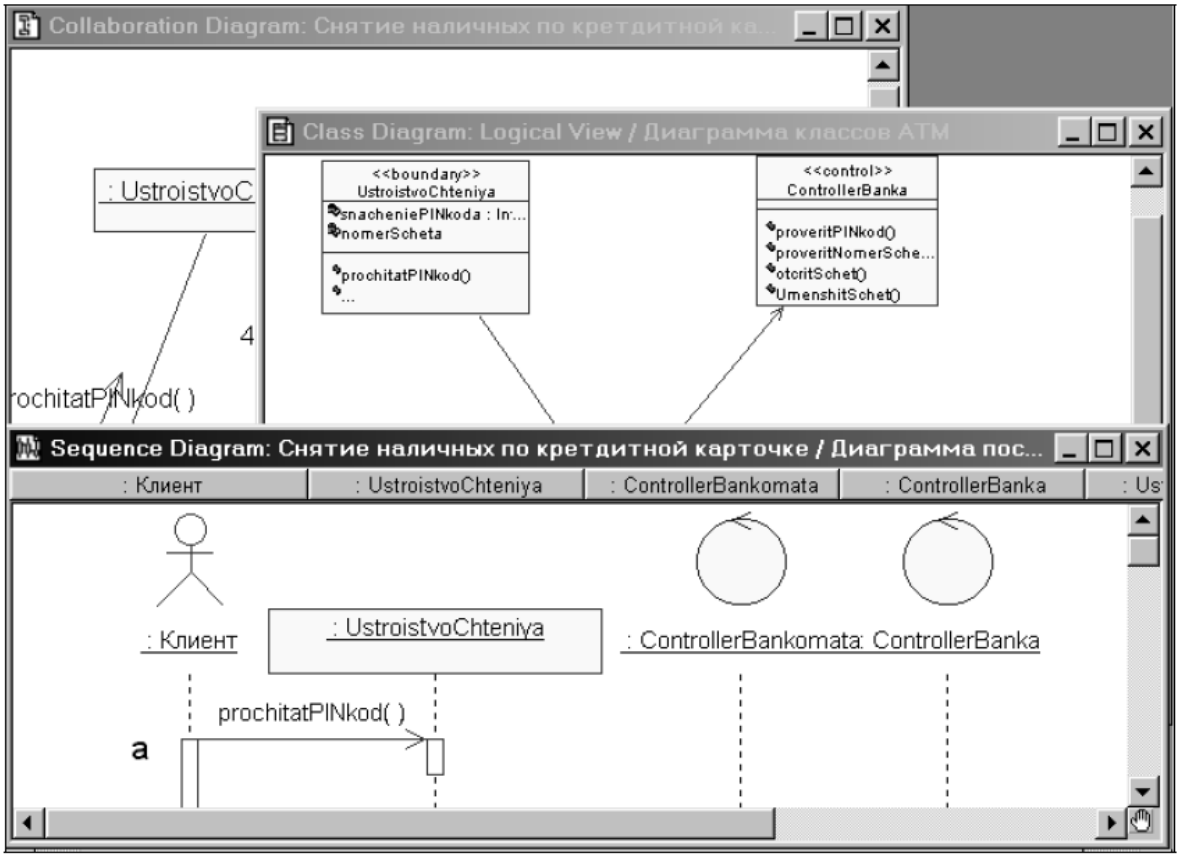



*Рис. 10.15.* Зовнішній вигляд вікна діаграм з різними видами зображення моделі

Перемикання між діаграмами можна здійснити вибором потрібного подання на стандартній панелі інструментів або через пункт меню Window (Вікно). При активізації окремого виду діаграми змінюється зовнішній вигляд спеціальної панелі інструментів, яка налаштовується під конкретний різновид діаграми.

**Вікно документації.** Вікно документації, за замовчуванням, присутнє на екрані після завантаження програми. Якщо з якоїсь причини воно відсутнє, то його можна активізувати через пункт меню View → Documentation (Вид → Документація), після чого вікно документації з'явиться нижче вікна браузера (рис. 10.16).

Вікно документації призначене для документування елементів подання моделі. У нього можна записувати різну інформацію, яка в подальшому перетворюється в коментарі і ніяк не впливає на логіку виконання програмного коду.

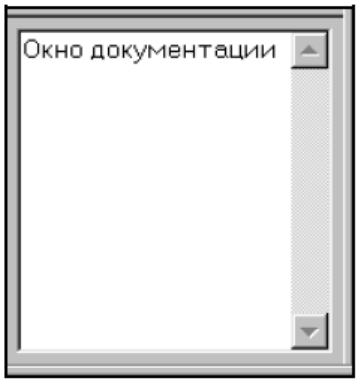



*Рис. 10.16.* Зовнішній вигляд вікна документації

У вікні документації активується інформація, яка відноситься до окремо виділеного елементу діаграми (при цьому виділити елемент можна або у вікні браузера, або у вікні діаграми). При додаванні нового елемента до діаграми (наприклад, класу) документація до нього буде порожньою (No documentation). В подальшому розробник самостійно вносить необхідну пояснювальну інформацію, яка запам'ятовується і може бути змінена під час роботи над проєктом. Розробник на свій розсуд може змінювати розміри і положення вікна документації.

**Вікно журналу.** Вікно журналу (Log) призначене для автоматичного запису різної службової інформації, яка створюється в ході роботи з програмою. У журналі фіксується час і характер виконуваних розробником дій, таких як оновлення моделі, налаштування меню і панелей інструментів, а також повідомлень про помилки, що виникають при генерації програмного коду.

Вікно журналу завжди присутнє на робочому інтерфейсі в області вікна діаграми (рис. 10.17). Однак воно може бути закрито іншими вікнами з діаграмами або бути згорнутим.

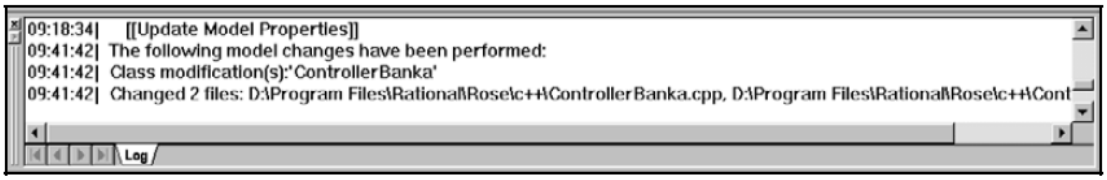



*Рис*. 10.17. Зовнішній вигляд вікна журналу

Активізувати вікно журналу можна через меню Window → Log (Вікно → Журнал). У цьому випадку вікно зображується поверх інших, у правій області робочого інтерфейсу (повністю видалити це вікно неможливо, його можна тільки мінімізувати).

**Призначення операцій головного меню.** Головне меню дозволяє користувачеві викликати інші графічні засоби роботи з системою Rational Rose: завантажувати і зберігати інформацію в файлах, змінювати зовнішній вигляд елементів графічного інтерфейсу, викликати довідкову інформацію тощо. Розглянемо призначення окремих пунктів головного меню.

**1. Пункт меню File** (Файл) головного меню містить такі операції (рис. 10.18).

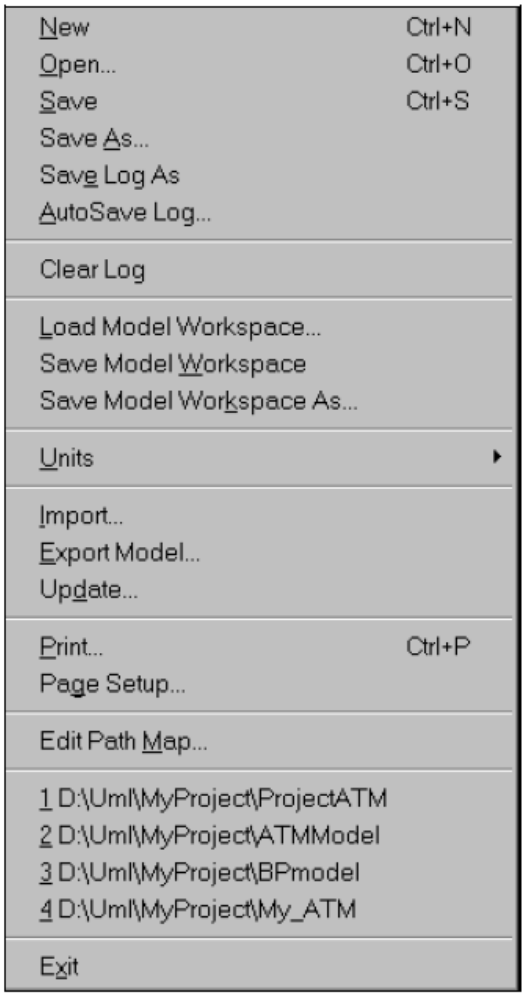



*Рис. 10.18***.** Операції пункту меню File (Файл) головного меню

New створює нову модель Rational Rose, яка за замовчуванням має ім'я untitled.

Open викликає стандартне діалогове вікно відкриття зовнішнього файлу з диска. Відкрити можна або файл моделі (файл з розширенням mdl), або файл підмоделі (файл з розширенням ptl).

Save дозволяє зберегти розробляючу модель у файлі на диску.

Save As дозволяє зберегти розробляючу модель під іншим ім'ям у файлі на диску. При цьому викликається стандартне діалогове вікно збереження файлу на диску з пропозицією ввести ім'я відповідного файлу моделі або підмоделі.

Save Log As дозволяє зберегти зміст журналу до файла на диску з ім'ям error.log. При цьому викликається стандартне діалогове вікно збереження файлу на диску з пропозицією змінити пропоноване за замовчуванням ім'я відповідного файлу.

AutoSave Log дозволяє автоматично зберігати зміст журналу до файла на диску з ім'ям error.log. При першому виконанні цього пункту меню також викликається стандартне діалогове вікно збереження файлу на диску з пропозицією змінити використовуваний за замовчуванням шлях та ім'я файлу.

Clear Log очищає журнал.

Load Model Workspace дозволяє завантажити робочу область з файлу на диску. Викликає стандартне діалогове вікно відкриття файлу з розширенням wsp.

Save Model Workspace дозволяє зберегти робочу область моделі у файлі на диску. Викликає стандартне діалогове вікно збереження файлу з розширенням wsp.

Save Model Workspace As дозволяє зберегти робочу область моделі у файлі на диску. Викликає стандартне діалогове вікно збереження файлу з пропозицією змінити пропоноване за замовчуванням ім'я відповідного файлу.

Units містить вкладені підпункти меню для забезпечення контролю версій моделі, що розробляється.

Import дозволяє імпортувати інформацію з файлів різних форматів, включаючи файли моделей, підмоделей, категорій і підсистем.

Export Model дозволяє експортувати інформацію про модель в файл. Вид цього пункту меню залежить від виділеного елемента моделі.

Update дозволяє вставити інформацію зворотного проєктування з зовнішнього файлу з розширенням red в розроблювану модель.

Print дозволяє роздрукувати на принтері окремі діаграми і специфікації різних елементів моделі, що розробляється. В цьому випадку викликається діалогове вікно вибору діаграм і специфікацій для друку на підключеному до комп'ютера принтера.

Print Setup — викликається стандартне діалогове вікно макета сторінки для налаштування властивостей друку.

Edit Path Map викликає вікно завдання шляхів доступу до файлів системи Rational Rose. Зазвичай значення шляхів, встановлені за замовчуванням, слід змінювати тільки в разі крайньої необхідності.

Секція з іменами останніх файлів, з якими здійснювалася робота в Rational Rose.

Exit припиняє роботу і закриває систему Rational Rose.

**2. Пункт меню Edit** (Редагування) містить такі операції (рис. 10.19).

Undo скасовує виконання останньої дії з видалення або переміщення елементів моделі.

Redo повторює дію після її відміни.

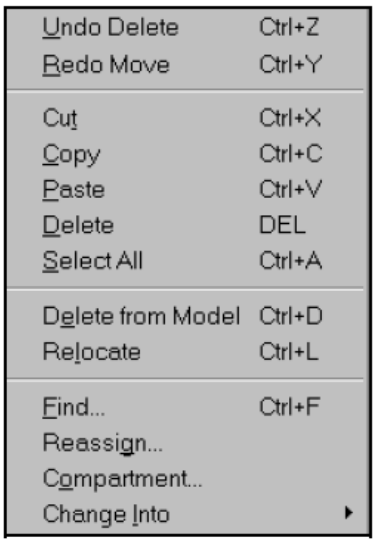



*Рис. 10.19.* Операції пункту меню Edit (Редагування) головного меню

Cut вирізає виділений елемент моделі, що розробляється, і поміщає його в буфер обміну.

Copy копіює виділений елемент моделі, що розробляється, і поміщає його в буфер обміну.

Paste вставляє елемент моделі, що розробляється, або його копію з буфера обміну в поточну активну діаграму.

Delete видаляє виділені елементи з поточної діаграми, але не з розроблюваної моделі.

Select AI виділяє всі елементи на поточній діаграмі розробляючої моделі.

Delete from Model видаляє всі виділені елементи з розроблюваної моделі.

Relocate дозволяє переміщати або скасувати переміщення класів, асоціацій або компонентів з одного пакету в інший.

Find викликає діалогове меню пошуку елемента в розроблюючій моделі по його імені.

Reassign дозволяє замінити виділений елемент моделі, що розробляється, іншим елементом моделі.

Compartment дозволяє відображати додаткову інформацію про об'єкти, класи, акторів або пакети.

Change Info дозволяє змінити тип виділеного елемента на поточній діаграмі на інший тип елемента.

**3. Пункт меню View** (Вигляд) дозволяє відображати на екрані різні елементи робочого інтерфейсу і змінювати графічне подання діаграм, містить такі операції (рис. 10.20):

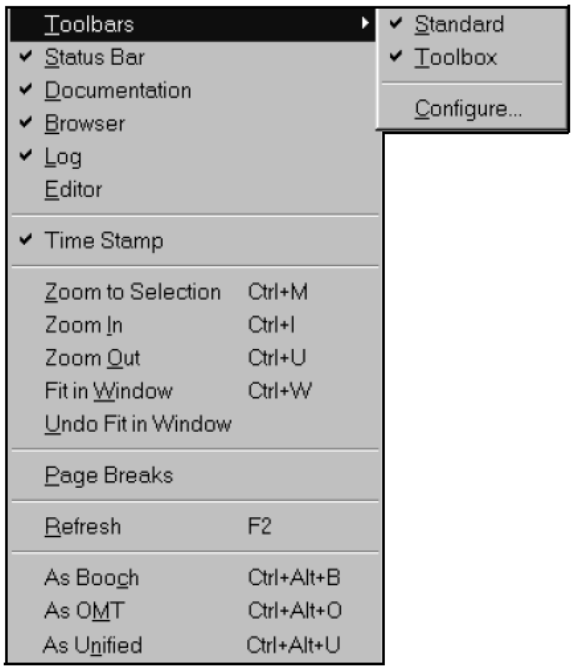



*Рис. 10.20*. Операції пункту меню View (Вигляд) головного меню

Toolbars дозволяє налаштувати зовнішній вигляд робочого інтерфейсу системи і містить додаткові підпункти: Standard робить видимою / невидимою стандартну панель інструментів (рис. 10.12); Toolbox робить видимою / невидимою спеціальну панель інструментів поточної активної діаграми (рис. 10.13); Configure викликає діалогове вікно налаштування параметрів моделі, відкрите на вкладці налаштування панелей інструментів.

Status bar робить видимим / невидимим рядок стану.

Documentation робить видимим / невидимим вікно документації.

Browser робить видимим / невидимим браузер проєкту.

Log робить видимим / невидимим вікно журналу.

Editor робить видимим / невидимим вбудований текстовий редактор.

Time Stamp включає / вимикає режим відображення часу в записах журналу.

Zoom to Selection змінює масштаб зображення виділених елементів моделі, так щоб вони розмістилися в одному вікні.

Zoom In збільшує масштаб зображення.

Zoom Out зменшує масштаб зображення.

Fit in Window змінює (зменшує) масштаб зображення всіх елементів поточної діаграми, так щоб всі вони розмістилися в одному вікні.

Undo Fit in Window скасовує зміну масштабу зображення для розміщення елементів в одному вікні.

Page Breaks розбиває поточну діаграму на сторінки для подальшого друку.

Refresh перемальовує поточну діаграму.

As Booch зображує елементи моделі відповідно до нотації Буча.

As OMT зображує елементи моделі відповідно до нотації OMT.

As Unified зображує елементи моделі відповідно до нотації мови UML.

**4. Пункт меню Format** (Формат) містить такі операції (рис. 10.21).

Font Size змінює розмір використовуваного шрифту.

Font викликає діалогове вікно вибору шрифту.

Line Color викликає діалогове вікно вибору кольору ліній.

Fill Color викликає діалогове вікно для вибору кольору заливки елементів діаграм.

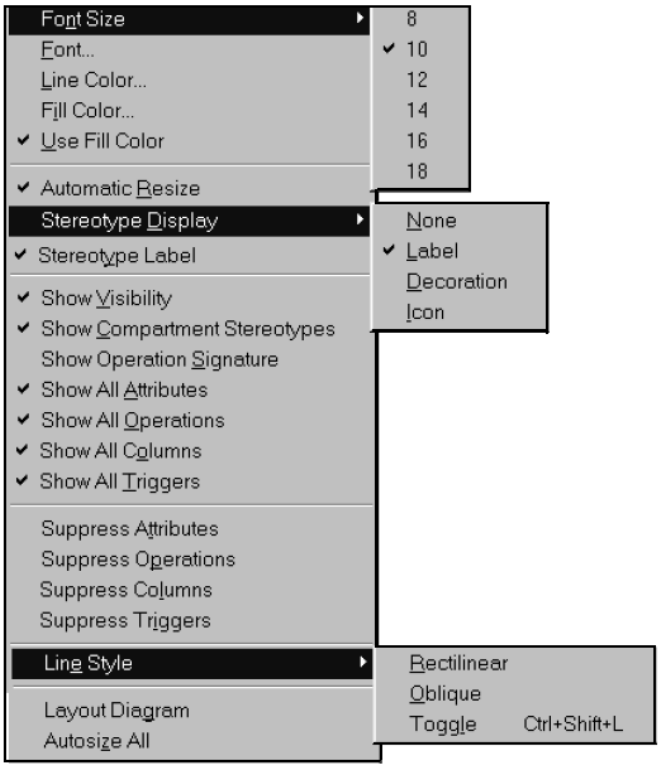



*Рис. 10.21.* Операції пункту меню Format (Формат) головного меню

Use Fill Color включає / вимикає режим відображення кольору заливки елементів діаграм.

Automatic Resize включає / вимикає режим автоматичної зміни розмірів графічних елементів діаграм для відображення текстової інформації про їх властивості.

Stereotype Display дозволяє вибрати спосіб зображення стереотипів виділених елементів діаграми і містить додаткові підпункти: None — стереотип не відображається; Label — стереотип відображається в формі тексту; Decoration — стереотип відображається у формі невеликої піктограми в

правому верхньому куті графічного елементу; Icon — графічний елемент діаграми відображається у формі спеціального стереотипу.

Stereotype Label включає / вимикає режим відображення текстових стереотипів для взаємозв'язків (асоціацій, залежностей тощо) діаграми.

Show Visibility включає / вимикає режим відображення кванторів видимості атрибутів та операцій класів.

Show Compartment Stereotypes включає / вимикає режим відображення текстових стереотипів атрибутів та операцій класів.

Show Operation Signature включає / вимикає режим відображення сигнатури операцій класів.

Show All Attributes включає / вимикає режим відображення текстових стереотипів атрибутів та операцій класів.

Show All Operations робить видимими / невидимими операції класів.

Show All Columns робить видимими / невидимими стовпці (поля) таблиці моделі даних.

Show All Triggers робить видимими / невидимими тригери таблиці моделі даних.

Suppress Attributes робить видимою / невидимою секцію атрибутів класів.

Suppress Operations робить видимою / невидимою секцію операцій класів.

Suppress Columns робить видимою / невидимою секцію стовпців таблиці моделі даних.

Suppress Triggers робить видимою / невидимою секцію тригерів таблиці моделі даних.

Line Style дозволяє вибрати спосіб графічного зображення ліній взаємозв'язків і містить такі додаткові підпункти: Rectilinear — лінія зображується у формі вертикальних і горизонтальних відрізків; Oblique — лінія зображується у формі похилих відрізків; Toggle — проміжний варіант зображення лінії.

Layout Diagram дозволяє автоматично розмістити графічні елементи у вікні діаграми з мінімальною кількістю перетинів і накладень з'єднувальних ліній.

Autosize All дозволяє автоматично змінити розміри графічних елементів поточної діаграми таким чином, щоб текстова інформація містилася всередині зображень відповідних елементів.

**5. Пункт меню Browse** (Огляд) містить такі операції (рис. 10.22):

Use Case Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм варіантів використання моделі або почати розробку нової діаграми.

Class Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм класів моделі або приступити до розробки нової діаграми.

Component Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм компонентів моделі або приступити до розробки нової діаграми.

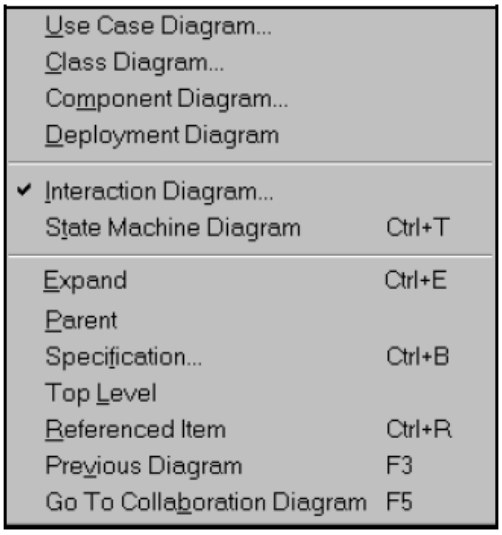



*Рис.* 10.22. Операції пункту меню Browse (Обзор) головного меню

Deployment Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм розгортання моделі або приступити до розробки нової діаграми.

Interaction Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм кооперації (послідовності) моделі або приступити до розробки нової діаграми.

State Machine Diagram викликає діалогове вікно з пропозицією вибрати для відображення в робочому вікні одну з існуючих діаграм станів моделі або приступити до розробки нової діаграми.

Expand відображає в робочому вікні першу з діаграм виділеного пакету моделі.

Parent відображає в робочому вікні батька виділеної діаграми моделі.

Specification викликає діалогове вікно властивостей виділеного елемента моделі.

Top Level відображає в робочому вікні діаграму самого верхнього рівня для поточної діаграми моделі.

Referenced Item відображає в робочому вікні діаграму класів, яка містить клас для виділеного об'єкта моделі.

Previous Diagram відображає в робочому вікні попередню діаграму моделі.

Go To Collaboration Diagram (Go To Sequence Diagram) дозволяє перейти від подання діаграми послідовності до подання діаграми кооперації і навпаки.

**6. Пункт меню Report** (Звіт) містить такі операції (рис. 10.23):

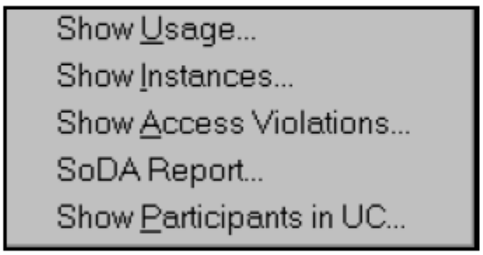



*Рис. 10.23.* Операції пункту меню Report (Звіт) головного меню

Show Usage відображає в діалоговому вікні інформацію про використання виділеного елемента моделі на різних діаграмах.

Show Instances відображає в діалоговому вікні інформацію про використання об'єктів виділеного класу моделі на різних діаграмах.

Show Access Violations відображає в діалоговому вікні інформацію про посилання класів одного пакета на класи іншого, при відсутності відповідної залежності доступу (імпорту) між цими пакетами моделі.

SoDA Report дозволяє згенерувати звіт про розроблювану модель в форматі MS Word з використанням спеціального засобу IBM Rational SoDA.

Show Participants in UC відображає в діалоговому вікні інформацію про класи, компоненти та операції, які беруть участь в реалізації виділеного варіанту використання моделі на різних діаграмах.

**7. Пункт меню Query** (Запит) містить такі операції (рис. 10.24):

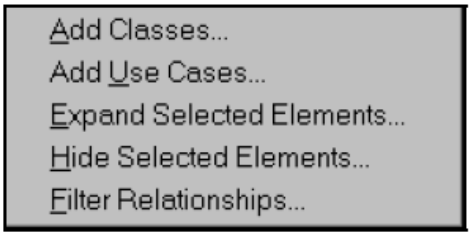



*Рис.* 10.24. Операції пункту меню Query (Запит) головного меню

Add Classes (Add Use Cases) викликає діалогове вікно з пропозицією додати до поточної діаграми класи (варіанти використання), які є в моделі на різних діаграмах.

Expand Selected Elements викликає діалогове вікно з пропозицією додати до поточної діаграми елементи моделі, які пов'язані з виділеним елементом на інших діаграмах.

Hide Selected Elements викликає діалогове вікно з пропозицією видалити з поточної діаграми елементи моделі, які пов'язані з виділеним елементом.

Filter Relationships викликає діалогове вікно, що дозволяє ввімкнути / вимкнути режим відображення різних відношень на поточній діаграмі.

**8. Пункт меню Tools** (Інструменти) залежить від встановлених на конкретний засіб Rational Rose розширень і може містити, наприклад, такі операції (рис. 10.25).

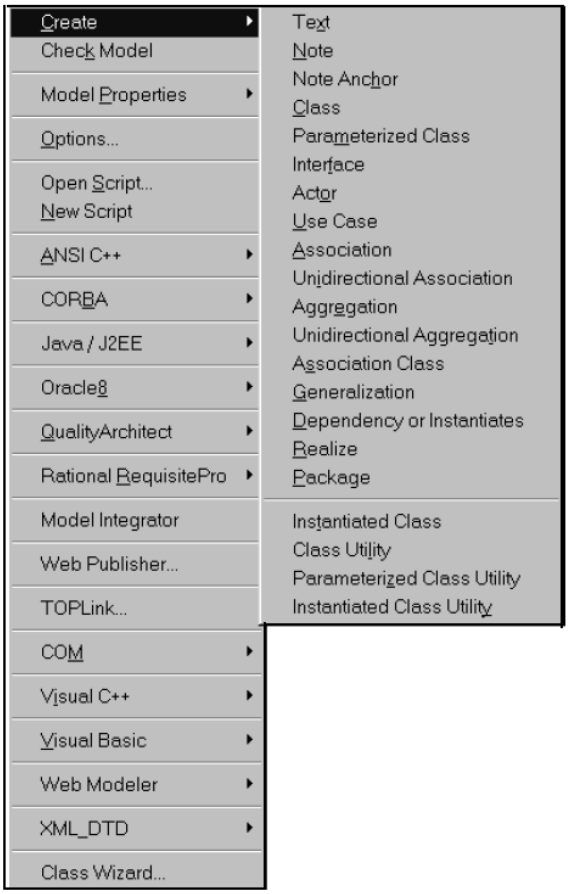



*Рис.* 10.25. Операції пункту меню Tools (Інструменти) головного меню

Create створює новий елемент моделі із запропонованого списку для подальшого розміщення його на діаграмі, дублює натиснення відповідної кнопки на спеціальній панелі інструментів.

Check Model перевіряє модель, яка розробляється, на наявність помилок, інформація про які відображається у вікні журналу.

Model Properties дозволяє виконати налаштування властивостей мови реалізації виділеного елемента моделі і містить додаткові підпункти: Edit - редагування набору властивостей; View - перегляд набору властивостей; Replace - заміна існуючого набору властивостей на новий набір, що завантажується з зовнішнього файлу з розширенням pip або pty; Export - збереження існуючого набору в файл з розширенням pip або pty; Add - доповнення існуючого набору властивостей новим набором властивостей, що завантажується з зовнішнього файлу з розширенням pip або pty; Update - оновлює існуючий набір властивостей після його редагування або доповнення.

Options викликає діалогове вікно налаштування параметрів моделі, відкрите на вкладці General.

Open Script викликає стандартне діалогове вікно відкриття зовнішнього файлу, який містить текст скрипту (файл з розширенням ebs) для його редагування у вікні спеціального редактора скриптів.

New Script відкриває додаткове вікно спеціального редактора скриптів для створення, налагодження, виконання і збереження нового скрипту в зовнішньому файлі з розширенням ebs.

ANSI C++ дозволяє виконати налаштування властивостей мови програмування C ++ стандарту ANSI, обраного в якості мови реалізації окремих елементів моделі.

CORBA дозволяє виконати налаштування властивостей і специфікацію моделі для генерації об'єктів CORBA для реалізації окремих елементів моделі.

Java/J2EE дозволяє виконати налаштування властивостей мови програмування Java, обраної в якості мови реалізації окремих елементів моделі.

Orac1e8 дозволяє виконати налаштування властивостей і специфікацію моделі для генерації схем СУБД Oracle 8 для окремих елементів моделі.

QualityArchitect дозволяє виконати налаштування властивостей і тестування моделі за допомогою спеціального засобу IBM Rational QualityArchitect.

Rational Requisite Pro дозволяє виконати налаштування властивостей моделі для встановлення зв'язку зі спеціальним засобом специфікації та управління вимогами.

Model Integrator відкриває робоче вікно спеціального засобу для інтеграції моделі.

Web Publisher дозволяє виконати налаштування властивостей моделі для її публікації в гіпертекстовому форматі.

TOPLink викликає майстер перетворення таблиць моделі даних в класи мови програмування Java, обраної в якості мови реалізації окремих елементів моделі.

COM дозволяє виконати налаштування властивостей і специфікацію моделі для генерації об'єктів COM з метою реалізації окремих елементів моделі.

Visual C++ дозволяє виконати налаштування властивостей і специфікацію моделі для генерації програмного коду MS Visual C++, обраного в якості мови реалізації окремих елементів моделі.

Visual Basic дозволяє виконати налаштування властивостей і специфікацію моделі для генерації програмного коду MS Visual Basic, обраної в якості мови реалізації окремих елементів моделі.

Web Modeler викликає майстер перетворення існуючого Web-вузла в модель IBM Rational Rose.

XML_DTD дозволяє виконати налаштування властивостей і специфікацію моделі для її публікації в форматі розширюваної мови розмітки XML.

Class Wizard викликає майстер створення нового класу і його розміщення на обраній діаграмі моделі.

При виборі пункту головного меню Add-Ins (Розширення) викликається діалогове вікно Add-In Manager для вибору розширень, з встановлених при інсталяції засобів IBM Rational Rose (рис. 10.26).

Пункт меню Window (Вікно) дозволяє керувати вікнами діаграм та містить такі операції (рис. 10.27).

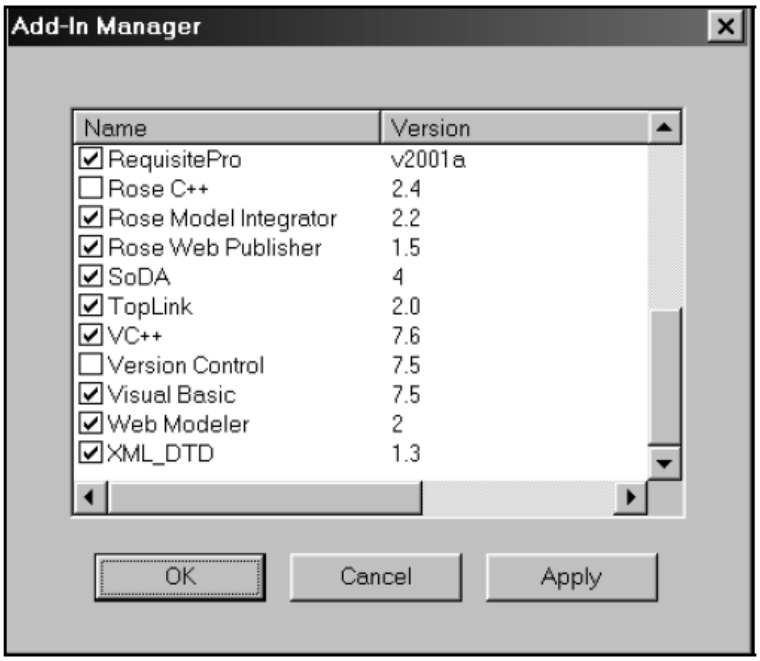



*Рис.* 10.26. Діалогове вікно Add-In Manager (Розширення) головного меню

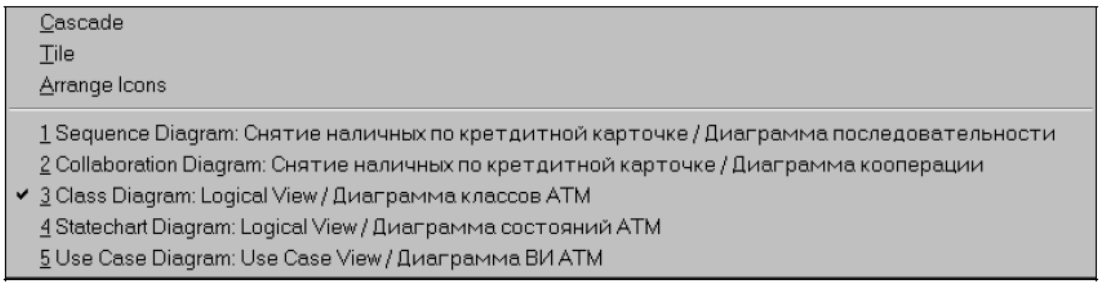



*Рис. 10.27.* Операції пункту меню Window (Вікно) головного меню

Cascade каскадно розміщує всі відкриті діаграми моделі.

Tile відображає всі відкриті діаграми моделі.

Arrange Icons упорядковує розташування всіх відкритих діаграм.

Секція, яка містить імена всіх відкритих діаграм моделі для перемикання між ними. Якщо відкривається нова діаграма, то в цій секції з'являється новий рядок з ім'ям цієї діаграми та її типом, обравши яку, можна відразу перейти в потрібне вікно.

**9. Пункт меню Help** (Довідка) містить такі операції (рис. 10.28).

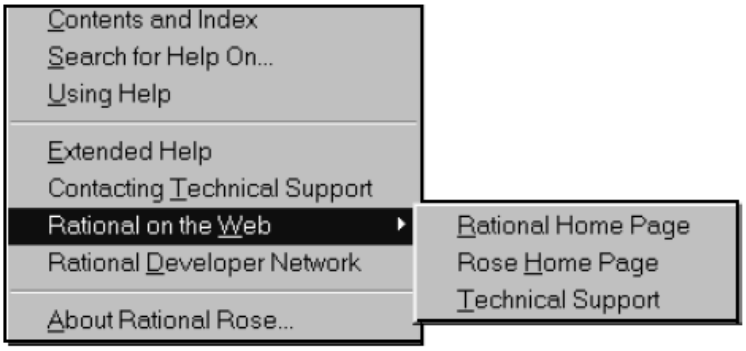



*Рис.* 10.28. Операції пункту меню Help (Довідка) головного меню

Contents and Index викликає програму перегляду довідкової системи, відкритої на вкладці Зміст.

Search for Help On викликає програму перегляду довідкової системи, відкритої на вкладці Покажчик.

Using Help викликає програму перегляду довідкової системи MS Windows.

Extended Help викликає спеціальну програму розширеної довідкової системи.

Contacting Technical Support викликає встановлений в операційній системі за замовчуванням браузер і робить спробу з'єднатися з Web-сайтом технічної підтримки компанії IBM Rational (в разі наявності доступу до Інтернету).

Rational on the Web викликає встановлений в операційній системі за замовчуванням браузер і робить спробу з'єднатися з Web-сайтом компанії IBM Rational (в разі наявності доступу до Інтернету). Вибір окремої операції цього пункту меню визначає завантаження тієї чи іншої Web-сторінки компанії, призначеної для виконання спеціальних дій по додатковій підтримці засобу Rational Rose або завантаження наявних оновлень.

Rational Developer Network викликає встановлений в операційній системі за замовчуванням браузер і робить спробу з'єднатися з Web-сайтом розробників компанії IBM Rational (в разі наявності доступу до Інтернету).

About Rational Rose відображає інформацію про поточну робочу версію IBM Rational Rose. Одним з найбільш цікавих характеристик засобу IBM Rational Rose є можливість генерації програмного коду на основі побудованої моделі. Можливість генерації тексту програми на тій чи іншій мові програмування залежить від встановлених розширень IBM Rational Rose.

## 11. Розробка діаграми класів

*Мета заняття* – ознайомитися із процедурою розробки діаграми класів. *Об'єкт вивчення* – діаграма класів.

План

1. Створення моделі бізнес-аналізу
2. Аналіз системи: архітектурний аналіз

**Завдання**

І. Ознайомитися із процедурою розробки діаграми класів на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Створення структури моделі за вимогами архітектурного аналізу - створення класів

Крок 2. Створення класів, які беруть участь в реалізації конкретного бізнес-процесу (варіанту використання), і діаграми класів.

Крок 3. Додавання атрибутів до класів

Крок 4. Додавання зв'язків між сутностями

Крок 5. Додавання класів-асоціацій

Крок 6. Побудова повної діаграми класів

ІІ. Повторити зазначені дії для свого проєкту.

**Аудиторна робота**

**1. Разробка діаграми класів.** Діаграма класів є основним логічним поданням моделі та містить детальну інформацію про внутрішній устрій об'єктно-орієнтованої програмної системи. Активувати робоче вікно діаграми класів можна декількома способами.

1. Вікно діаграми класів з'являється за замовчуванням в робочому вікні діаграми після створення нового проєкту.

2. Клацнути на кнопці із зображенням діаграми класів на стандартній панелі інструментів.

3. Розкрити логічне подання (Logical View) в браузері і двічі клацнути на піктограмі Main (Головна).

4. Виконати операцію головного меню Browse → Class Diagram (Огляд → Діаграма класів).

При цьому з'являється нове вікно з чистим робочим аркушем діаграми класів і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми класів. Призначення окремих кнопок панелі можна дізнатися також із спливаючих підказок.

На спеціальній панелі інструментів за замовчуванням присутня тільки частина піктограм елементів, які можуть бути використані для побудови діаграми класів. Додати кнопки з піктограмами інших графічних елементів, таких як, наприклад, відношення агрегації, шаблон, клас бізнес-сутність, організаційний підрозділ, або видалити непотрібні кнопки можна за допомогою налаштування спеціальної панелі інструментів. При цьому діалогове вікно налаштувань можна викликати аналогічно іншим панелям, за допомогою

пункту контекстного меню Customize (Налаштування) при позиціонуванні курсора на спеціальній панелі інструментів.

**Додавання класу до діаграми класів.** Для додавання класу до діаграми класів потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми класу на спеціальній панелі інструментів, відпустити ліву кнопку миші і клацнути лівою кнопкою миші на вільному місці робочого аркуша діаграми. На діаграмі з'явиться зображення класу з маркерами зміни його геометричних розмірів і запропонованим за умовчанням ім'ям, яке розробнику слід змінити (рис. 11.1).

*Примітка.* Оскільки передбачається програмна реалізація моделі, що розробляється, імена класів, атрибутів і операцій записані символами латиниці, щоб не ускладнювати візуальне сприйняття діаграм, в цьому випадку можна скористатися способом простої транслітерації відповідних імен російською (цей варіант обраний для подання діаграми класів системи управління банкоматом).

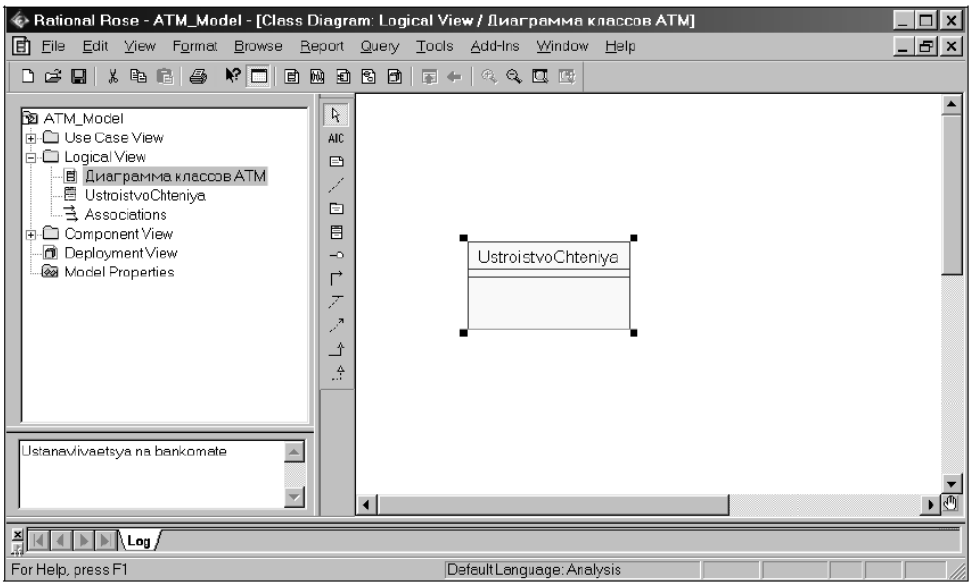



Рис. 11.1. Діаграма класів після додавання на неї класу Пристрій зчитування

**Додавання атрибутів класів.** З усіх графічних елементів клас володіє максимальним набором властивостей, головними з яких є його атрибути і операції. Додати атрибут до створеного раніше класу можна одним з таких способів:

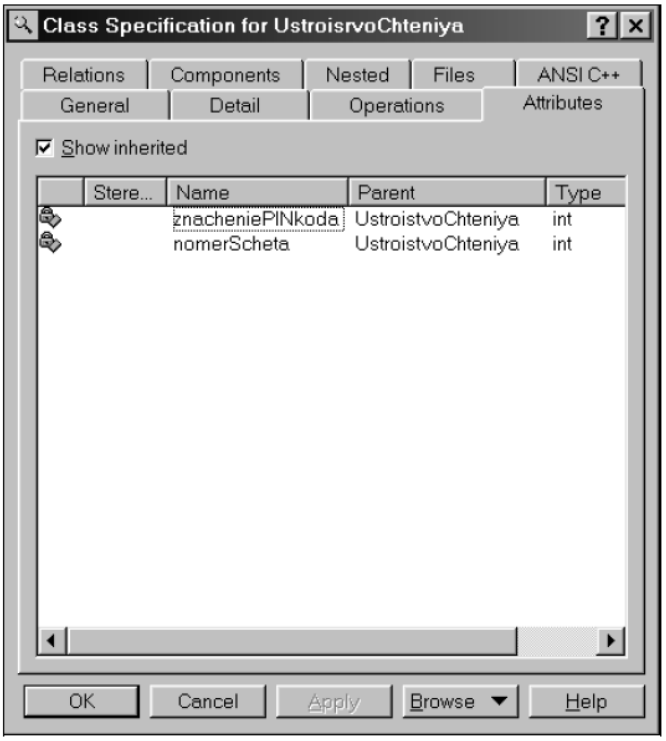
1. За допомогою операції контекстного меню New Attribute (Новий атрибут) для класу, виділеного на діаграмі класів. В цьому випадку активується курсор введення в області графічного зображення класу на діаграмі.

2. За допомогою операції контекстного меню New Attribute (Новий атрибут) для класу, що виділений на діаграмі класів. В даному випадку активується курсор введення у зоні графічного зображення класу на діаграмі.

3. За допомогою операції контекстного меню New Attribute (Новий атрибут) для класу, який виділений у браузері проєкту. В такому випадку

активується курсор введення у зоні ієрархічного подання класса у браузері під назвою відповідного класу.

4. За допомогою операції контекстного меню Insert (Вставити), що був викликаний при позиціюванні курсору у зоні відкритої вкладки атрибутів у діалоговому вікні властивостей Class Specification відповідного класу (рис. 11.2). Відкрити діалогове вікно властивостей класу можна подвійним натисканням лівої кнопки миші на зображенні цього класу на діаграмі чи у браузері проєкту.




є

*Рис.* 11.2. Діалогове вікно налаштування властивостей класу, відкрите на вкладці атрибутів

Після додавання атрибута до класу йому присвоюється за замовчуванням ім’я *name* і деякий квантор видимості. При цьому видимість атрибутів та операцій зображується у формі спеціальних піктограм або прикрашень. Використовувані піктограми видимості зображуються перед ім'ям відповідного атрибута або операції. Для атрибутів класів можна задавати також тип даних і початкові значення, а також призначити стереотип зі списку.

**Додавання операцій класів.** Додати операцію до створеного раніше класу можливо одним із таких способів: за допомогою

1) операції контекстного меню New Operation для класу, виділеного на діаграмі класів; в цьому випадку активується курсор введення в області графічного зображення класу на діаграмі;

2) операції контекстного меню New Operation (Нова операція) для класу, виділеного у браузері проєкту (при цьому активується курсор введення в області ієрархічного подання класу у браузері під ім'ям відповідного класу);

3) операції контекстного меню Insert (Вставити), викликаного при позиціонуванні курсора в області відкритої вкладки операцій у діалоговому вікні властивостей Class Specification відповідного класу.

Після додавання операції до класу їй присвоюється за замовчуванням ім'я opname і деякий квантор видимості. Використовувані піктограми видимості операцій зображуються аналогічно видимості атрибутів. Кожна з операцій класів має власне діалогове вікно властивостей Operation Specification, яке може бути відкрито подвійним натисканням миші на імені операції, на відповідній вкладці властивостей класу або на імені цієї операції в браузері проєкту (рис. 11.3).

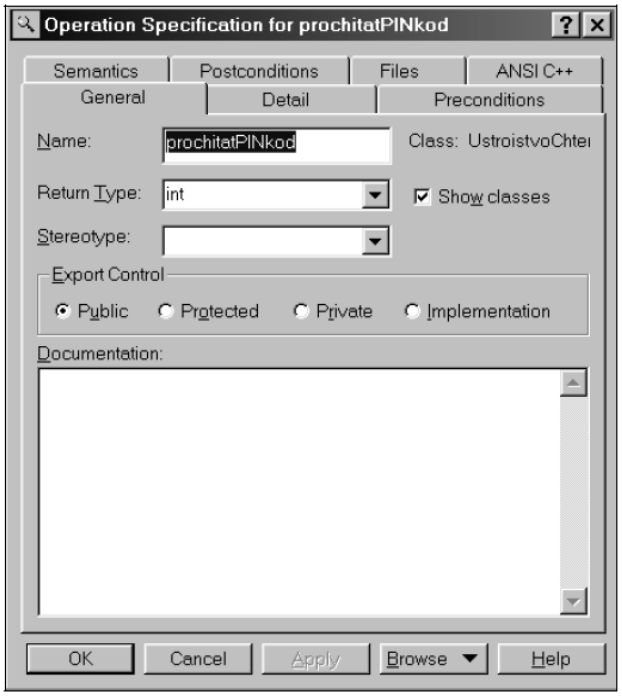



*Рис.* 11.3. Діалогове вікно налаштування властивостей операції класу

Для окремих операцій обраного класу можна задавати такі параметри: тип результату, який повертається, аргументи та їх тип, стереотип, а також визначити протокол і розмір, задати виняткові ситуації, специфікувати перед- і післяумови, а також ряд інших властивостей, детальний розгляд яких можливий тільки в контексті конкретно обраної мови реалізації проєкту.

**Додавання відношень до діаграми класів.** Додавання до діаграми класів відношень між класами, типу асоціації, залежності, агрегації, композиції, реалізації та узагальнення виконується таким чином. На спеціальній панелі інструментів кліком по кнопці з відповідним зображенням вибирається необхідний тип відношення. Якщо відношення є спрямованим, то на діаграмі класів треба виділити перший елемент відношення (джерело, від якого виходить стрілка) і, не відпускаючи натиснуту ліву кнопку миші, перемістити її покажчик до другого елементу відношення (приймач, до якого спрямована стрілка). Після переміщення до другого елементу кнопку миші слід відпустити, і на діаграмі класів буде додано нове відношення.

*Примітка.* За замовчуванням на спеціальній панелі інструментів діаграми класів може бути відсутня кнопка з піктограмою агрегації. В цьому випадку необхідно попередньо додати її на панель інструментів.

Для зображення відношення агрегації можна спочатку зобразити звичайну асоціацію, після чого, відкривши вікно її властивостей на вкладці деталей відповідного кінця асоціації, виставити позначку у рядку вибору Aggregate (Агрегація). Для зображення відношення композиції можна спочатку зобразити звичайну асоціацію, після чого, відкривши вікно її властивостей на вкладці деталей відповідного кінця асоціації, виставити позначку в рядку вибору Aggregate (Агрегація) і вибрати опцію By Value (За значенням) в секції Containment (Включення атрибутів).

Якщо відношення ненаправлене (двонаправлене), то порядок вибору класів для цього відношення довільний. Для відношень можна визначити кратність кожного з кінців, задати ім'я і стереотип, використовувати обмеження і ролі, а також деякі інші властивості. Доступ до діалогового вікна властивостей відношень можна отримати після виділення лінії цього відношення на діаграмі класів або у браузері проєкту і двічі клацнути ліву кнопку миші. Для остаточної побудови діаграми класів розглянутого прикладу слід описаним раніше способом додати решту класів та асоціацій, а також уточнити стереотипи, атрибути і операції цих класів. Побудована таким чином діаграма класів матиме вигляд, зображений на рис. 11.4. На зображеній діаграмі класів обрано текстовий спосіб зображення стереотипів класів, при якому стереотип записується в кутових лапках вище імені відповідного класу.

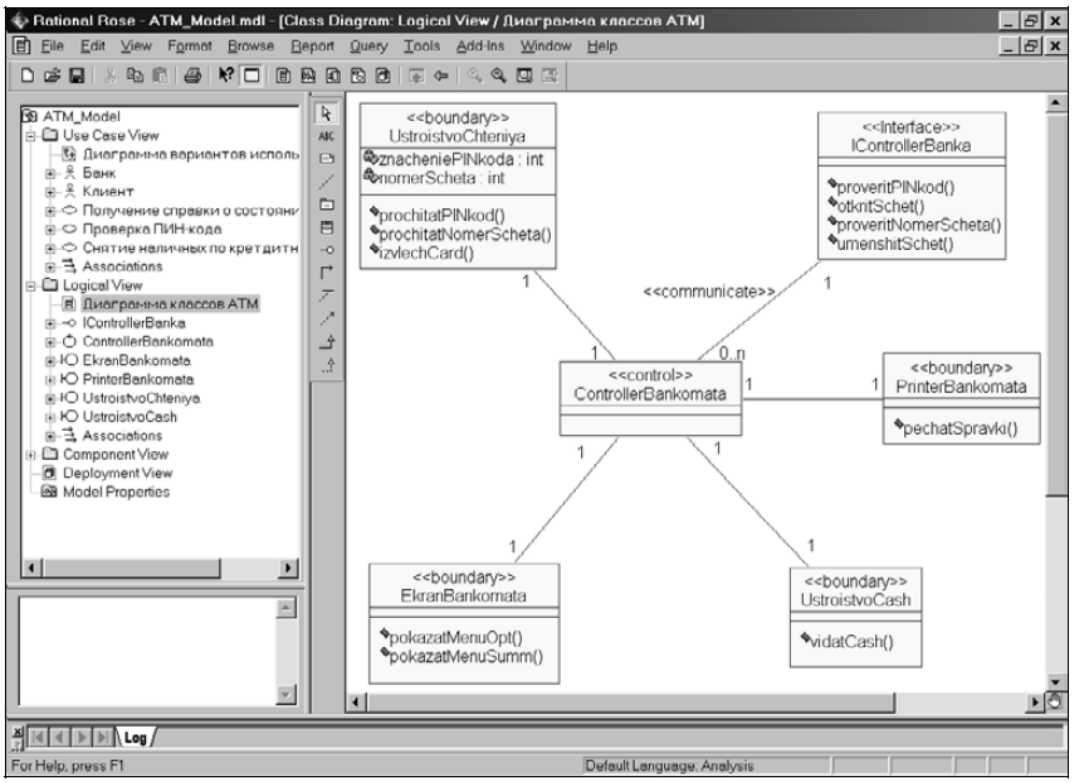



*Рис. 11.4.* Остаточний вигляд діаграми класів розробленої моделі управління банкоматом

Rational Rose дозволяє також зображувати стереотипи у формі графічних зображень (як у браузері проєкту), у формі піктограми у верхній секції зображення класу на діаграмі або взагалі відмовитися від зображення стереотипів. Змінити зображення стереотипу для обраного класу можна, наприклад, за допомогою операції контекстного меню Options → Stereotype Display (Параметри → Зображення стереотипів).

## 12. Розробка діаграми кооперації

*Мета заняття* – ознайомитися із процедурою розробки діаграми кооперації. *Об'єкт вивчення* – діаграма кооперації.

### Завдання

І. Ознайомитися із процедурою розробки діаграми кооперації на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Створення діаграми кооперації

ІІ. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

**1. Розробка діаграми кооперації.** Діаграма кооперації є різновидом діаграми взаємодії і в контексті мови UML описує динамічний аспект взаємодії об'єктів при реалізації окремих варіантів використання. Активувати робоче вікно діаграми кооперації можна декількома способами: 1) клацнути на кнопку із зображенням діаграми взаємодії на стандартній панелі інструментів і вибрати для побудови діаграму кооперації; 2) виконати операцію головного меню Browse → Interaction Diagram (Огляд → Діаграма взаємодії) і вибрати для побудови діаграму кооперації; 3) виконати операцію контекстного меню New → Collaboration Diagram (Нова → Діаграма кооперації) для логічного подання або подання варіантів використання в браузері проєкту. При цьому з'являється нове вікно з чистим робочим листом діаграми кооперації і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми кооперації. Призначення окремих кнопок панелі можна дізнатися з спливаючих підказок. На спеціальній панелі інструментів за замовчуванням присутні практично всі піктограми елементів, які можна використати для побудови діаграми.

У моделі діаграма кооперації відповідає варіанту використання Зняття готівки по кредитній картці і розміщується у поданні варіантів використання (Use Case View). У загальному випадку робота з діаграмою кооперації полягає у додаванні або видаленні об'єктів і повідомлень, а також у фіксації і зміні їх властивостей. При цьому зміни, які вносяться до діаграми кооперації, автоматично вносяться в діаграму послідовності, що можна побачити, активувавши її натисканням клавіші <F5>.

**Додавання об'єкта до діаграми кооперації.** Додати об'єкт до діаграми кооперації можна стандартним чином за допомогою відповідної кнопки на спеціальній панелі інструментів. Однак, в разі наявності побудованої діаграми класів (як в нашому випадку), більш зручним є наступний спосіб. У браузері проєкту виділити необхідний клас і, утримуючи ліву кнопку миші, перетягнути зображення піктограми класу з браузера на вільне місце робочого листа діаграми. В результаті цих дій на діаграмі кооперації з'явиться зображення об'єкта з ім'ям класу і маркерами зміни його геометричних розмірів (рис. 12.1).

За замовчуванням кожен об'єкт, який додається, вважається анонімним. При необхідності можна задати власне ім'я об'єкта, для чого вже відомим способом (наприклад, подвійним клацанням на зображенні об'єкта на діаграмі) слід викликати діалогове вікно властивостей об'єкта (рис. 12.2). Як видно з властивостей цього вікна, для об'єкта обраного класу можна задавати власне ім'я об'єкта, особливості його реалізації і множинність екземплярів.

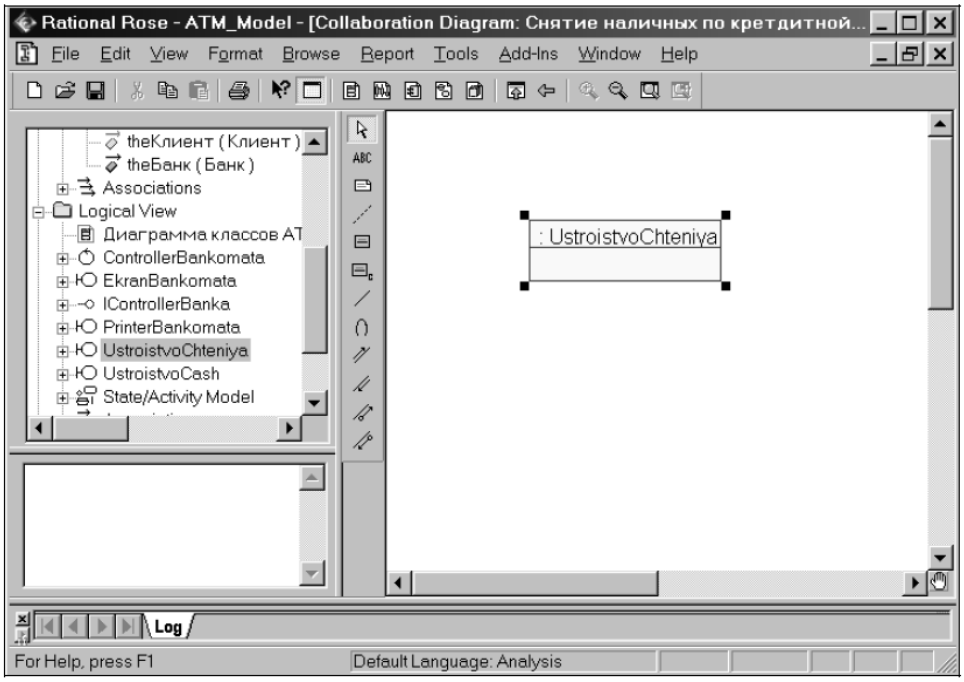



*Рис. 12.1.* Діаграма кооперації після додавання до неї анонімного об'єкту класу UstroistvoChteniya

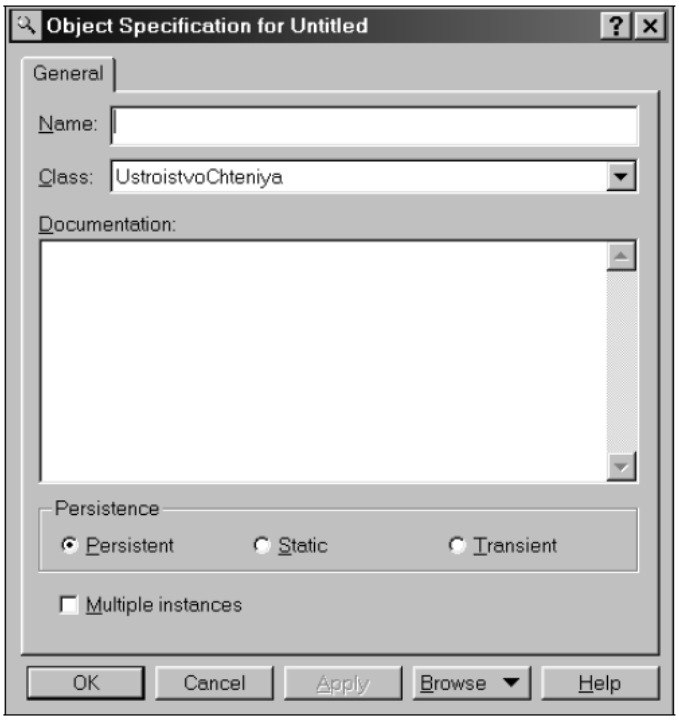



*Рис. 12.2.* Діалогове вікно налаштування властивостей об'єкта

**Додавання зв'язку.** Для додавання зв'язку між попередньо розміщеними на діаграмі об'єктами потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням зв'язку на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні одного об'єкта на діаграмі і відпустити її на зображенні другого об'єкта. В результаті цих дій на діаграмі з'явиться зображення зв'язку, наприклад, що з'єднує об'єкт класу Клієнт (актор) і об'єкт класу UstroistvoChteniya (рис. 12.3). За замовчуванням кожен зв'язок, який додається, вважається анонімним. При необхідності можна задати ім'я зв'язку, для чого вже відомим способом (наприклад, подвійним клацанням на зображенні об'єкта на діаграмі) слід викликати діалогове вікно властивостей зв'язку (рис. 12.4).

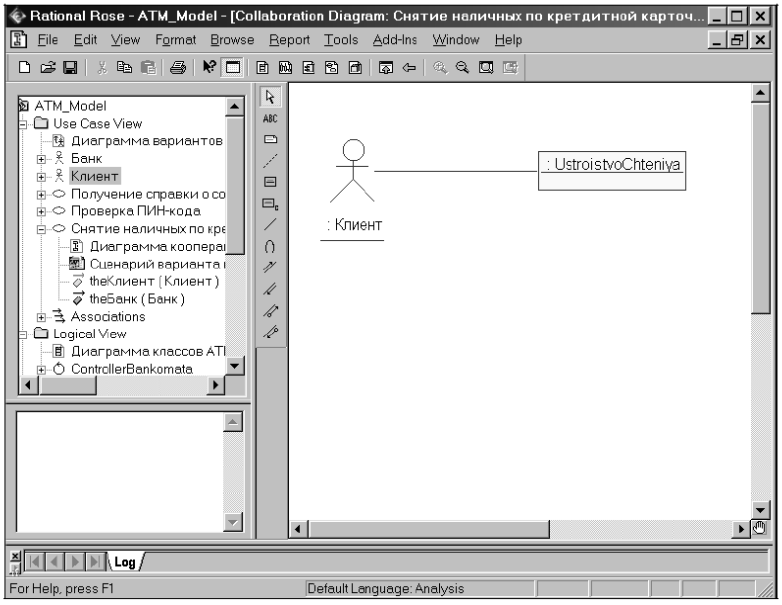



*Рис. 12.3.* Діаграма кооперації після додавання зв'язку між об'єктом класу Клієнт (актор) і об'єктом класу UstroistvoChteniya


*Рис. 12.4.* Діалогове вікно налаштування властивостей зв'язку

Крім імені зв'язку можна також задати такі атрибути: ім'я асоціації, видимість відповідної пари об'єктів і наявність загальних ролей. Однак більш важливою може бути вкладка Messages (Повідомлення), яка служить для специфікації повідомлень, переданих між відповідною парою об'єктів.

**Додавання повідомлення.** Додати повідомлення на діаграмі кооперації можна декількома способами.

1. *Стандартний спосіб* полягає у використанні кнопки з піктограмою повідомлення на спеціальній панелі інструментів. В цьому випадку необхідно лівою кнопкою миші натиснути кнопку із зображенням прямого або зворотного повідомлення на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні лінії зв'язку на діаграмі і відпустити її. В результаті цих дій на діаграмі поряд з лінією зв'язку з'явиться зображення стрілки повідомлення.

2. *Метод додавання повідомлень за допомогою діалогового вікна властивостей зв'язків.* Для цього подвійним клацанням на лінії зв'язку викликається вікно її властивостей і розкривається вкладка Messages (Повідомлення). Після цього слід виконати операцію контекстного меню Insert Te (Вставити в напрямку), в результаті чого з'являється вкладений список з пропозицію вибрати операцію відповідного класу для специфікації повідомлення. Після вибору операції в якості імені додається повідомлення, яке додається в список повідомлень зв'язку з ним (рис. 12.5), а поруч з лінією зв'язку на діаграмі кооперації з'явиться стрілка з номером та ім'ям цього повідомлення. Окрім імені повідомлення можна також задати стереотип синхронізації і частоту передачі. Для цієї мети слід скористатися діалоговим вікном властивостей повідомлень, яке в свою чергу можна відкрити подвійним клацанням на імені повідомлення в списку вкладки Messages (Повідомлення), що розглядається. Для завершення побудови діаграми кооперації розглянутого прикладу слід описаним раніше способом додати об'єкти, що залишилися, зв'язок та повідомлення. Діаграма кооперації, яка описувала реалізацію типового перебігу подій варіанту використання *Зняття готівки кредитною карткою* системи управління банкоматом, буде мати вигляд, зображений на рис. 12.6.


*Рис. 12.5.* Діалогове вікно налаштування властивостей повідомлення

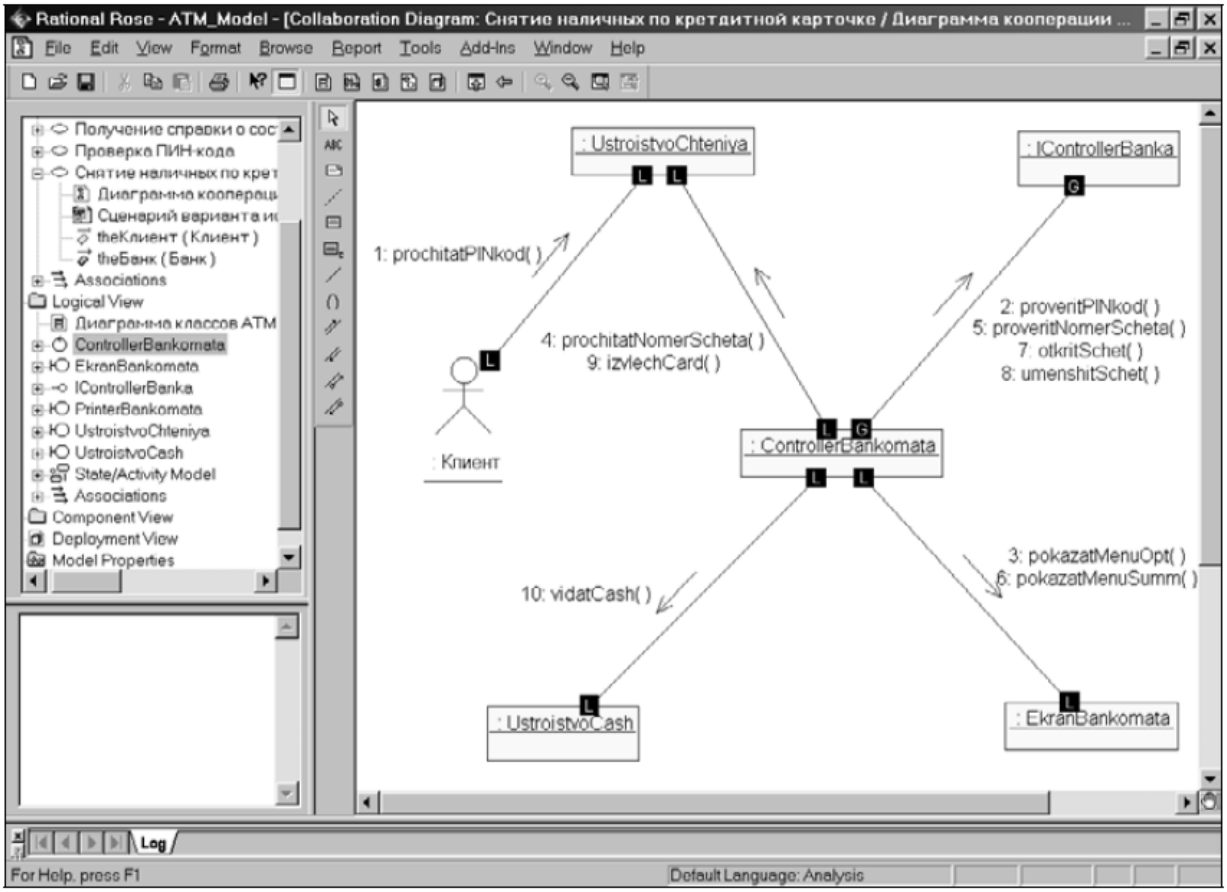



*Рис. 12.6.* Остаточний вигляд діаграми кооперації для варіанту використання Зняття готівки кредитною карткою

Замість текстових стереотипів ролей об'єктів в середовищі Rational Rose локальність і глобальність візуалізується за допомогою невеликих маркерів у формі чорних квадратів з літерами "L" і "G" на границі зображення прямокутника об'єкта. При необхідності можна змінити порядок проходження повідомлень та їх специфікацію, а також встановити додаткову синхронізацію повідомлень і зв'язати з повідомленнями примітки.

## 13. Розробка діаграми послідовності

*Мета заняття* – ознайомитися із процедурою розробки діаграми послідовності. *Об'єкт вивчення* – діаграма послідовності.

### Завдання

І. Ознайомитися із процедурою розробки діаграми послідовності на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Створення діаграми послідовності основного потоку і конкретних потоків.

ІІ. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

**1. Розробка діаграми послідовності.** Діаграма послідовності є іншою формою візуалізації взаємодії в моделі і, як і діаграма кооперації, оперує об'єктами і повідомленнями. Особливість роботи в середовищі Rational Rose полягає в тому, що цей вид канонічної діаграми може бути створений автоматично після побудови діаграми кооперації і натискання клавіші <F5>. За допомогою цієї ж клавіші здійснюється перемикання в моделі між діаграмами послідовності і кооперації.

Однак в окремих випадках буває зручно почати побудову діаграм взаємодії з діаграми послідовності. В цьому випадку активувати робоче вікно діаграми послідовності можна декількома способами:

- клацнути на кнопці із зображенням діаграми взаємодії на стандартній панелі інструментів і вибрати для побудови діаграму послідовності;

- виконати операцію головного меню Browse → Interaction Diagram (Огляд → Діаграма взаємодії) і вибрати для побудови діаграму послідовності;

- виконати операцію контекстного меню New → Sequense Diagram (Нова → Діаграма послідовності) для логічного подання або подання варіантів використання в браузері проєкту.

При цьому з'являється нове вікно з чистим робочим листом діаграми класів і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми послідовності. Призначення окремих кнопок панелі можна дізнатися із спливаючих підказок.

На спеціальній панелі інструментів за замовчуванням присутні практично всі піктограми елементів, які можна використати для побудови діаграми послідовності.

**Додавання об'єкта на діаграму послідовності.** Додати об'єкт на діаграму послідовності можна як стандартним чином за допомогою відповідної кнопки на спеціальній панелі інструментів, так і більш зручним способом - за допомогою перетягування зображення піктограми класу з браузера на вільне місце робочого листа діаграми. В результаті цих дій на діаграмі кооперації

з'явиться зображення об'єкта з ім'ям класу із маркерами зміни його геометричних розмірів і вертикальної пунктирною лінією, яка означає лінію життя цього об'єкта (рис. 13.1).

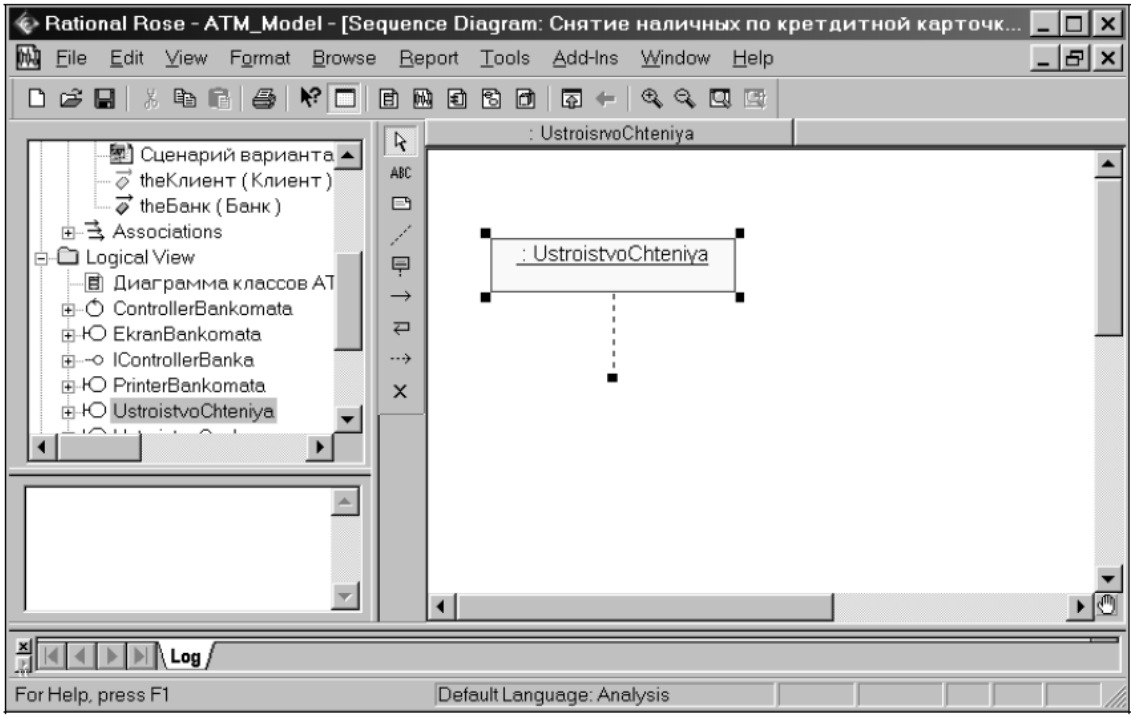



*Рис. 13.1.* Діаграма послідовності після додавання анонімного об'єкту класу UstroistvoChteniya

Як і для діаграми кооперації, для діаграми послідовності кожен об'єкт, що додається за умовчанням вважається анонімним. При необхідності можна задати власне ім'я об'єкта, для чого (наприклад, подвійним клацанням на зображенні об'єкта на діаграмі) слід викликати діалогове вікно властивостей об'єкта, аналогічно об'єктам діаграми кооперації.

**Додавання повідомлення.** Для додавання повідомлення між попередньо розміщеними на діаграмі об'єктами потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням повідомлення на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні лінії життя одного об'єкта на діаграмі і відпустити її на зображенні лінії життя другого об'єкта. В результаті цих дій на діаграмі з'явиться зображення повідомлення, наприклад, що передається від об'єкта класу Клієнт (актор) об'єкту класу UstroistvoChteniya (рис. 13.2). Ім'я повідомлення можна вибрати зі списку операцій відповідного класу-приймача. При необхідності можна задати нову операцію і додати її до опису відповідного класу.

Побудова діаграми послідовності зводиться до додавання або видалення окремих об'єктів і повідомлень, а також до їх специфікації. Доступ до специфікації властивостей цих елементів організований або через контекстне меню, або за допомогою операції головного меню Browse → Specification (Огляд → Специфікація). При додаванні повідомлень до діаграми послідовності вони отримують за замовчуванням свій номер в послідовності.

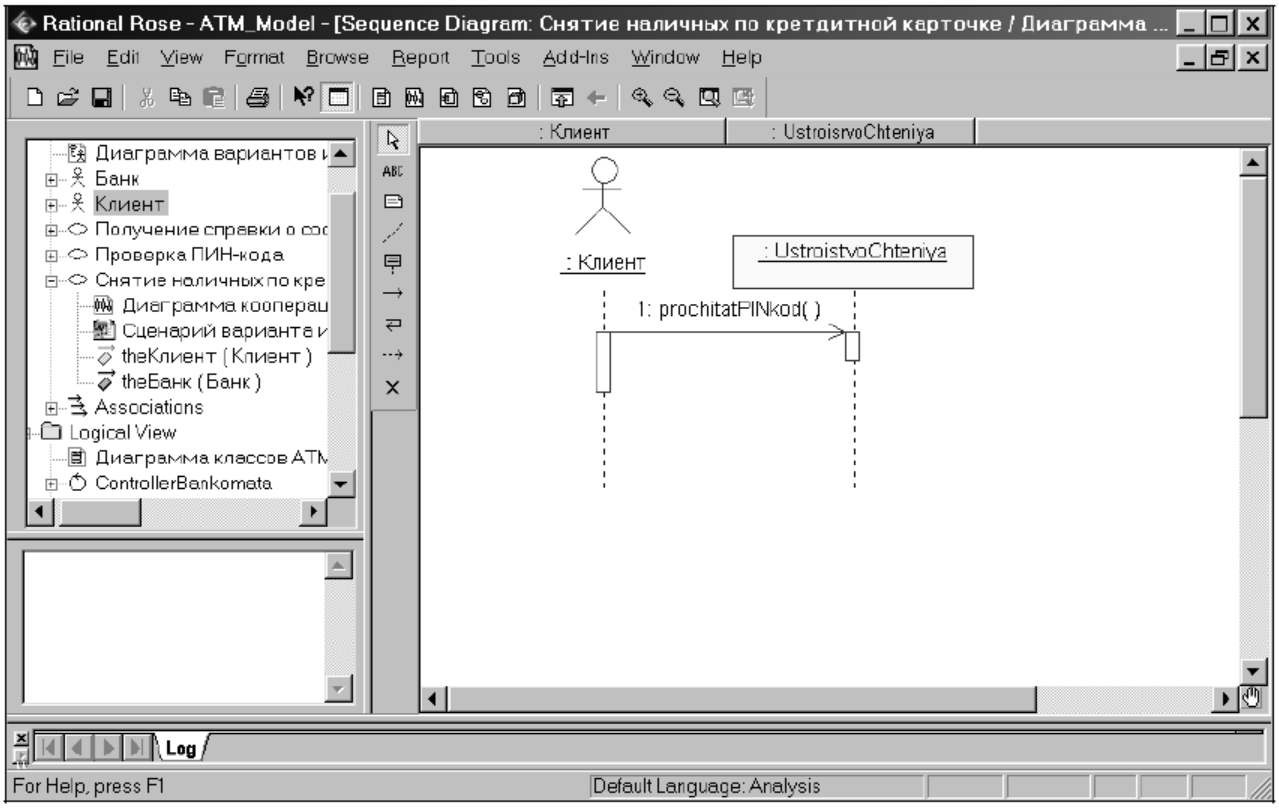



*Рис. 13.2.* Діаграма послідовності після додавання повідомлення між об'єктом класу клієнт (актором) і об'єктом класу UstroistvoChteniya

*Примітка.* За замовчуванням нумерація повідомлень на діаграмі послідовності може бути відключена. При необхідності відобразити номери повідомлень слід виконати операцію головного меню Tools → Options (Інструменти → Параметри), відкрити вкладку Diagram (Діаграма) і виставити позначку вибору рядка Sequence numbering (Нумерація повідомлень на діаграмі послідовності).

Для завершення процесу побудови діаграми послідовності розглянутого прикладу необхідно описаним раніше способом додати об'єкти, що залишилися, а також повідомлення. Діаграма послідовності реалізації типового перебігу подій варіанту використання Зняття готівки кредитною карткою системи управління банкоматом може мати вигляд, зображений на рис. 13.3.

Якщо необхідно змінити порядок проходження повідомлень, то це зручніше зробити на діаграмі послідовності, ніж на діаграмі кооперації. У цьому випадку досить натиснути ліву кнопку миші на стрілці відповідного повідомлення і, не відпускаючи її, перетягнути вертикально вгору або вниз зазначене повідомлення. Додатково можна додавати потоки даних і визначати стійкість об'єктів на основі активації відповідних специфікацій.

Після закінчення сеансу роботи над проєктом виконану роботу необхідно зберегти в файлі проєкту з розширенням MDL. Це можна зробити через меню File → Save (Файл → Зберегти) або File → Save As (Файл → Зберегти як). При цьому вся інформація про проєкт, включаючи діаграми і специфікації елементів, буде збережена в одному файлі.

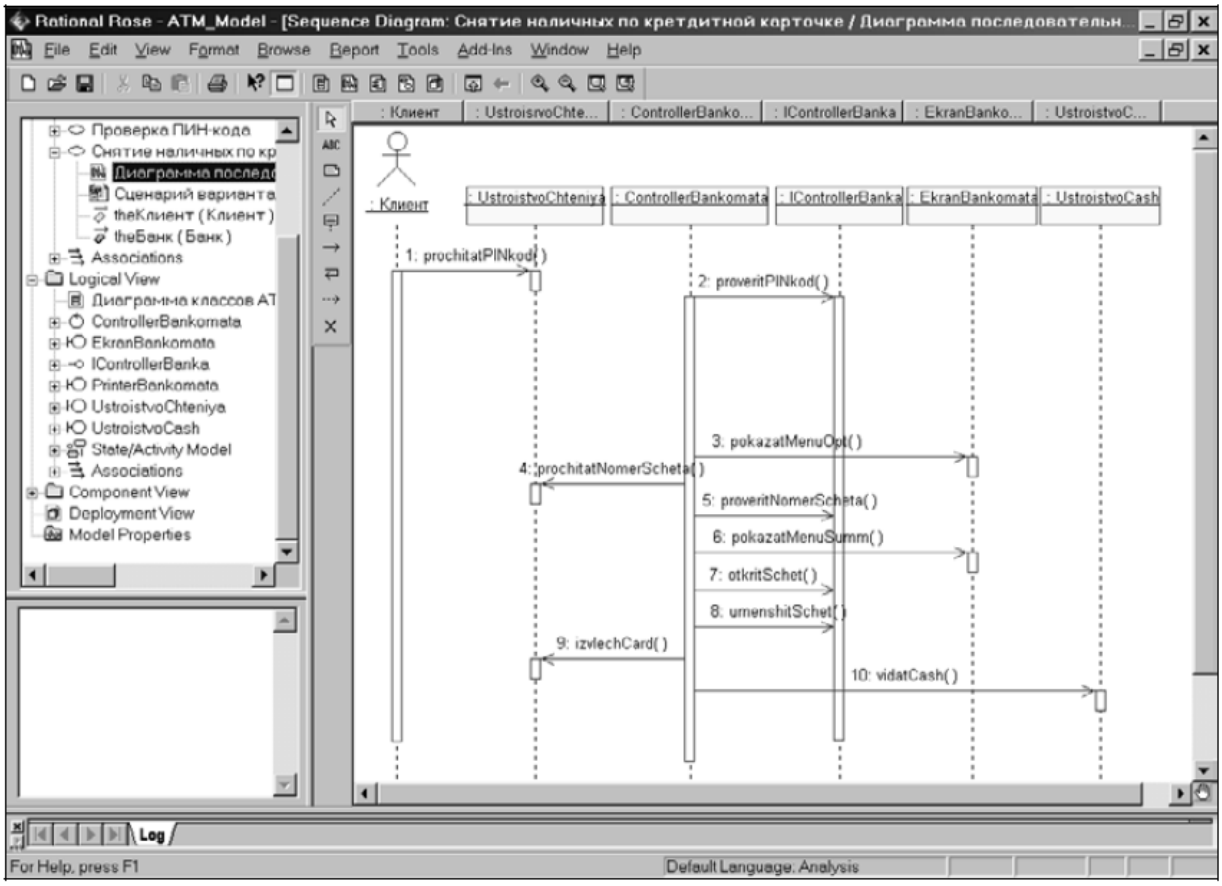



*Рис. 13.3.* Остаточний вигляд діаграми послідовності для варіанту використання Зняття готівки кредитною карткою

*Примітка.* Можна встановити в налаштуваннях опцію автоматичного збереження проєкту через певні проміжки часу. Ця опція за замовчуванням відключена.

Для завершення проєкту системи управління банкоматом потрібно розробити остаточні типи канонічних діаграм, з яких необхідною для генерації програмного коду є лише діаграма компонентів. Проте, з методичних міркувань розроблена модель буде містити всі типи діаграм, включаючи діаграми станів, діяльності та розгортання.

## 14. Розробка діаграми станів

*Мета заняття* – ознайомитися із процедурою розробки діаграми станів. *Об'єкт вивчення* – діаграма станів.

### Завдання

І. Ознайомитися із процедурою розробки діаграми станів на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Моделювання станів для класів

Крок 2. Створення діаграми станів

ІІ. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

#### 1. Розробка діаграми станів в середовищі Rational Rose

Ознайомимося із процесом завершення розробки проєкту в середовищі Rational Rose для чого необхідно розглянути особливості побудови інших 4-х канонічних даіграм. Наприкінці наведено інформацію, необхідну для генерації програмного коду на мові ANSI C++ на основі моделі системи управління банкоматом.

Для продовження роботи над проєктом слід відкрити вже існуючий файл моделі (файл з розширенням MDL). Це можна зробити, наприклад, за допомогою операції головного меню File → Open (Файл → Відкрити) або в результаті подвійного клацання на піктограмі цього файлу. В останньому випадку Rational Rose завантажиться з існуючим проєктом з усіма наявними в ньому діаграмами, специфікаціями і документацією.

Діаграма станів може відноситись до окремого класу, варіанту використання, пакету або подання. Щоб її побудувати для класу або пакета, спочатку необхідно її створити й уточнити. Почати побудову діаграми станів для обраного класу можна одним з таких способів: 1) клацнути на кнопці із зображенням діаграми станів на стандартній панелі інструментів, після чого слід вибрати форму подання і тип розробляючої діаграми станів; 2) розкрити логічне подання (Logical View) в браузері проєкту, виділити розглянутий клас, пакет, варіант використання або подання, після чого виконати операцію контекстного меню New → Statechart Diagram (Нова → Діаграма станів); 3)виконати операцію головного меню Browse → State Machine Diagram (Огляд → Діаграма станів), після чого слід вибрати подання і тип розробляючої діаграми. В результаті виконання цих дій з'являється нове вікно з чистим робочим листом діаграми станів і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми станів. Призначення окремих кнопок панелі можна дізнатися з спливаючих підказок.

За замовчуванням в середовищі відсутні три графічних елемента з розглянутих раніше елементів діаграми станів. При необхідності їх можна додати на спеціальну панель діаграми станів.

Для розроблювального проєкту системи управління банкоматом діаграма станів будується для системи в цілому. В цьому випадку необхідно в браузері проєкту виділити логічне подання (Logical View) і виконати операцію контекстного меню New → Statechart Diagram (Нова → Діаграма станів).

**Додавання стану до діаграми станів.** Для додавання стану до діаграми станів необхідно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми стану на спеціальній панелі інструментів, відпустити ліву кнопку миші і клацнути лівою кнопкою миші на вільному місці робочого листа діаграми. На діаграмі з'явиться зображення стану з маркерами зміни його геометричних розмірів і запропонованим за умовчанням ім'ям, яке розробнику слід змінити (рис. 14.1).

Для доданого стану можна відкрити діалогове вікно його властивостей подвійним клацанням лівою кнопкою миші на зображенні цього елемента на діаграмі. При цьому активується діалогове вікно зі спеціальними вкладками, в поля яких можна занести всю інформацію по даному стану (рис. 14.2). При необхідності, можна визначити такі властивості станів: задати стереотип, специфікувати стан як історичний, визначити внутрішні дії на вході і виході, а також внутрішню діяльність, які доступні для редагування на вкладці Actions (Дії). На вкладці Transitions (Переходи) можна визначати і редагувати переходи, які входять і виходять з розглянутого стану. Остання вкладка Swimlanes (Доріжки) служить для специфікації доріжок, які, в контексті мови UML, визначаються для діаграми діяльності.

**Додавання переходу**. Для додавання переходу між двома станами потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням переходу на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні вихідного стану на діаграмі і відпустити її на зображенні цільового стану.

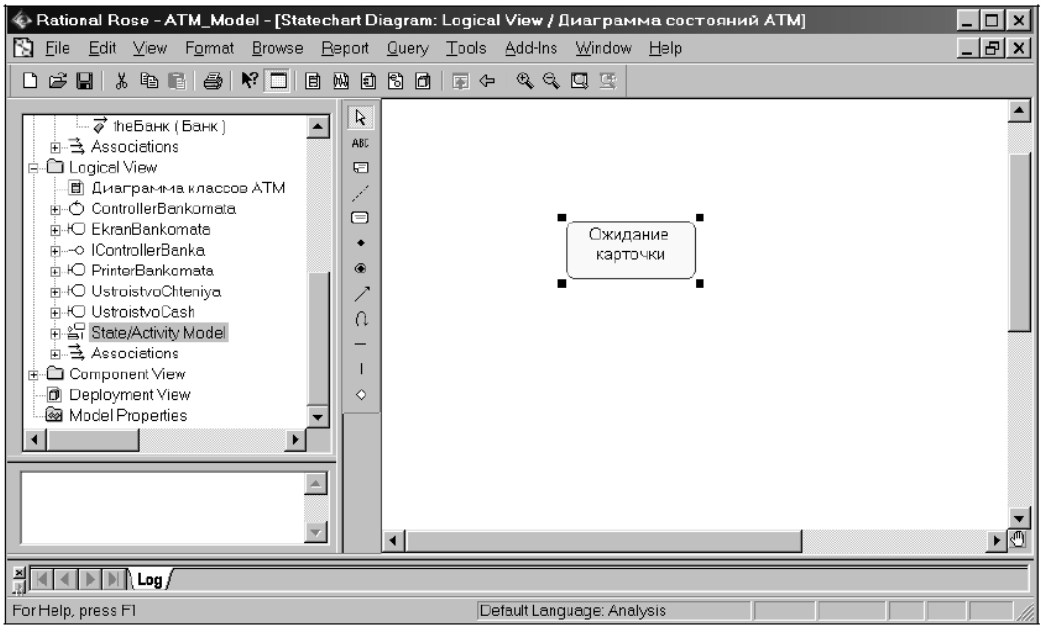



*Рис.* 14.1. Діаграма станів після додавання до неї стану Очікування картки

В результаті цих дій на діаграмі з'явиться зображення переходу, яке з'єднує два обраних стани (рис. 14.3). Після додавання переходу до діаграми станів можна відкрити діалогове вікно його властивостей і специфікувати додаткові властивості, доступні на відповідних вкладках (рис. 14.4). Тут два перші рядки вкладки Detail: перший з рядків Guard Condition служить для фіксації умови, яка визначає правило спрацьовування переходу; у другому рядку — Action — можна специфікувати дію, яка відбувається при спрацьовуванні переходу.

При необхідності можна визначити повідомлення про подію, яка відбувається при спрацьовуванні переходу, а також візуалізувати вкладеність станів і підключити історію окремих станів.

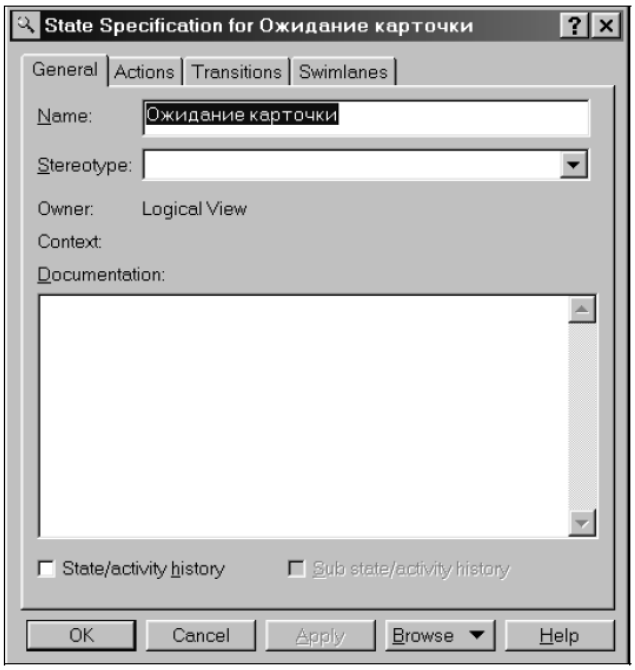



*Рис. 14.2.* Діалогове вікно налаштування властивостей стану

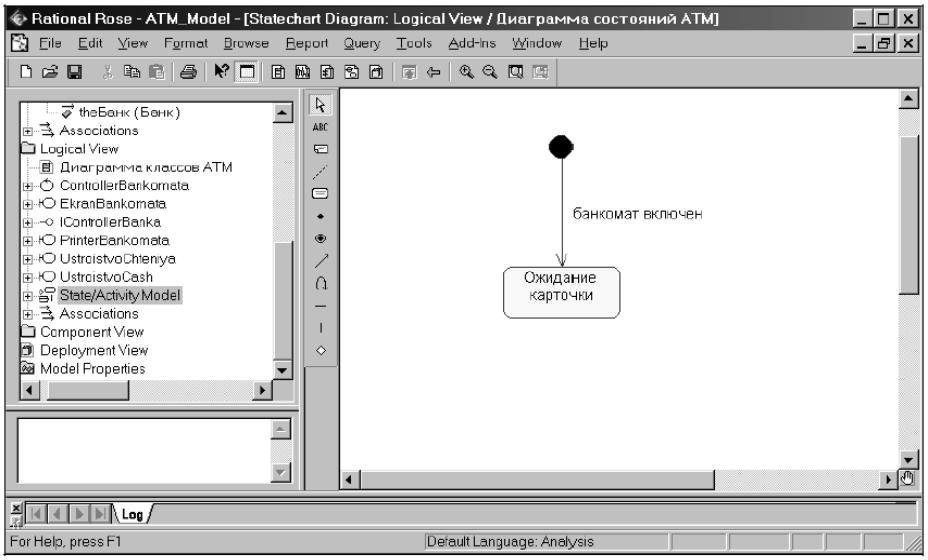



*Рис. 14.3.* Діаграма станів після додавання до неї переходу з початкового стану в стан Очікування картки

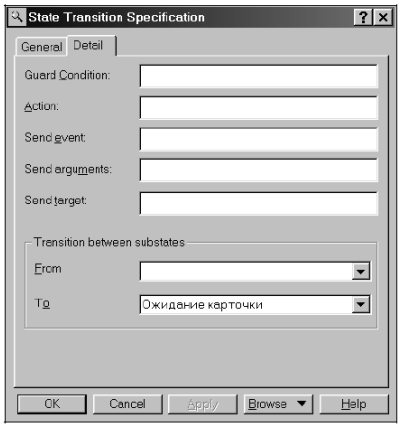



*Рис. 14.4.* Діалогове вікно налаштування властивостей переходу, відкрите на вкладці Detail

Для остаточної побудови діаграми станів розглянутого прикладу слід аналогічним чином додати відсутні стани і переходи. Побудована таким чином діаграма станів матиме вигляд, зображений на рис. 14.5. У розробленій моделі діаграма станів описує поведінку системи управління банкоматом в цілому. Головною перевагою цієї діаграми є можливість візуалізувати умовний характер реалізації всіх варіантів використання в формі зміни окремих станів системи, що розробляється. В Rational Rose ця діаграма не є необхідною для генерації програмного коду. Тому в разі дублювання інформації, поданої на діаграмах взаємодії (кооперації і послідовності), розробку діаграми станів, особливо в умовах дефіциту часу, відпущеного на виконання проєкту, іноді опускають.

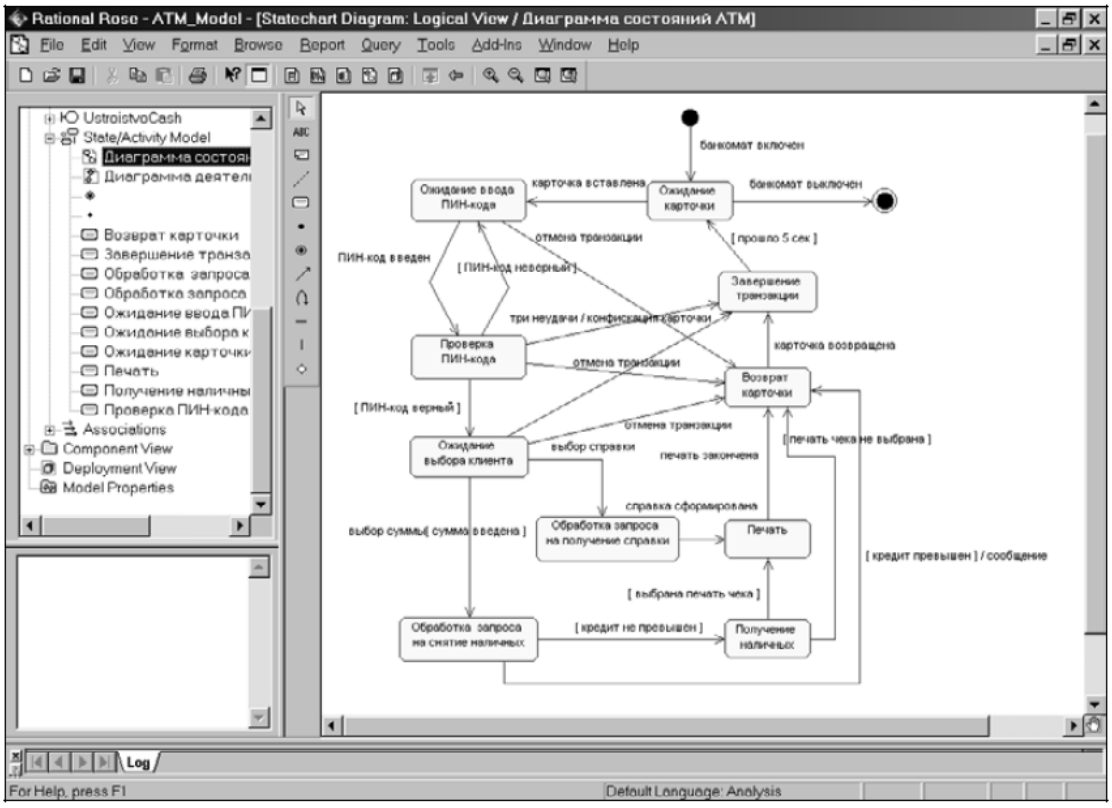



*Рис. 14.5.* Остаточний вигляд діаграми станів моделі управління банкоматом

## 15. Розробка діаграми діяльності

*Мета заняття* – ознайомитися із процедурою розробки діаграми діяльності. *Об'єкт вивчення* – діаграма діяльності.

### Завдання

І. Ознайомитися із процедурою розробки діаграми діяльності на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Створення діаграми діяльності

ІІ. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

**Розробка діаграми діяльності.** В середовищі Rational Rose цей тип діаграм, так само як і діаграма станів, може ставитися до окремого класу, варіанту використання, пакету або поданню. Щоб побудувати діаграму діяльності для класу, пакета або варіанту використання, його спочатку необхідно створити і специфікувати. Після цього виділити на діаграмі або в браузері проєкту.

Почати побудову діаграми діяльності для обраного класу, пакета або варіанту використання можна одним з таких способів:

- клацнути на кнопці із зображенням діаграми станів на стандартній панелі інструментів, після чого слід вибрати подання і тип розробляючої діаграми діяльності;

- розкрити логічне подання (Logical View) в браузері проєкту, виділити розглянутий клас, пакет або подання, після чого виконати операцію контекстного меню **New → Activity Diagram** (Нова → Діаграма діяльності);

- виконати операцію головного меню **Browse → State Machine Diagram** (Огляд → Діаграма станів), після чого слід вибрати подання і тип розробляючої діаграми діяльності.

В результаті виконання цих дій з'являється нове вікно з чистим робочим листом діаграми діяльності і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми діяльності (призначення окремих кнопок панелі можна дізнатися з спливаючих підказок). За замовчуванням на панелі інструментів відсутні тільки два графічних елемента з розглянутих раніше елементів діаграми діяльності, а саме: кнопки з піктограмами об'єкта і потоку об'єктів (при необхідності їх можна додати на спеціальну панель діаграми діяльності стандартним способом).

Для розроблювального проєкту системи управління банкоматом діаграма діяльності описує варіант використання Зняття готівки по кредитній картці. Для зручності включимо діаграму діяльності в логічне подання, для чого необхідно в браузері проєкту виділити логічне подання (Logical View) і виконати

операцію контекстного меню New → Activity Diagram (Нова →Діаграма діяльності).

**Додавання діяльності до діаграми діяльності.** Для додавання діяльності до діаграми діяльності потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми діяльності на спеціальній панелі інструментів, відпустити ліву кнопку миші і клацнути лівою кнопкою миші на вільному місці робочого листа діаграми. На діаграмі з'явиться зображення діяльності з маркерами зміни його геометричних розмірів і запропонованим середовищем ім'ям за умовчанням, яке розробнику слід змінити (рис. 15.1).

Після додавання діяльності на діаграмі станів можна відкрити діалогове вікно її властивостей і специфікувати додаткові властивості, доступні на відповідних вкладках, аналогічно властивостям станів (рис. 14.2). На вкладці Transitions (Переходи) можна визначати і редагувати переходи, які входять і виходять з даної діяльності. Остання вкладка Swimlanes (Доріжки) служить для специфікації доріжки, на яку поміщається розглянута діяльність.

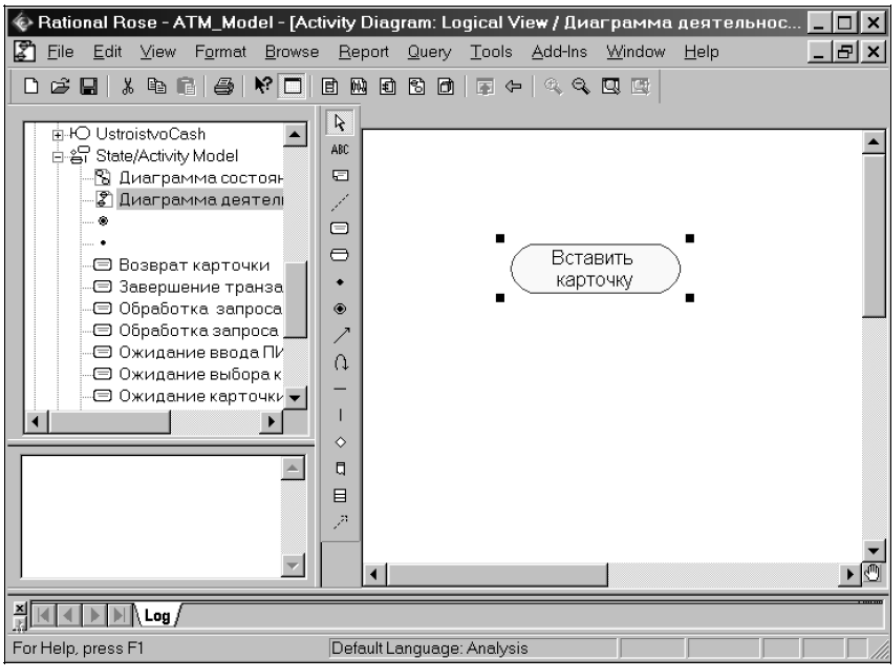



*Рис. 15.1.* Діаграма станів після додавання до неї діяльності Вставити картку

*Примітка.* Хоча система Rational Rose дозволяє визначити властивості діяльності, доступні на вкладці Actions (Дії), слід пам'ятати, що внутрішні дії є властивостями загального стану, а внутрішня діяльність служить ім'ям власне діяльності, що поміщається на діаграмі діяльності. Тому для діяльності, щоб уникнути непорозумінь краще залишити цю вкладку порожньою.

**Додавання переходу**. Додавання переходу до діаграми діяльності аналогічно діаграмі станів, а саме: для додавання переходу між двома діяльностями потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням переходу на спеціальній панелі інструментів, відпустити ліву кнопку миші, клацнути лівою кнопкою миші на зображенні вихідної діяльності на діаграмі і відпустити її на зображенні цільової діяльності. В результаті цих

дій на діаграмі з'явиться зображення переходу, яке з'єднує дві обрані діяльності. Якщо в якості однієї з діяльностей є символ розгалуження або з'єднання, то порядок додавання переходу зберігається незмінним.

Після додавання переходу на діаграмі діяльності стають доступними його властивості у відповідному діалоговому вікні, аналогічному розглянутому раніше (рис. 14.3). При цьому слід пам'ятати, що всі переходи на діаграмі діяльності є нетригерними, тобто не мають імен подій. Але всі переходи, що виходять із символів розгалуження (рішення), повинні мати сторожові умови, які специфікуються на вкладці Detail діалогового вікна властивостей переходу. Для остаточної побудови діаграми діяльності розглянутого прикладу слід додати відсутні діяльності і переходи. Побудована таким чином діаграма діяльності матиме вигляд, зображений на рис. 15.2.

У моделі, що розробляється, діаграма діяльності не є єдиною, оскільки описує лише поведінку системи управління банкоматом при реалізації варіанту використання зняття готівки по кредитній картці. Треба доповнити цю діаграму інформацією щодо реалізації варіанту використання отримання довідки про стан рахунку.

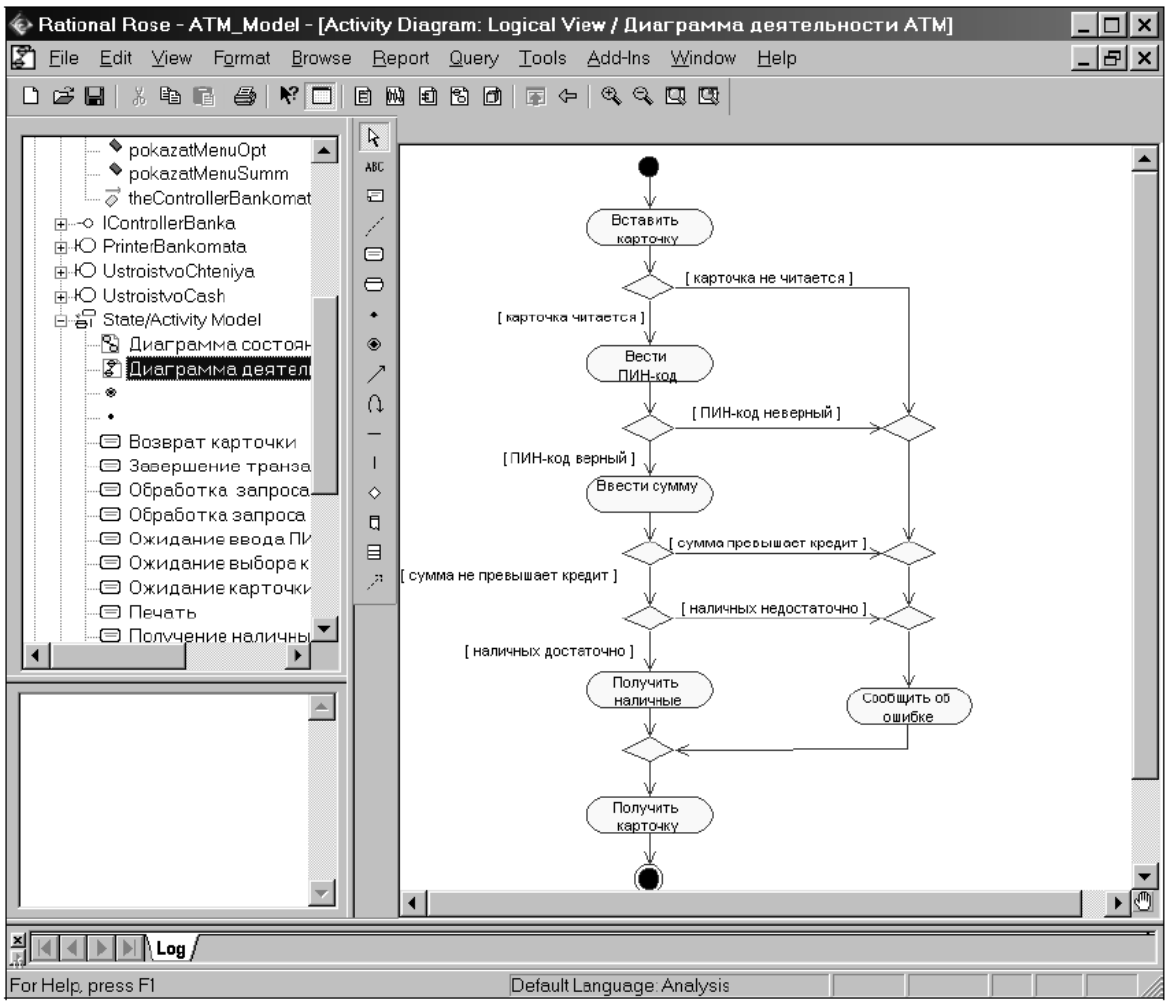



*Рис.* 15.2. Остаточний вигляд діаграми діяльності моделі управління банкоматом, що розробляється

*Примітка.* В рамках вказаної моделі побудова діаграми діяльності для варіанту використання Перевірка пін-коду, що реалізується окремими операціями, може вимагати знань технічних аспектів фізичної реалізації кредитної картки. Тому побудова відповідної діаграми діяльності пов'язана з труднощами концептуального характеру і тут не наводиться.

*У середовищі Rational Rose діаграма діяльності не є необхідною для генерації програмного коду. Тому розробку діаграм цього типу, особливо в умовах дефіциту часу, відпущеного на виконання проєкту, іноді опускають.*

У проєктах реінжинірингу і документування бізнес-процесів діаграма діяльності є основним засобом візуалізації бізнес-процесів в контексті мови UML.

## 16. Розробка діаграми компонентів

*Мета заняття* – ознайомитися із процедурою розробки діаграми компонентів. *Об'єкт вивчення* – діаграма компонентів.

План

1. Реалізація системи

**Завдання**

І. Ознайомитися із процедурою розробки діаграми компонентів на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Створення діаграми компонентів

ІІ. Повторити зазначені дії для свого проєкту.

**Аудиторна робота**

Діаграма компонентів служить частиною фізичного подання моделі, грає важливу роль в процесі ООП і є необхідною для генерації програмного коду. Для діаграм компонентів в моделі призначене окреме подання компонентів (Component View). Активація діаграми компонентів може бути виконана одним із таких способів:

- клацнути на кнопці із зображенням діаграми компонентів на стандартній панелі інструментів;
- розкрити компонентне подання в браузері (Component View) і двічі клацнути на піктограмі **Main** (Головна);
- через пункт меню **Browse → Component Diagram** (Огляд → Діаграма компонентів).

В результаті виконання цих дій з'являється нове вікно з чистим робочим листом діаграми компонентів і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми компонентов. За замовчуванням на панелі інструментів відсутні тільки три графічних елемента з розглянутих раніше елементів діаграми компонентів, а саме — кнопки з піктограмами типових підпрограм і пакета, а також бази даних. При необхідності їх можна додати на спеціальну панель діаграми діяльності стандартним способом.

**Додавання компонента на діаграму компонентів**. Додавання і видалення компонентів на діаграмі компонентів виконується за допомогою кнопок із зображенням піктограми необхідного компонента. Після додавання компонента до діаграми (рис. 16.1) слід визначити його ім'я. За допомогою відповідних маркерів можна змінити також геометричні розміри компонента.

Для кожного компонента можна визначити різні властивості, такі як: стереотип, мова програмування, декларації, реалізовані класи. Специфікація властивостей цих компонентів здійснюється за допомогою діалогового вікна властивостей (рис. 16.2). Зокрема, для компонента Transactions слід вибрати стереотип «DLL» із запропонованого вкладеного списку, оскільки стосовно розроблюваної моделі передбачається реалізація цього компонента в формі бібліотеки динамічного компонування.

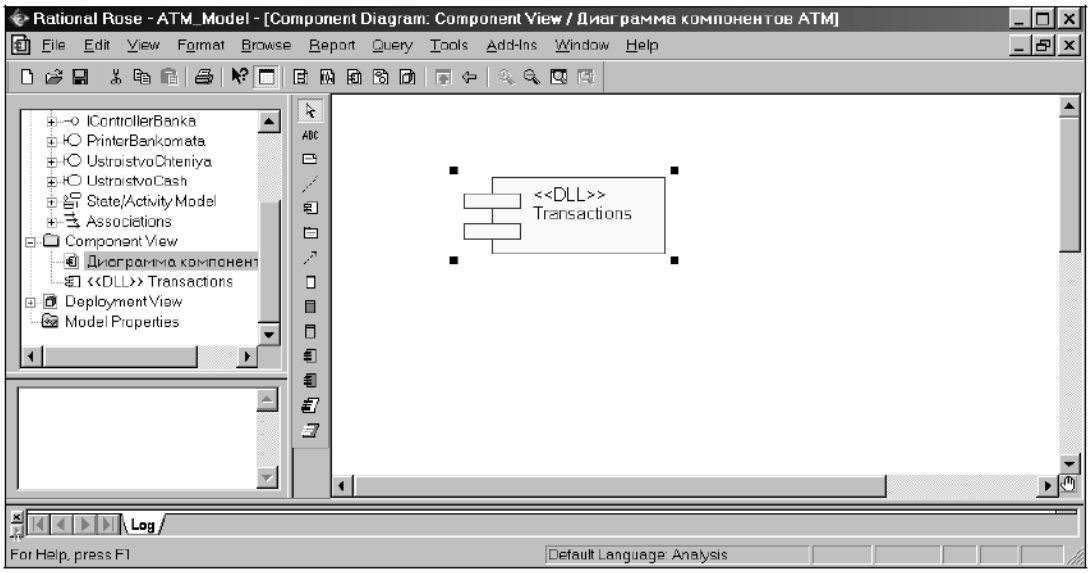



*Рис.* 16.1. Діаграма компонентів після додавання компонента Transactions зі стереотипом бібліотеки динамічного компонування


*Рис.* 16.2. Діалогове вікно налаштування властивостей компонента Transactions

*Примітка.* За замовчуванням для всіх додаваємих до діаграми компонентів в якості мови реалізації вибрана мова аналізу, яку слід змінити на ту мову програмування, яку передбачається використовувати для написання програмного коду. Rational Rose підтримує можливість використання різних мов програмування для реалізації окремих компонентів моделі. Особливості мовної реалізації моделі розглядаються далі.

**Додавання відношення залежності.** Додавання відношення залежності до діаграми компонентів аналогічно додаванню відповідного відношення до діаграми варіантів використання. В результаті на діаграмі компонентів з'явиться зображення відношення залежності між двома компонентами (рис. 16.3).

Для побудови діаграми компонентів розглянутого прикладу слід додати відсутні компоненти, відношення залежності і примітки. Для додавання до діаграми компонентів інтерфейсу слід виділити його в браузері проєкту і перетягнути на робочий лист діаграми. Примітки не є необхідними для даної діаграми, але підвищують наочність візуалізації моделі. Побудована таким чином діаграма компонентів матиме вигляд, зображений на рис. 16.4.

При роботі з діаграмою компонентів можна створювати пакети і розміщувати в них компоненти, використовувати спеціальні графічні стереотипи для окремих різновидів компонентів, змінювати їх специфікацію і відношення залежності між різними елементами діаграми.

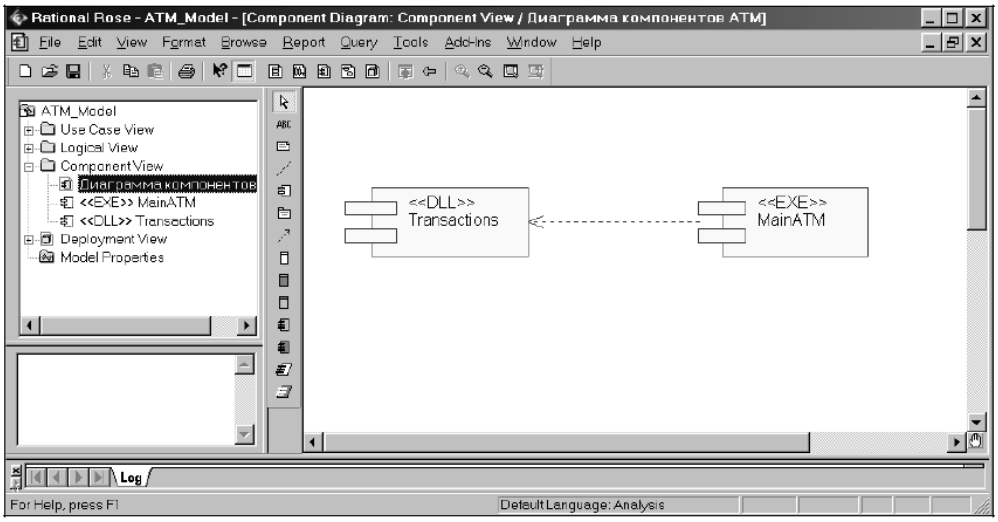



*Рис.* 16.3. Діаграма компонентів після додавання відношення залежності між компонентами Transactions і MainATM

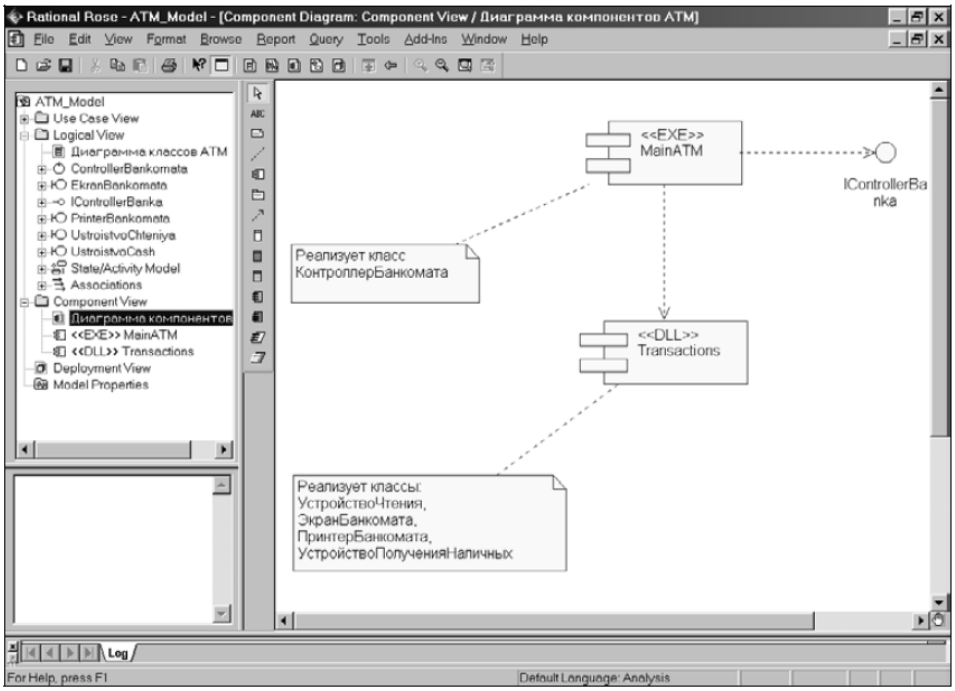



*Рис.* 16.4. Остаточний вигляд діаграми компонентів моделі управління банкоматом, що розробляється

## 17. Розробка діаграми розгортання

*Мета заняття* – ознайомитися із процедурою розробки діаграми розгортання. *Об'єкт вивчення* – діаграма розгортання.

План

1. Проєктування системи:
Проєктування архітектури системи
Формування архітектурних рівнів
Моделювання розподіленої конфігурації системи
Проєктування класів

**Завдання**

І. Ознайомитися із процедурою розробки діаграми розгортання на основі розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Побудова діаграми розгортання

ІІ. Повторити зазначені дії для свого проєкту.

**Аудиторна робота**

**1. Розробка діаграми розгортання**. Діаграма розгортання є другою складовою частиною фізичного подання моделі і розробляється, зазвичай, для територіально розподілених систем. Активацію діаграми розгортання можна виконати одним із таких способів: 1) клацнути на кнопці із зображенням діаграми розгортання на стандартній панелі інструментів; 2) двічі клацнути на піктограмі подання розгортання (Deployment View) в браузері проєкту; 3)виконати операцію головного меню **Browse → Deployment Diagram** (Огляд → Діаграма розгортання). В результаті виконання цих дій з'являється нове вікно з чистим робочим листом діаграми розгортання і спеціальна панель інструментів, яка містить кнопки із зображенням графічних примітивів, необхідних для розробки діаграми розгортання. За замовчуванням на панелі інструментів присутні всі графічні елементи з розглянутих раніше елементів діаграми розгортання, тому змінювати спеціальну панель немає необхідності. Робота з діаграмою розгортання полягає у створенні процесорів і пристроїв, їх специфікації, у встановленні зв'язків між ними, а також додаванні і специфікації процесів.

**Додавання вузла до діаграми розгортання**. Додавання вузла виконується стандартним чином за допомогою кнопок із зображенням піктограми необхідного вузла — процесора або пристрою. Після додавання вузла до діаграми (рис. 17.1) слід визначити його ім'я, а за допомогою відповідних маркерів можна змінити його геометричні розміри.

Для кожного процесора можна визначити різні властивості, такі як стереотип, характеристика, процеси та їх пріоритет. Специфікація цих властивостей процесора здійснюється за допомогою діалогового вікна властивостей (рис. 17.12). При цьому для завдання стереотипу слід ввести його текст без кутових лапок в рядок з ім'ям **Stereotype.**

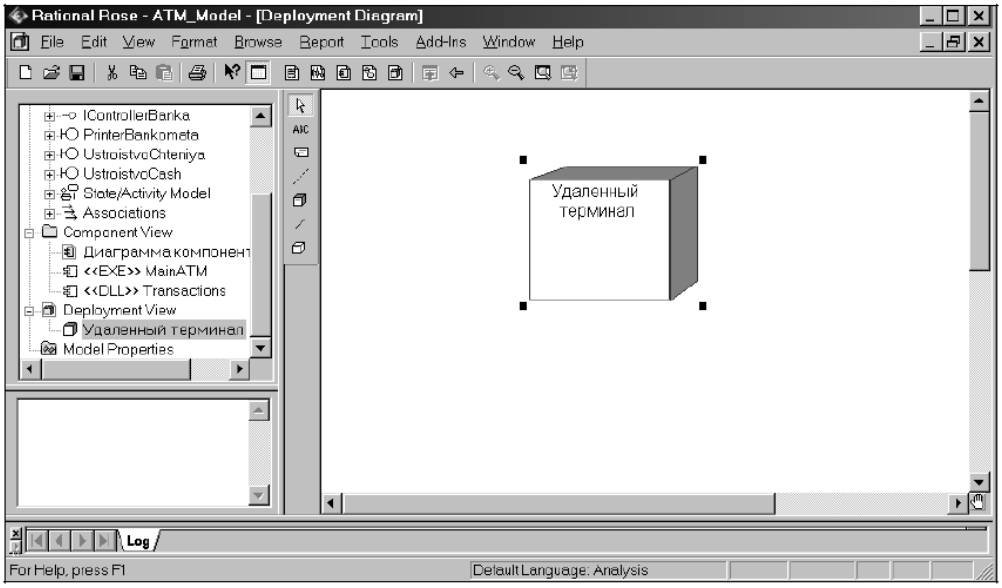



*Рис. 17.1.* Діаграма розгортання після додавання вузла Віддалений термінал


*Рис. 17.2.* Діалогове вікно налаштування властивостей вузла Віддалений термінал

Для пристрою набір редагованих властивостей менше, оскільки для нього за допомогою відповідного вікна властивостей можна визначити тільки стереотип і характеристику. Цей факт узгоджується з визначенням пристрою як вузла, на якому відсутній процесор.

**Додавання з'єднання.** Додавання з'єднання до діаграми розгортання здійснюється способом, аналогічним раніше розглянутим додаванням стрілок різних відношень. В результаті вибору з'єднання на діаграмі розгортання з'явиться зображення лінії з'єднання між двома вузлами (рис. 17.3).

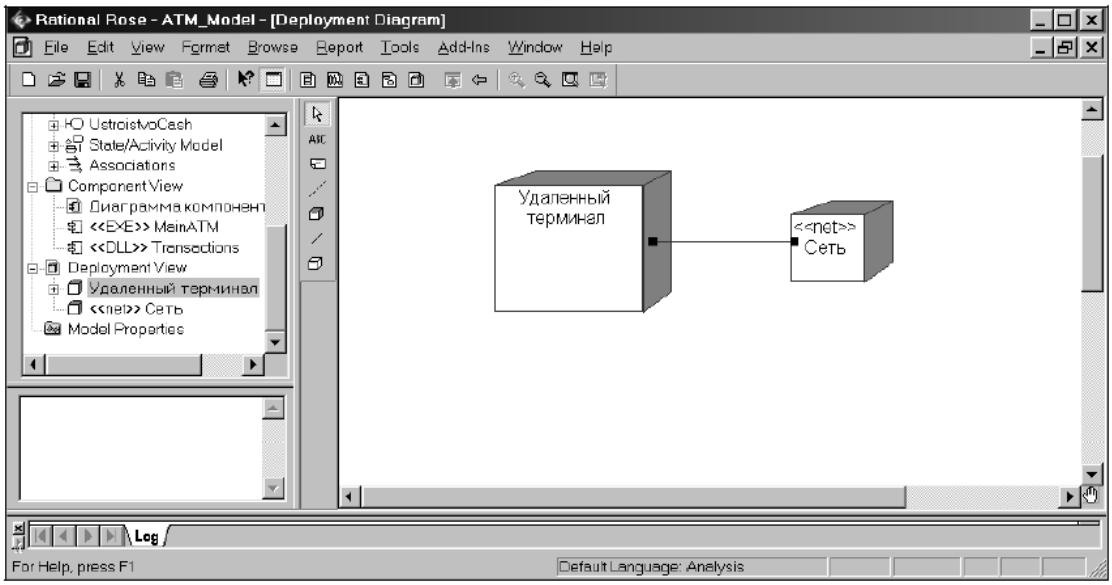



*Рис. 17.3.* Діаграма розгортання після додавання з'єднання між вузлами Віддалений термінал і Мережа

Для остаточної побудови діаграми розгортання даного прикладу слід додати відсутні вузли і з'єднання між ними. Побудована таким чином діаграма компонентів матиме вигляд, зобрбажений на рис. 17.4.

Побудовою діаграми розгортання завершується розробка моделі в нотації мови UML для системи управління банкоматом.

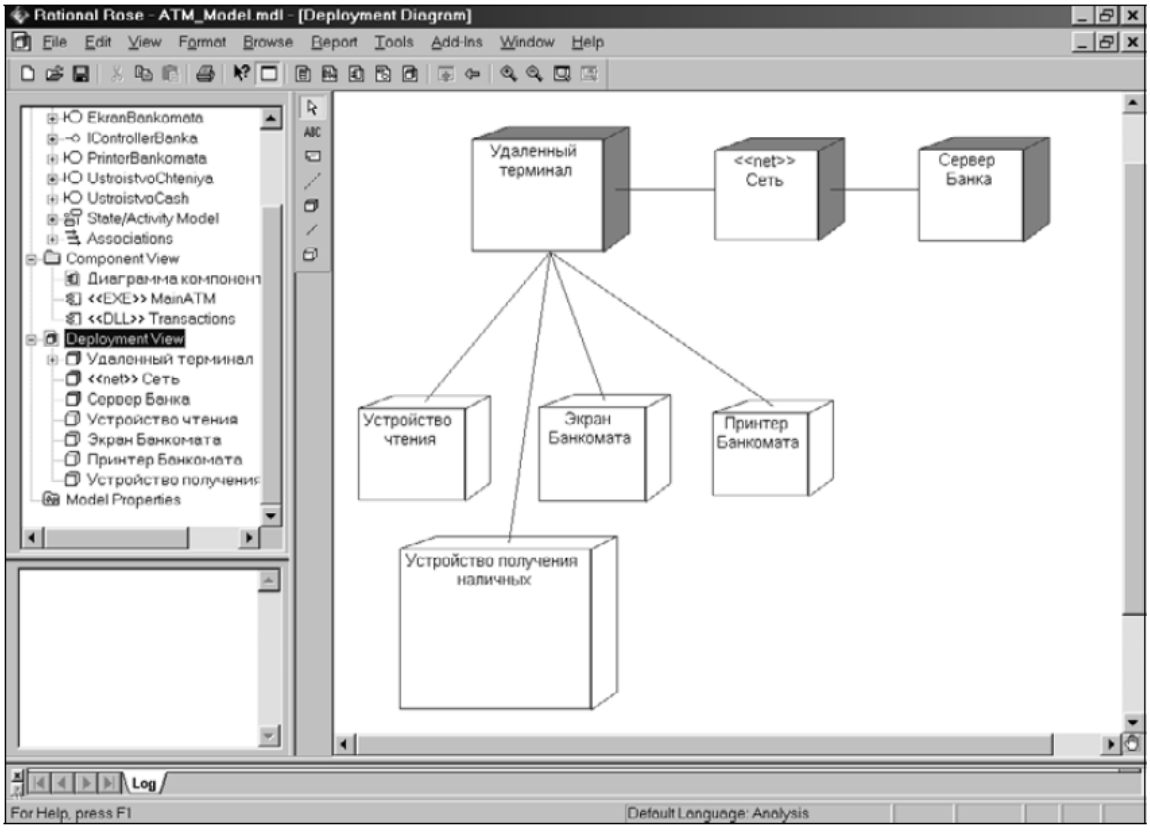



*Рис. 17.4.* Остаточний вигляд діаграми розгортання моделі управління банкоматом, яку розробляють

Подальша робота з моделлю залежить від цілей виконання проєкту. Якщо проект не передбачає програмну реалізацію, то можна обмежитися формуванням проєктної документації. Для цього слід виконати операцію головного меню **Report → SoDA Report** (Звіт → Звіт за допомогою SoDA), в результаті чого буде відкрито діалогове вікно властивостей для вибору шаблоном генерації звіту. Після вибору шаблонів буде автоматично згенерований звіт про модель, що розробляється, в форматі MS Word з використанням спеціального засобу Rational SoDA (якщо воно було встановлено в системі при інсталяції Rational Rose).

Якщо проєкт передбачає програмну реалізацію, то слід познайомитися з особливостями генерації програмного коду в середовищі Rational Rose на основі розробленої моделі.

## 18. Генерація програмного коду

*Мета заняття* – ознайомитися із процедурою Генерації програмного коду. *Об'єкт вивчення* – процедура Генерації програмного коду.

### План

1. Генерація коду

### Завдання

I. Ознайомитися із процедурою Генерації програмного коду для розробляємого проєкту у вигляді моделі системи управління банкоматом.

Крок 1. Генерація коду на мові C++

II. Повторити зазначені дії для свого проєкту.

### Аудиторна робота

Одним із найбільш потужних властивостей є можливість генерації програмного коду після побудови моделі. Для цього є великий вибір мов програмування і схем баз даних, проте можливість генерації тексту програми на тій чи іншій мові програмування залежить від встановленої версії Rational Rose. Загальна послідовність дій, які необхідно виконати для генерації програмного коду, складається із таких кроків:

1. Перевірка моделі незалежно від вибору мови генерації коду.
2. Створення компонентів для реалізації класів.
3. Відображення класів на компоненти.
4. Вибір мови програмування для генерації тексту програмного коду.
5. Встановлення властивостей генерації програмного коду.
6. Вибір класу, компонента або пакета.
7. Генерація програмного коду.

Особливості виконання кожного з етапів можуть змінюватися в залежності від вибору мови програмування або схеми бази даних. У середовищі Rational Rose передбачено задавання різних властивостей, які характеризують як окремі класи, так і проєкт у цілому. Для визначеності виберемо в якості мови реалізації проєкту мову програмування ANSI C++, яка не вимагає вивчення додаткових пакетів бібліотек і поставляється практично в усіх конфігураціях Rational Rose. Розглянемо особливості виконання кожного з етапів для вибраної мови програмування ANSI C ++.

**Перевірка моделі незалежно від вибору мови генерації коду**

У загальному випадку перевірка моделі може виконуватися на будь-якому етапі роботи над проєктом. Однак після завершення розробки графічних діаграм вона є обов'язковою, оскільки дозволяє виявити цілий ряд помилок розробника. До числа таких помилок і попереджень відносяться, наприклад, невикористовувані асоціації і класи, що залишилися після видалення окремих графічних елементів з діаграм, а також операції, які не є іменами відповідних повідомлень.

Для перевірки моделі слід виконати операцію головного меню **Tools → Check Model** (Інструменти → Перевірити модель). Результати перевірки розробленої модель на наявність помилок відображаються у вікні журналу. Перш ніж приступити, власне, до генерації тексту програмного коду розробнику слід домогтися усунення всіх помилок і попереджень, про що має свідчити чисте вікно журналу (рис. 18.1).


*Рис. 18.1.* Вигляд журналу при відсутності помилок в результаті перевірки моделі

**Створення компонентів для реалізації класів.** Даний етап виконується під час розробки діаграми компонентів. Однак середовище IBM Rational Rose дозволяє генерувати текст програмного коду на мові ANSI C++ для кожного класу моделі без попередньої побудови діаграми компонентів (оскільки стосовно проєкту, що розробляється, даний етап вже виконано, немає необхідності повторно описувати особливості побудови діаграми компонентів).

**Відображення класів на компоненти.** Для відображення класів на компоненти можна скористатися вікном специфікації властивостей компонента, відкритого на вкладці **Realizes** (Реалізації). Для включення реалізації класу в даний компонент слід виділити необхідний клас на цій вкладці і в контекстному меню (рис. 18.2), що викликається по кліку правої кнопки миші, виконати операцію Assign (Призначити).

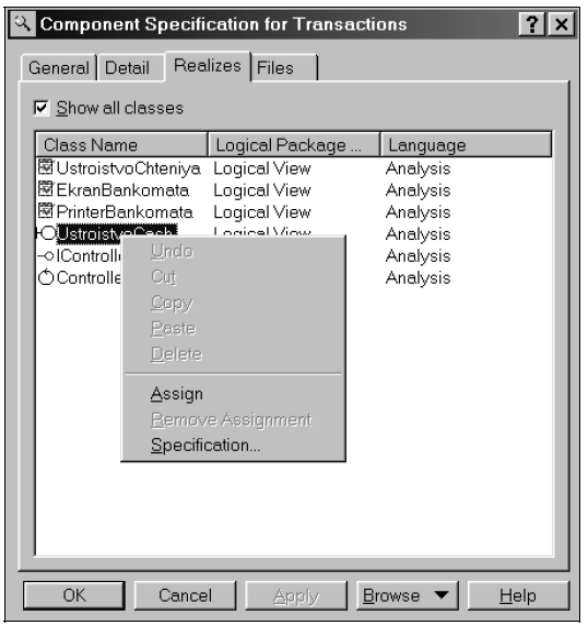



*Рис. 18.2.* Діалогове вікно налаштування властивостей реалізації класів в компоненті Transactions

Після призначення обраного класу для реалізації, в даному компоненті з'явиться відмітка зліва в рядку з ім'ям цього класу. Подібна операція повинна бути виконана для всіх класів моделі, які передбачається реалізовувати на обраній мові програмування.

**Вибір мови програмування для генерації тексту програмного коду.** Для вибору мови ANSI C++ в якості мови реалізації моделі слід виконати операцію головного меню Tools Options (Інструменти → Параметри), в результаті чого буде викликано діалогове вікно налаштування параметрів моделі. Далі, на вкладці Notation (Нотація), в рядку Default Language (Мова за замовчуванням) з вкладеного списку слід вибрати мову ANSI C++.

*Примітка.* Якщо у вкладеному списку мови ANSI C++ не виявилося, то слід переконатися в тому, що ця мова встановлена в якості розширення Rational Rose. Для цього слід відкрити вікно встановлених розширень, виконавши операцію головного меню Add-Ins (Розширення), і переконатися в тому, що виставлена відмітка в рядку з ім'ям мови ANSI C++. Якщо її немає, то її слід додати, після чого з'явиться група доступних операцій ANSI C++ головного меню Tools.

Після вибору мови програмування за замовчуванням слід змінити мову реалізації кожного з компонентів моделі. З цією метою слід змінити мову аналізу на вкладці General (Загальні) в рядку Language (Мова), для чого з вкладеного списку слід вибрати мову ANSI C++.

**Встановленняа властивостей генерації програмного коду.** Редагування загальних властивостей генерації програмного коду можливо в спеціальному діалоговому вікні, яке можна відкрити в результаті виконання операції головного меню **Tools → ANSI C++ → Open ANSI C++ Specification** (Інструменти → Мова ANSI C++ → Відкрити специфікацію мови ANSI C++).

Додаткові властивості генерації програмного коду окремого класу можна уточнювати в діалоговому вікні, яке можна відкрити в результаті виконання операції контекстного меню **ANSI C++ Class Customization** (Мова ANSI C++ → Налаштування властивостей класу). При цьому відповідний клас повинен бути виділений в браузері проєкту. Стосовно генерації програмного коду на мові ANSI C++, то можна залишити без зміни значення властивостей, запропонованих середовищем Rational Rose за замовчуванням.

**Вибір класу, компонента або пакета для генерації програмного коду.** Вибір класу, компонента або пакета для генерації програмного коду означає виділення відповідного елемента моделі в браузері проєкту. Що стосується аналізованої моделі системи управління банкоматом для генерації програмного коду на мові ANSI C++ виберемо компонент з ім'ям Transactions.

**Генерація програмного коду.** Генерація програмного коду в середовищі Rational Rose можлива для окремого класу, пакета або компонента. Для цього потрібний елемент моделі попередньо слід виділити в браузері проєкту і виконати операцію контекстного меню **ANSI C++ → Generate Code** (Мова ANSI C++ → Генерувати код). В результаті цього буде відкрито діалогове вікно з пропозицією вибрати папку для розміщення файлів з текстом програмного

коду на обраній мові програмування. Після вибору відповідної папки і закриття цього вікна Rational Rose виконує генерацію коду.

Для перегляду і редагування створених файлів з текстом програмного коду на мові ANSI C++ використовують вбудований текстовий редактор, який можна відкрити за допомогою операції контекстного меню **ANSI C++ → Browse Header** (Мова ANSI C++ → Переглянути заголовки) або **ANSI C++ → Browse Body** (Мова ANSI C++ → Переглянути файл реалізації).

Що стосується аналізованої моделі системи управління банкоматом програмного коду на мові ANSI C++ виберемо компонент з ім'ям Transactions. Після генерації програмного коду в обраній папці з'являться 8 файлів з текстом коду на мові ANSI C++. При цьому кожному класу, наприклад, ustroistvoChteniya, реалізованому на цьому компоненті, будуть відповідати два файли — заголовний, з розширенням h (рис. 18.3) і файл реалізації з розширенням cpp (рис. 18.4).

Як видно з розгляду отриманого фала заголовку, в ньому міститься оголошення всіх атрибутів і операцій класу UstroistvoChteniya, відповідно до правил синтаксису мови ANSI++. У файлі UstroistvoChtenie.cpp міститься заготовка для реалізації відповідно до правил синтаксису мови ANSI C++ всіх операцій класу UstroistvoChteniya (рис. 18.4).

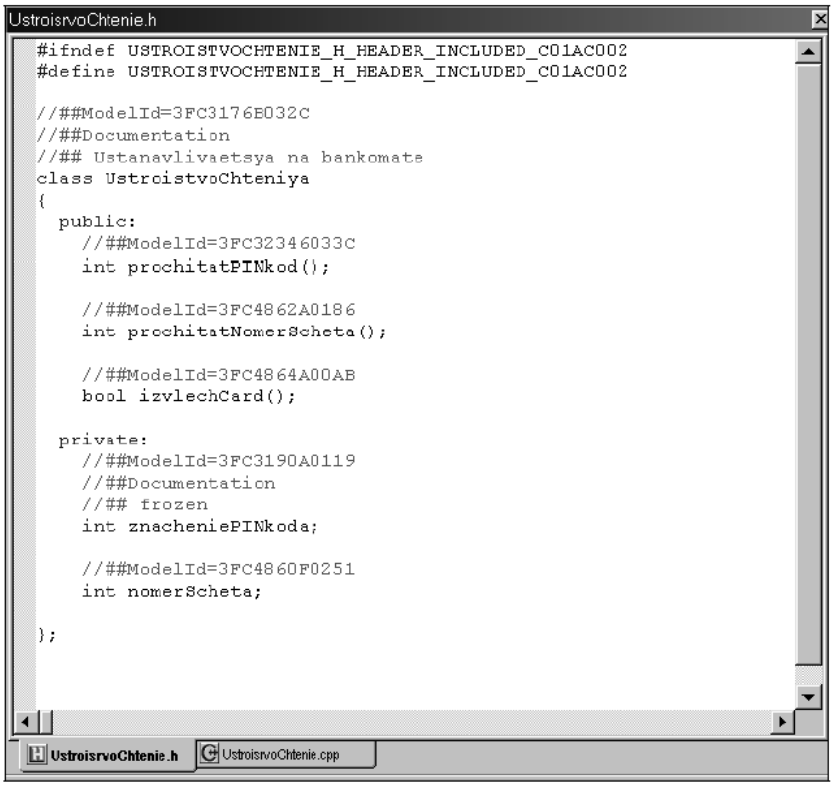



*Рис. 18.3.* Вигляд вбудованого текстового редактора з завантаженим в нього заголовним файлом UstroistvoChtenie.h

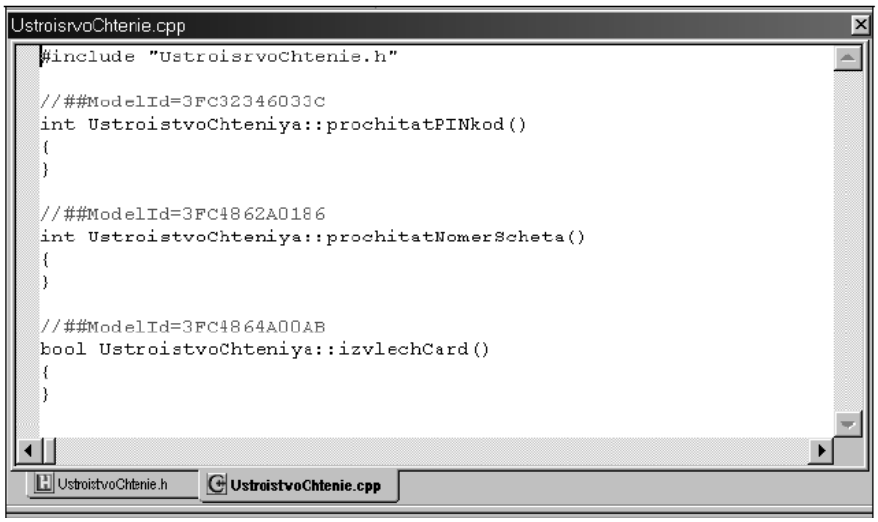



*Рис. 18.4.* Вигляд вбудованого текстового редактора з завантаженим в нього файлом реалізації UstroistvoChtenie.cpp

При цьому кожна з операцій має порожнє тіло реалізації, яке повинен написати програміст самостійно, виходячи з функціональних вимог моделі і синтаксису мови програмування ANSI C++. Вказану роботу зручно виконувати в середовищі програмування, наприклад, MS Visual C++. При використанні середовища програмування, крім компіляції, налагодження і тестування вихідних модулів програми розробник отримує можливість доповнити додаток графічним інтерфейсом, необхідним для взаємодії з користувачем.

*Примітка.* При встановленні на комп'ютер розробника середовища програмування згенеровані файли з текстом програмного коду автоматично відкриваються в ньому після подвійного клацання на піктограмі цих файлів. Проте, краще копіювати вміст цих файлів в попередньо створені програмні проєкти в цих середовищах для повного контролю власне процесу програмування.

Згенеровані файли з текстом програмного коду містять мінімум інформації. Для включення додаткових елементів в програмний код слід змінити властивості генерації програмного коду, встановлені за замовчуванням. Нагадаємо, що зміна цих властивостей описана раніше при виконанні етапу «Встановлення властивостей генерації програмного коду».

Слід зазначити, що ефект від використання засобу Rational Rose проявляється при розробці масштабних проєктів. При розгляді моделі системи управління банкоматом може скластися враження про те, що написати й налагодити відповідну програму набагато простіше безпосередньо в тому чи іншому інтегрованому середовищі програмування. Однак ситуація здасться не настільки очевидною, коли необхідно виконати проєкт з декількома десятками варіантів використання і сотнею класів. Саме для подібних проєктів виявляється перевага використання засобу Rational Rose і нотації мови UML для документування та реалізації відповідної моделі.

В даний час повністю специфікована і документально підтверджена версія 2.0 мови UML і триває подальша робота по її розвитку. Хід цієї роботи та її стан відображаються на офіційному сайті консорціуму OMG **www.omg.org.** Там же містяться повні специфікації стандарту OMG-UML, надані для вільного доступу.

Іншим джерелом інформації з мови UML в Інтернеті є сайт компанії IBM Rational Software Coip. **www.rational.com,** з боку якої здійснюється загальна координація роботи над черговими версіями мови UML. Ця компанія як і раніше продовжує розробку CASE-засобу Rational Rose, в якому реалізуються поточні доповнення мови UML.

## Питання до розділу 3

3.1. Що є класом в UML?

3.2. Що таке «атрибут класу»?

3.3. Вкажіть основні властивості мови моделювання UML.

3.4. Вкажіть можливі типи відношень між класами UML.

3.5. Що визначає властивість «видимість атрибута»?

3.6. Вкажіть можливі значення видимості властивості класу.

3.7. Визначте призначення діаграми використання.

3.8. В яких випадках доцільно використовувати діаграми діяльності?

3.9. Визначте призначення діаграм послідовностей.

## ЛІТЕРАТУРА

1. Брауде Э. Дж. Технология разработки программного обеспечения. Питер, 2004.

2. Брукс Ф. Мифический человеко-месяц, или как создаются программные системы. - СПб.: Символ-Плюс, 2001.

3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: учебник. — 2-е изд., перераб. и доп. - М.: Финансы и статистика, 2005. - 544 с: ил.

4. Илес П. Что такое архитектура программного обеспечения?

Ресурс: http://www.interface.ru/home.asp?artId=2116

5. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем: учебное пособие. – СПб: Университет ИТМО, 2015. – 206 с.

6. Мінухін С. В. Методи і моделі проектування на основі сучасних CASE-засобів. Навчальний посібник / С.В. Мінухін, О.М. Беседовський, С. В. Знахур. – Харків: Вид. ХНЕУ, 2008. – 272 с. (укр. мов.)

7. Проектирование информационных систем: учеб. пособие / В.И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – М. Интернет-Ун-т Информ. технологий, 2005. – 304 с.

8. Реінжиніринг бізнес-процесів.

Ресурс: https://library.if.ua/book/28/1899.html

9. Соммервилл И. Инженерия программного обеспечения. - Вильямс, 2002.

10. Технологии проэктирования
Ресурс: https://studfile.net/preview/3997729/page:5/

11. Фаулер М. и др. Архитектура корпоративных программных приложений. - Вильямс, 2004.

12. Фаулер, К. Скотт. UML в кратком изложении. - М.: Мир, 1999.

# ДОДАТКИ

## Додаток А

### Приклад моделювання програмного продукту «Системи автоматизації товарообігу аптек» на основі UML

При моделюванні програмного продукту додатку було використано мову графічного опису UML (Unified Modeling Language), зручну для системного проєктування та відображення організаційних структур. Вона є уніфікованою мовою об’єктно-орієнтованого моделювання, що використовується у парадигмі ООП (об’єктно-орієнтованого програмування).

UML-модель працює за допомогою графічних позначень стандарту мови широкого профілю UML. UML-модель призначена для візуалізації, проєктування та документування програмних систем. UML не є мовою програмування, але використовуючи UML-моделі можна згенерувати код.

Моделювання на основі UML – це складний процес, який починається з концептуальної схеми, а також з роботи з логічної та фізичними моделями.

Модель базується на використанні діаграми (use-case diagram), яка описує призначення системи. При проєктуванні діаграма варіантів є вихідною концептуальною моделлю системи в процесі та в період її розробки.

**Функціональне моделювання функціоналу.** Для графічного опису систем та процесів діяльності проєктів використовується методологія IDEF0 (для опису множини функцій). Вона дозволяє розуміти функції організації, при цьому не об’єднувати їх з об'єктами, що мають вплив при реалізації. Завдяки стандартизованому IDEF0 існує можливість: 1) бачити об'єкти, при цьому потоки (інформаційні та матеріальні) перетворюються в бізнес-процес; 2) керувати об'єктами в потоках (матеріальних та інформаційних), які перетворюються на процеси, що потрібні для виконання їх функцій. Система IDEF0 демонструє інструменти та ресурси для функціонування бізнес-процесів в проєкті.

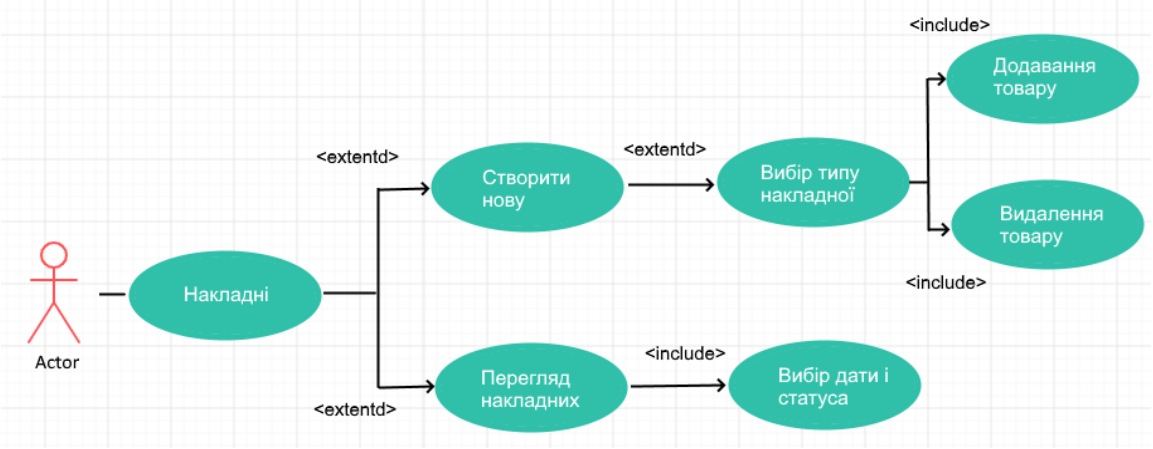



Рис. А.1. Діаграма варіантів використання роботи з накладними

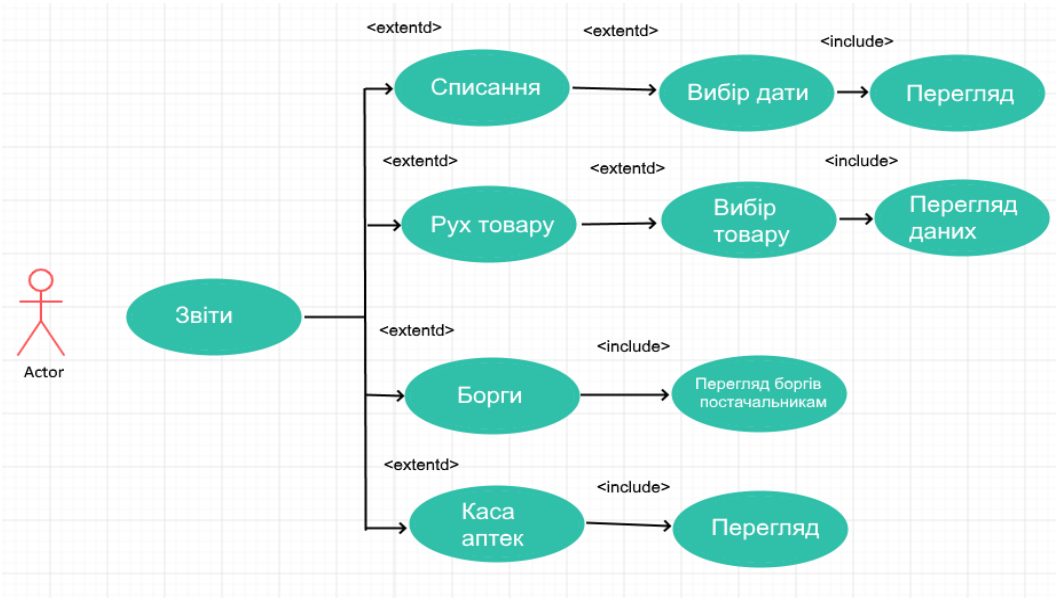



Рис. А.2. Діаграма варіантів використання роботи зі звітами


Рис. А.3. IDEF0

**Функціональне моделювання IDEF1**. Механізм IDEF1 використовують при побудові структурної інформаційної моделі, необхідної для підтримки роботи системи (рис. А.4).

Основною метою моделювання є процес вивчення та аналіз структури, способів взаємозв'язків інформації організації, виявлення потреб при керуванні та при роботі. Основними складовими інформаційної моделі є діаграми та словник: діаграми – це структурні зображення інформаційної моделі, що подають відповідно до набору правил склад та логічні зв'язки, які використовують дані; словник - це текстові й табличні фрагменти, що описують значення кожного елемента моделі.

Базові елементи методології IDEF1 наступні:

1. *Сутність змісту* (тобто визначення реального або віртуального об'єкта з набором властивостей чи атрибутів). Головними властивостями є стійкість та унікальність. Характерні властивості для атрибутів є ознаки об'єктів реального світу, які належать певній сутності.

2. *Відношення* (тобто взаємозв’язки екземлярів із сутностями інших класів) демонструє взаємовідносини між сутностями моделі.

**Структурне моделювання**. Для опису поведінки системи використовують діаграми автоматів (рис. А.5), які реалізуються за допомогою поведінки екземпляра сутності або в рамках варіанту використання (класу, об'єкта, компонента, вузла або системи в цілому).

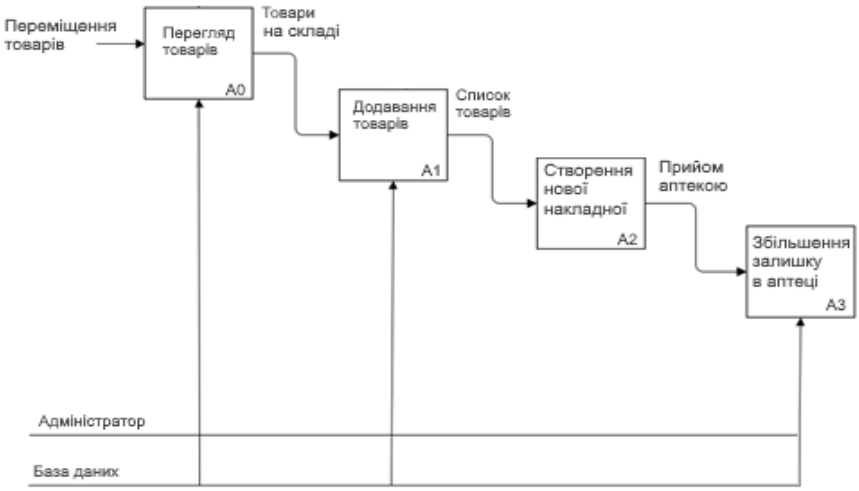



Рисунок А.4 – IDEF1


Рис. А.5. Діаграма автоматів

Дана поведінка проводить моделювання опису через стани екземплярів сутності та переходів на проміжку часу його життєвого циклу, який починається від народження та до його знищення.

**Діаграми послідовності.** Для демонстрування ілюстрацій застосовують діаграми послідовності, які відображають взаємодію об'єктів від частини використання або повного варіанту. Взаємодія об'єктів, яка втілює варіант використання, ілюструється однією або кількома діаграмами послідовності.

Зазвичай для висвітлення основного потоку подій використовується одна діаграма послідовності, для кожного наступного підпотоку – варіанти використання. При проєктуванні діаграми послідовності допомагають усвідомити ролі об'єктів в потоці, і завдяки чому надають дані для початку при реалізації інтерфейсу.

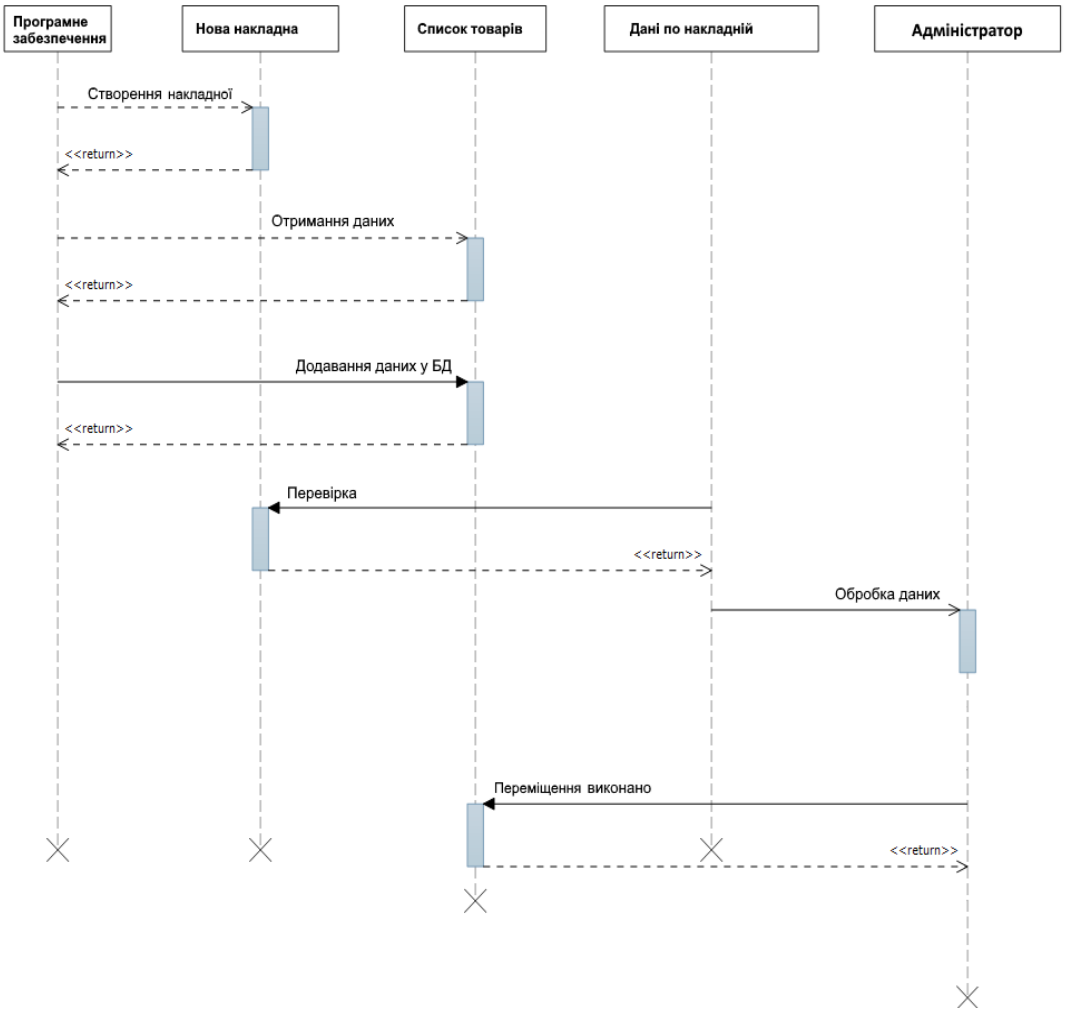



Рис. 3.6. Діаграма послідовності

**Діаграми комунікації**. Якщо аналізувати діаграми послідовності на діаграмі комунікації, то можна звернути увагу на структуру взаємодії. Користувачі процесу можуть демонструвати ненаправлені асоціації, що показують, які повідомлення були передані (крім загальних елементів). Наступною рисою є застосування нумерації в специфікації, що показує порядок виконання дій.


Рис. А.7. Діаграма комунікації

**Діаграма діяльності**. Діаграма діяльності використовується при моделюванні поведінки системи проєкту, де є потреба в деталізуванні алгоритму та логіки при реалізації операцій. Головною ціллю є застосування блок-схеми та схем алгоритмів. Кожна схема акцентує увагу на послідовність виконання деяких процесів чи операцій, які приводять користувача до бажаного результату. Під час моделювання процесу за допомогою мови UML використовують *діаграму діяльності*, що подається у вигляді графа, вершинами відповідно є стани, дуги – це переходи від одного до іншого стану. Роботою з використанням діяльності є демонстрація, візуалізація, показ особливостей реалізації операцій класів, коли є потреба зобразити роботу алгоритма та хід його процесу. При цьому кожна дія виконується операціями деякого класу.

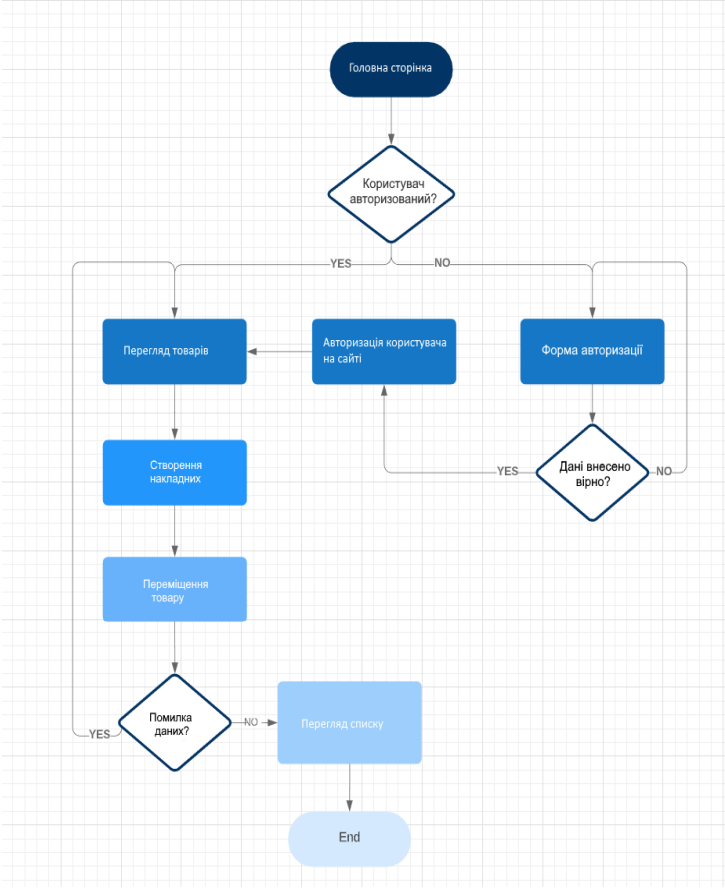



Рис. А.8. Діаграма діяльності

**Діаграма розгортання**. Діаграма, на якій демонструються обчислювальні вузли називається діаграмою розгортання. Вона відображає робочі екземпляри компонентів, а *діаграма компонентів*, натомість, відображає зв'язки між типами компонентів.

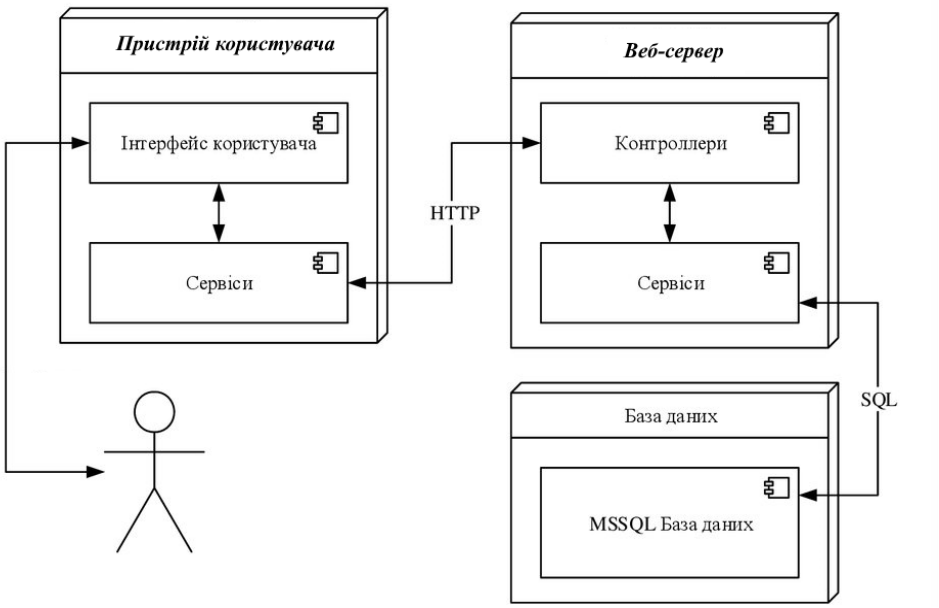



Рис. А.9. Діаграма розгортання

**Прототип дизайну.** Під час розробки веб-сторінки застосовано принципи дизайну UI/UX на основі аналізу користувацького досвіду та поведінки.

Метою проєкту було створити інтерфейс для відвідувача сайту, який дозволить в співвідношенні затраченого часу та поставлених питань вирішити завдання легко.

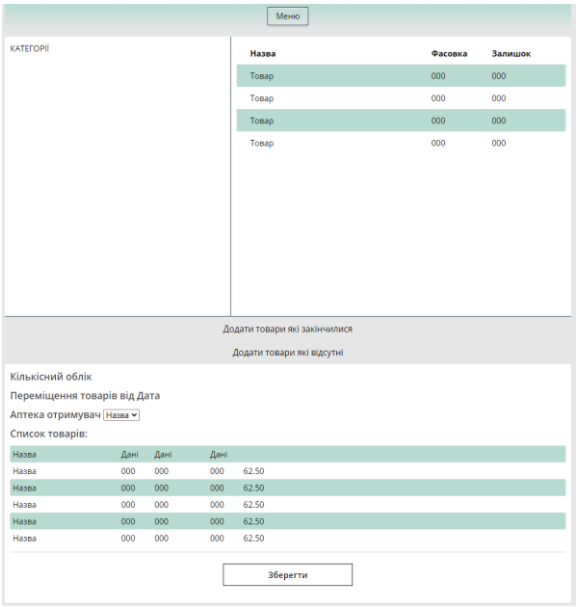



Рис. А.10. Прототип сторінки переміщення товару

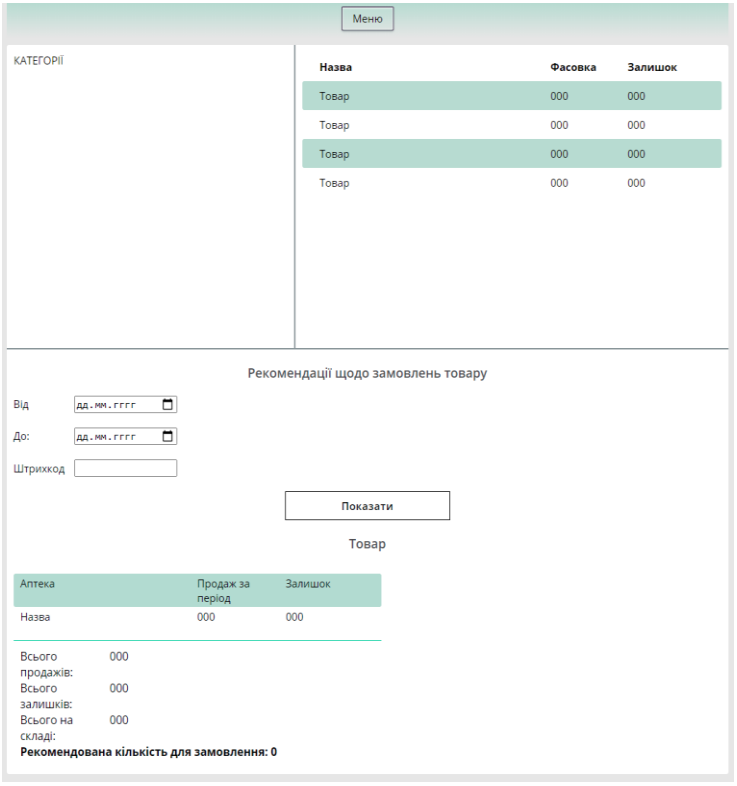



Рис. А.11. Прототип сторінки рекомендацій що до замовлення товарів

**Розробка структури даних**. Одним з головних елементів більшості баз даних мого типу, незалежно від її представлення, це структура даних, що зображує певне представлення даних та інформації щоб розуміти методи її зберігання, пошуку та обробки. Структури вхідних даних потребують обробки цих даних, особливо тих що не пов'язані з обчислювальною структурою бази даних. Вхідна інформація виступає одиницею інформації структури даних, що дозволяє зберігати її а також логічно пов'язані з нею дані.

Модель з даними об'єкта БД дає можливість розглядати її як сукупність об'єктів кожного окремого запису. Всі системи бази даних мають можливість реалізувати власну модель даних, яка описує правила для генерації певної структур даних, необхідних для системи, а також можливі дії з цими структурами. Через це модель бази даних встановлює певні обмеження для деяких конкретних структур даних, які ця система використовує або створює.

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нет* |
| ☐ | 2 | **datetime** | datetime | | | Да | *NULL* |
| ☐ | 3 | **cash** | decimal(10,2) | | | Да | *NULL* |
| ☐ | 4 | **present_cash** | decimal(10,2) | | | Да | *NULL* |
| ☐ | 5 | **credit_checks_sum** | decimal(10,2) | | | Да | *NULL* |
| ☐ | 6 | **sum_difference** | decimal(10,2) | | | Да | *NULL* |
| ☐ | 7 | **status** | tinyint(1) | | | Нет | 0 |
| ☐ | 8 | **store_id** | int(11) | | | Нет | *Нет* |

Рис. А.12. Розроблена таблиця бази даних audits

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нем* |
| ☐ | 2 | **product_barcode** | bigint(13) | | | Нет | *Нем* |
| ☐ | 3 | **quantity_exist** | decimal(10,2) | | | Нет | *Нем* |
| ☐ | 4 | **quantity_verified** | decimal(10,2) | | | Нет | *Нем* |
| ☐ | 5 | **audit_id** | int(11) | | | Нет | *Нем* |

Рис. А.13. Розроблена таблиця бази даних audits_list

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нем* |
| ☐ | 2 | **cash_operations_id** | int(11) | | | Нет | *Нем* |
| ☐ | 3 | **encashment_sum** | decimal(10,2) | | | Нет | *Нем* |
| ☐ | 4 | **status** | tinyint(1) | | | Нет | 0 |
| ☐ | 5 | **user_id** | smallint(6) | | | Да | *NULL* |

Рис. А.14. Розроблена таблиця бази даних cash_collection

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нем* |
| ☐ | 2 | **current_sum** | decimal(10,2) | | | Нет | 0.00 |
| ☐ | 3 | **store_id** | smallint(6) | | | Нет | *Нем* |

Рис. А.15. Розроблена таблиця бази даних cash_current_amount

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нем* |
| ☐ | 2 | **datetime** | datetime | | | Нет | *Нем* |
| ☐ | 3 | **type** | tinyint(1) | | | Нет | *Нем* |
| ☐ | 4 | **sum** | decimal(10,2) | | | Нет | *Нем* |
| ☐ | 5 | **operation_description** | varchar(500) | utf8_general_ci | | Нет | *Нем* |
| ☐ | 6 | **store_id** | smallint(6) | | | Нет | *Нем* |
| ☐ | 7 | **user_id** | smallint(6) | | | Нет | *Нем* |

Рис. А.16. Розроблена таблиця бази даних cash_operations

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нем* |
| ☐ | 2 | **name** | varchar(255) | utf8_general_ci | | Нет | *Нем* |
| ☐ | 3 | **parent_id** | int(11) | | | Нет | *Нем* |

Рис. А.17. Розроблена таблиця бази даних category

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(10) | | UNSIGNED | Нет | *Нем* |
| ☐ | 2 | **incoming_date** | date | | | Нет | *Нем* |
| ☐ | 3 | **payment_date** | date | | | Да | *NULL* |
| ☐ | 4 | **total_sum** | decimal(8,2) | | | Нет | *Нем* |
| ☐ | 5 | **status_id** | tinyint(1) | | | Нет | *Нем* |
| ☐ | 6 | **supplier_id** | int(11) | | | Нет | *Нем* |
| ☐ | 7 | **last_update** | date | | | Да | *NULL* |
| ☐ | 8 | **update_qty** | tinyint(3) | | UNSIGNED | Да | *NULL* |

Рис. А.18. Розроблена таблиця бази даних incoming_invoice

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(10) | | UNSIGNED | Нет | *Нет* |
| ☐ | 2 | **quantity** | decimal(10,3) | | UNSIGNED | Нет | *Нет* |
| ☐ | 3 | **price** | decimal(5,2) | | UNSIGNED | Нет | *Нет* |
| ☐ | 4 | **incoming_invoice_id** | int(10) | | UNSIGNED | Нет | *Нет* |
| ☐ | 5 | **product_barcode** | bigint(20) | | UNSIGNED | Нет | *Нет* |

Рис. А.19. Розроблена таблиця бази даних incoming_invoice

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | tinyint(1) | | UNSIGNED | Нет | *Нет* |
| ☐ | 2 | **name** | varchar(45) | utf8_general_ci | | Нет | *Нет* |

Рис. А.20. Розроблена таблиця бази даних invoice_status

| | # | Имя | Тип | Сравнение | Атрибуты | Null | По умолчанию |
|---|---|---|---|---|---|---|---|
| ☐ | 1 | **id** | int(11) | | | Нет | *Нет* |
| ☐ | 2 | **active** | tinyint(1) | | | Нет | 1 |
| ☐ | 3 | **login** | varchar(255) | utf8_general_ci | | Нет | *Нет* |
| ☐ | 4 | **password** | varchar(500) | utf8_general_ci | | Нет | *Нет* |
| ☐ | 5 | **name** | varchar(255) | utf8_general_ci | | Нет | *Нет* |

Рис. А.21. Розроблена таблиця бази даних users_wh3.4. Розрахунок економічного ефекту