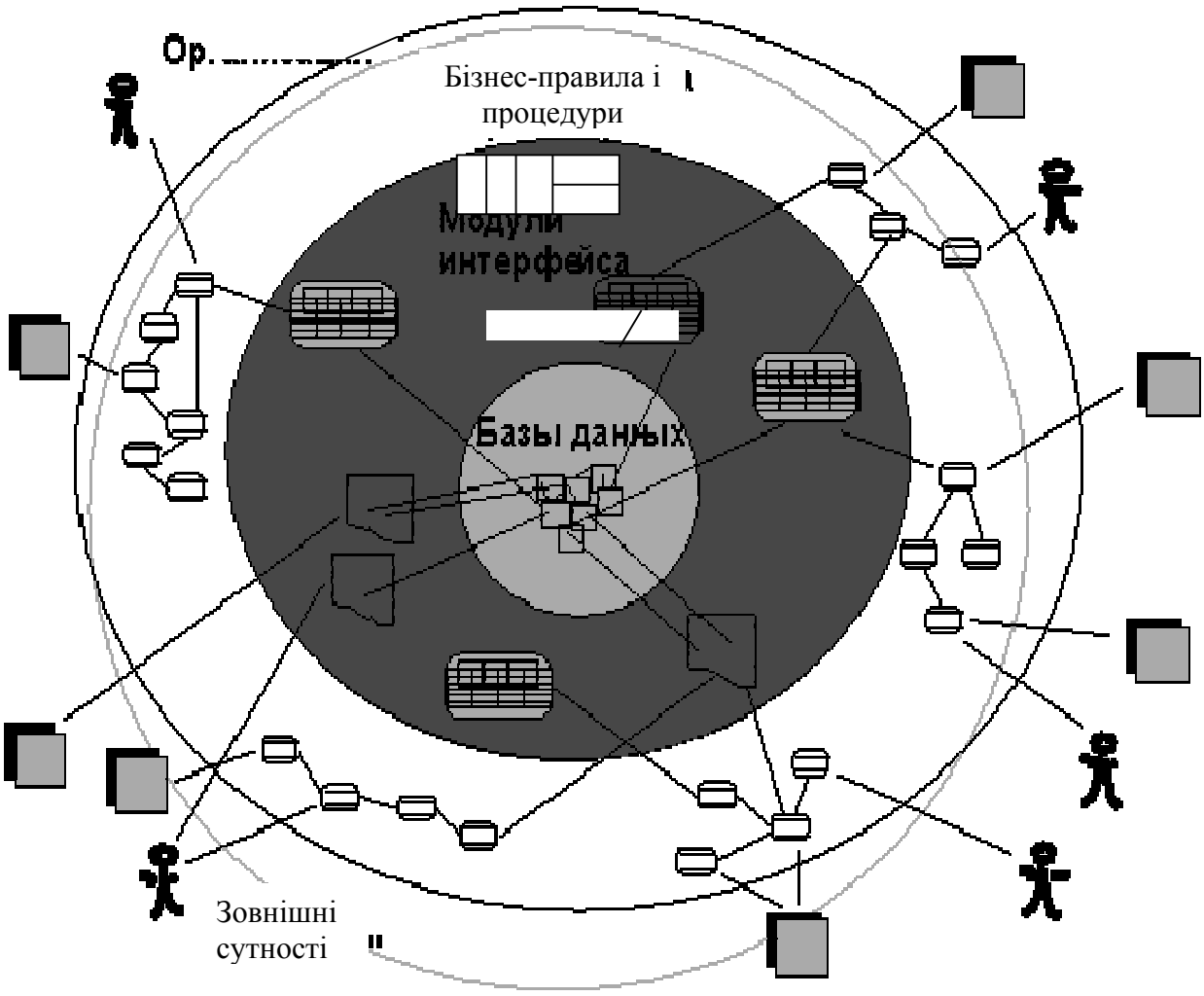
І.М.Пістунов

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ



Міністерство освіти і науки України
Національний гірничий університет

І.М. Пістунов

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Навчальний посібник

Дніпропетровськ
НГУ
2008

УДК 681.518.001.063(075.8)
БКК 32.973.2:30.2я73
П34

Затверджено вченою радою університету як навчальний посібник по дисципліні "Проектування інформаційних систем" для студентів очної та заочної форм навчання зі спеціальності 7.050102 Економічна кібернетика (Протокол № 6 від 9.10.2007 р).

Рецензенти:
О.І. Михальов, д-р техн. наук, проф., завідувач кафедри інформаційних технологій і систем Дніпропетровської національної металургійної академії України;
Т.М. Пашова, канд. техн. наук, доц., завідувач кафедри інформаційних систем і технологій Дніпропетровського державного аграрного університету.

Пістунов І.М.
П34 Проектування інформаційних систем: Навч. посібник – Д.: Національний гірничий університет, 2008. – 71 с.

Розглянуто теоретичні і практичні аспекти проектування інформаційних систем, прийоми формалізації опису інформаційних потоків, використання CASE-методу автоматичного проектування.

Посібник містить приклади вирішення задач із застосуванням пакету Open Office вільного програмного забезпечення.

В посібнику подано завдання для самостійного вирішення, тому він може слугувати і як посібник для практичних чи лабораторних занять із застосуванням комп'ютерної техніки.

Призначений для студентів вищих учбових закладів і може бути корисним для економістів, плановиків, менеджерів та маркетологів.

Посібник базується на літературних джерелах вітчизняних та зарубіжних авторів та на досвіді викладання дисципліни „Проектування інформаційних систем" в Національному гірничому університеті.

УДК 681.518.001.063(075.8)
БКК 32.973.2:30.2я73

ЗМІСТ

Розділи Стор.

ВСТУП

Тенденції розвитку сучасних інформаційних технологій приводять до постійного зростання складності інформаційних систем (ІС), створюваних в різних областях економіки. Сучасні крупні проекти ІС характеризуються, як правило, наступними особливостями:

– складність опису (достатньо велика кількість функцій, процесів, елементів даних і складні взаємозв'язки між ними), які вимагають ретельного моделювання і аналізу даних і процесів;

– наявність сукупності тісно взаємодіючих компонентів (підсистем), що мають свої локальні задачі і цілі функціонування (наприклад, традиційних додатків, пов'язаних з обробкою трансакцій і рішенням регламентних задач, і додатків аналітичної обробки (підтримка ухвалення рішень), які використовують нерегламентовані запити до даних великого об'єму);

– відсутність прямих аналогів, обмежуюче можливість використовування будь-яких типових проектних рішень і прикладних систем;

– необхідність інтеграції до існуючих додатків і до таких, що знов розробляються;

– функціонування в неоднорідному середовищі на декількох апаратних платформах;

– роз'єднаність і різнорідність окремих груп розробників за рівнем кваліфікації і традиціям використовування тих або інших інструментальних засобів;

– істотна тимчасова протяжність проекту, обумовлена, з одного боку, обмеженими можливостями колективу розробників, і, з другого боку, масштабами організації-замовника і різним ступенем готовності окремих її підрозділів до упровадження ІС.

Для успішної реалізації проекту об'єкт проектування (ІС) повинен бути перш за все адекватно описаний, повинні бути побудовані повні і несуперечливі функціональні і інформаційні моделі ІС.

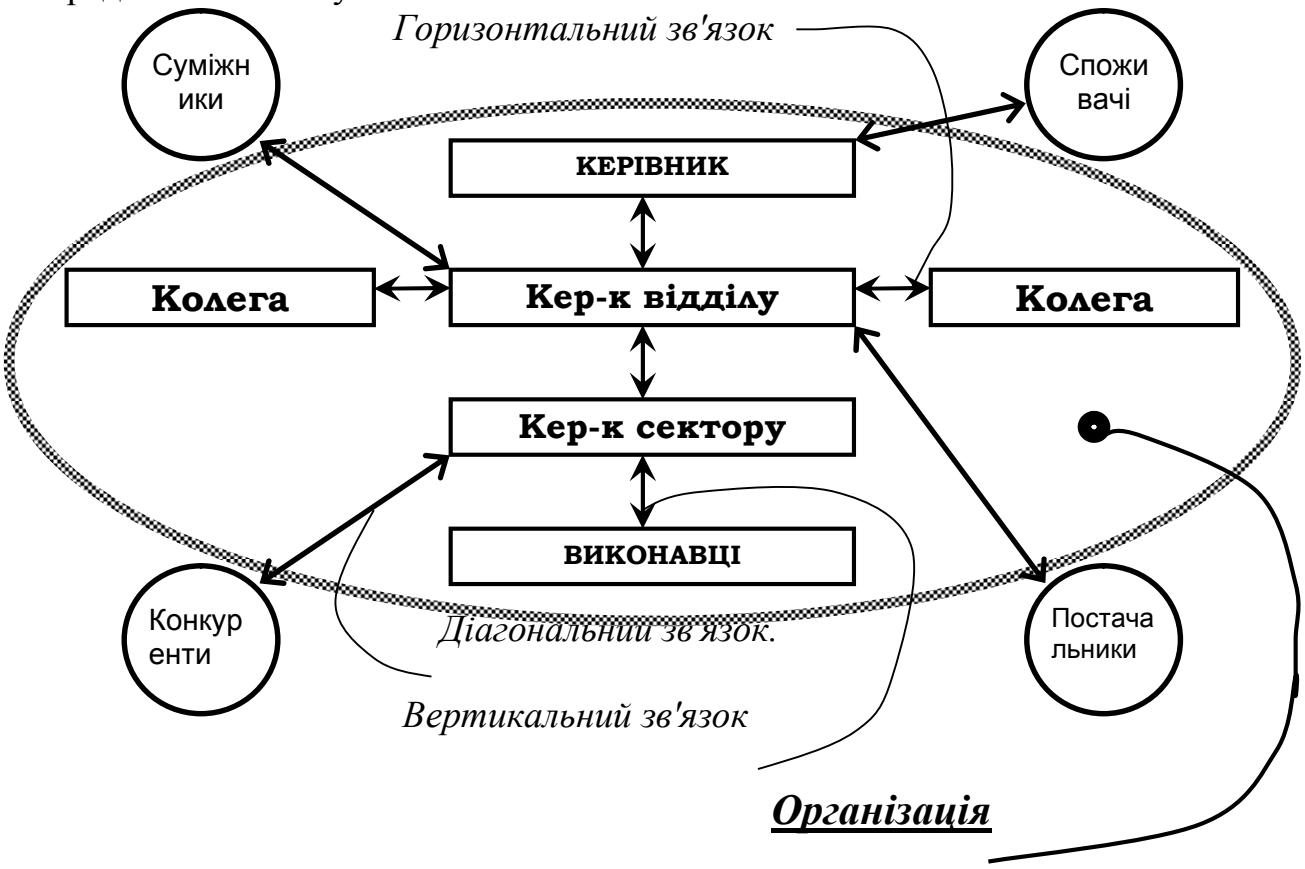
Раніше при розробці ІС достатньо широко застосовувалася структурна методологія, що надає в розпорядження розробників суворо формалізовані методи опису ІС і технічних рішень. Вона заснована на наочній графічній техніці: для опису різного роду моделей ІС використовуються схеми і діаграми. Наочність і точність засобів структурного аналізу дозволяла розробникам і майбутнім користувачам системи із самого початку неформально брати участь в її створенні, обговорювати і закріплювати розуміння основних технічних рішень. Проте, широке застосування цієї методології і її впровадження при розробці конкретних ІС зустрічалося достатньо рідко, оскільки при неавтоматизованій (ручний) розробці це практично неможливо.

Перераховані чинники сприяли появі програмно-технологічних засобів спеціального класу - CASE-засобів, що реалізовують CASE-технологію створення і супроводу ІС. Термін CASE (Computer Aided Software Engineering)

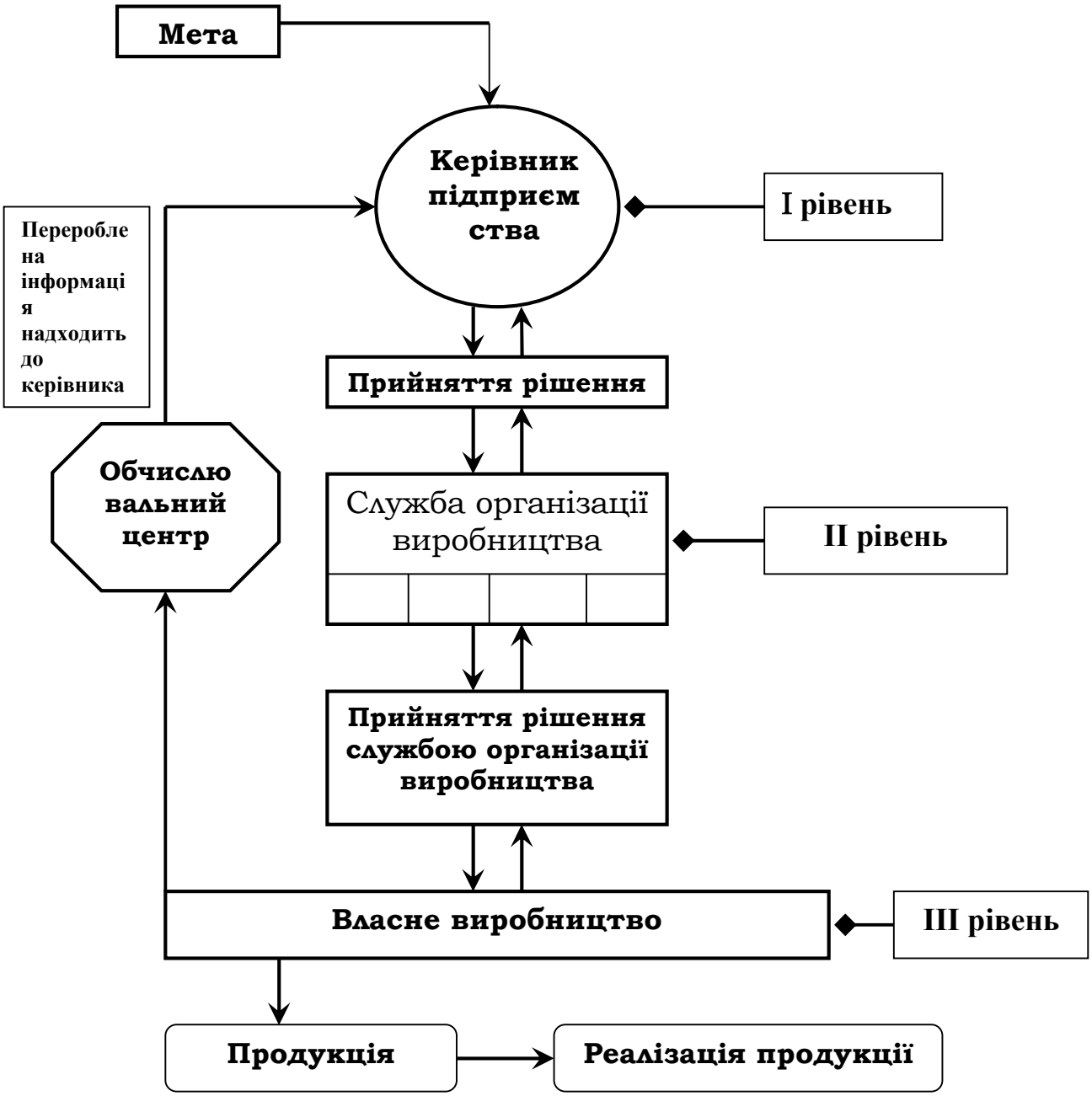
використовується в даний час у вельми широкому значенні. Первинне значення терміну CASE, обмежене питаннями автоматизації розробки тільки програмного забезпечення (ПО), на сьогодні набуло нового значення, що охоплює процес розробки складних ІС в цілому. Тепер під терміном CASE-засобу маються на увазі програмні засоби, що підтримують процеси створення і супроводу ІС, включаючи аналіз і формулювання вимог, проектування прикладного ПО (додатків) і баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління і управління проектом, а також інші процеси. CASE-засоби разом з системним ПО і технічними засобами утворюють повне середовище розробки ІС.

CASE-технологія є методологією проектування ІС, а також набором інструментальних засобів, що дозволяють в наочній формі моделювати наочну область, аналізувати модель на всіх етапах розробки і супроводу ІС і розробляти додатки відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів заснована на методологіях структурного (в основному) або об'єктно-орієнтованого аналізу і проектування, що використовує специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв'язків між моделями системи, динаміки поведінки системи і архітектури програмних засобів. CASE-технологія в даний час потрапила в розряд найстабільніших інформаційних технологій (її використовувала половина всіх опитаних користувачів більш ніж в третині своїх проектів, з них 85% завершилися успішно).

Інформаційні потоки в системі управління організації можуть бути представлені наступними зв’язками:



При створенні інформаційної системи для управління виробництвом, треба врахувати основні напрямки інформаційних потоків:



В ІС можна виділити дві основні підсистеми:

- Функціональну, що виконує функції, пов'язані з управлінням виробництва;
- Забезпечення, що обслуговує власні процеси (математичне і програмне забезпечення).

1. ОСНОВНІ ПРИНЦИПИ СТВОРЕННЯ ПРОЕКТУ ІНФОРМАЦІЙНИХ СИСТЕМ

З матеріалів цього розділу студент зрозуміє основи класифікації та проектування інформаційних систем.

1.1. Класифікація ІС

Системи інформаційного забезпечення мають самостійне цільове призначення і область застосування, вони входять до складу будь-якої автоматизованої системи управління. Вони є найважливішими компонентами систем автоматизованого проектування, автоматичних систем наукового дослідження, ІС. ІС, що мають самостійне призначення:

– інформаційні пошукові системи,
– інформаційно - довідкові системи.

Вони можуть бути поділені на три класи систем:

1 Інтелектуальні-діалогові (питання/відповідь);
2 Розрахунково-логічні(системи ухвалення рішення);
3 Експертні системи.

Системи ухвалення рішення – це системи які використовує програма, для реалізації моделі ухвалення рішення в конкретних задачах, що виникають у людей в ній професійній діяльності. Суть задачі полягає у виборі деякої підмножини з безлічі альтернатив або в їх впорядковуванні. Інтелектуальний діалог призначений для пошуку методів рішення інтелектуальних задач із застосуванням нових інформаційних технологій використовування бази даних і бази знань.

Експертна система – система здатна замінити експерта при рішенні деяких задач. Від рівня автоматизації, системи розділяються на інформаційні, інформаційні, що радять керівникові, самоналагоджувальні системи управління. Інформаційна система включає всю необхідну інформацію для вироблення рішень не торкаючись самої істоти рішень, тобто після аналізу рішення ухвалює людина. Інформаційна система, що радить, представляє інформацію для ухвалення рішення що містить елементи оцінки рішень, остаточне рішення приймає людина. Управляюча система на підставі початкової інформації і вироблених рішень здійснює за заданими програмами вплив на виробничий процес з метою приведення його до заданого стану. Самоналагоджувальна система може в межах розробленого алгоритму змінити програму при ситуаціях не відповідних заданій програмі вироблених рішень.

1.2. Поняття економічної інформаційної системи (ЕІС)

Інформація про процеси виробництва, розподілу, обміну і споживання матеріальних благ називається економічною інформацією, а управління і обробка економічної інформації називається економічною інформаційною системою (ЕІС). Для економічної інформації характерні прості алгоритми, переважання логічних операцій над арифметичними, табличне представлення вхідних і вихідних даних.

Найважливішою ознакою класифікації ЕІС є:

1 Відношення до даної управляючої системи, це дозволяє розділити повідомлення на вхідні, внутрішні, вихідні.

2 Ознака часу, повідомлення діляться на перспективні (планова і прогнозована інформація), ретроспективні (облікова інформація).

3 За часом надходження: періодичні, запитальні.

4 Функціональна ознака, по функціональних підсистемах об'єкту:

– інформація про трудові, матеріальні, фінансові ресурси, про виробничі процеси.

– ознака стабільності: постійна (дані ніколи не міняються), умовно постійна, змінна.

Для ЕІС характерно, що коефіцієнт стабільності, який визначається відношенням кількості інформації, що не змінилася до загальної кількості інформації, близький до 0.88. Це означає, що в ЕІС інформація умовно постійна.

1.3. Структура економічної інформації

Структурою економічної інформації визначається її будова, виділення тих або інших документів. Такі елементи називаються інформаційними одиницями (ІО). З простих ІО утворюються складні – складові. Найменшої неподільної ІО є реквізит – атрибут. Реквізити можна розділити на дві групи: підстави і ознаки. Підстава характеризує кількісні властивості сутності, одержані в результаті видачі підрахунку натуральних одиниць, зважування, вимірювання, обчислення. Ознаки виражають як правило якісні властивості сутності і характеризують обставини при яких були одержані реквізити - підстави. Крупнішою ІО за реквізити є показник. Показники утворюються з однієї підстави і реквізитів ознак, що відносяться до нього. Ще крупнішими показниками є масиви і потоки. Масив є набором показників і реквізитів об'єднаних за ознакою однорідності. Сукупність масивів відносяться до однієї функції управління називається потоком. Сукупність потоків, які характеризують управлінську роботу в цілому називають інформаційною системою об'єкту управління.

1.4. Вимоги до методології проектування ЕІС

В процесі декомпозиції компонентів ЕІС виділяють: функціональні і частини, що забезпечують. Функціональні – ряд підсистем які залежить від особливостей тієї або іншої ЕІС. Ці підсистеми розділяються по певній ознаці (функціональному або структурному) і об'єднують в собі відповідні комплекси задач управління. Частина, що забезпечує ЕІС складається із: інформаційного, програмного, математичного, технічного, правового, лінгвістичного, ергономічного і метрологічних частин. До складу інформаційного забезпечення входить позамашине і машинне забезпечення.

Позамашинне забезпечення складається із: класифікатори техніко-економічної інформації, нормативно довідкова інформація, методичні матеріали організації і використовування перерахованих компонентів. Машинне інформаційне забезпечення – інформаційна база і система управління базами даних (СУБД), програмне забезпечення – сукупність програм, які реалізують цілі, і задачі ЕІС) До складу програмних засобів: загальносистемні, прикладне забезпечення, інструктивно – методичні матеріали п застосуванню засобів програмного забезпечення. Математичне забезпечення включає: сукупність методів рішення задач управління, моделей, алгоритмів обробки інформації. Технічне забезпечення включає весь комплекс технічних засобів, які забезпечують роботу системи тобто технічні засоби збору, реєстрації, передачі, обробки, відображення, розмноження інформації. Організаційно - методичне забезпечення представляє сукупність документів визначаючих організаційну структуру документа і систем автоматизації для виконання функцій, що конкретно автоматизуються. Правове забезпечення включає систему нормативно – правових документів які повинні чітко визначати права і обов'язки фахівців в умовах функціонування ЕІС, а також комплекс документів регламентуючих порядок зберігання і захисту інформації, правил ревізії даних, забезпечення юридичної достовірності виконаних операцій. Лінгвістичне забезпечення представляє сукупність мов засобів для формалізації природної мови. Ергономічне забезпечення сукупність методів і засобів для створення оптимальних умов діяльності людини при розробки ЕІС. Метрологічне забезпечення – метрологічні засоби і інструкції по їх застосуванню.

3 методи: індивідуальний (оригінальний), типове проектування, автоматизоване проект(САПР).

Індивідуальне характеризується – всі види робіт для різних об'єктів виконуються за індивідуальними проектами. В процесі індивідуального проектування застосовуються свої оригінальні методики і засоби проведення робіт. Методики проведення робіт на всіх етапах обстеження, формування технічного завдання, розробки технічного проекту і рабинь документації створюються для конкретного об'єкту в міру необхідності. “Недоліки” висока трудомісткість, великі терміни проектування, погана доступність модернізації, погана можливість супроводу програм..

Типове проектування – розбиття системи на безліч складових компонентів і створення для кожної з них закінченого проектного рішення, яке про упровадженні прив'язується до конкретних умов об'єкту. Залежно від

декомпозиції розрізняють: елементне проектування, підсистемне проектування, об'єктне проектування. При елементному методі проектування, вся система розбивається на кінцеву безліч елементів, кожний з яких є типовим. Як елементи можуть виступати проектні рішення по інформаційному, технічному, програмному виду забезпечення.

Підсистемний метод проектування характеризується вищим ступенем інтеграції елементів ЕІС. Декомпозиція системи здійснюється на рівні функціональних підсистем, іноді комплексу задач, кожна з виділених підсистем представляється в закінченому виді пакету прикладних програм (ППП). При цьому забезпечується функціональна повнота системи, мінімум інформаційного зв'язку, можливість настройки параметрів. Для кожного ППП оформляється пакет документації. Об'єктне проектування - декомпозиція ЕІС не проводиться. Типовий проект створюється в цілому для деякого узагальненого об'єкту, певної групи. Автоматизоване проектування – автоматизація основних етапів створення ЕІС, починаючи від вибору складу задач і закінчуючи автоматичним отриманням проектної документації

1.5. Стадії і етапи проектування ЕІС

Передбачається 8 стадій створення ЕІС (рис. 1.1):

1 Формування вимог до ЕІС.
- 1.1 Обстеження об'єкту і обґрунтування необхідності створення ЕІС.
- 1.2 Формування вимог користувача до ЕІС.
- 1.3 Оформлення звіту про виконану роботу і заявки на розробку ЕІС.

2 Розробка концепції ЕІС.
- 2.1 Вивчення об'єкту.
- 2.2 Проведення необхідних науково-дослідних робіт.
- 2.3 Розробка варіантів концепції ЕІС і вибір варіанту концепції ЕІС, задовольняючого вимогам користувача.
- 2.4 Оформлення звіту про виконану роботу.

3 Технічне завдання.
- 3.1 Розробка і затвердження технічного завдання ЕІС.

4 Ескізний проект.
- 4.1 Розробка попередній рішень по вибраному варіанту ЕІС.
- 4.2 Розробка документації на ЕІС і її частин.

5 Технічний проект.
- 5.1 Розробка проектних рішень за системою і її частинам.
- 5.2 Розробка документації на ЕІС.
- 5.3 Розробка і оформлення документації на поставку виробів для комплектування ЕІС.

Рис. 1.1. Схема стадій побудови проекту ЕІС

6 Робоча документація (РД).

6.1 Розробка РД на систему або її частин.

6.2 Розробка або адаптація програм.

7 Введення в дію.

7.1 Підготовка об'єкту автоматизації до введення в дію.

7.2 Підготовка персоналу, проводиться навчання персоналу.

7.3 Будівельно-монтажні роботи, в тому випадку, якщо будується спеціалізована будівля.

7.4 Проведення попередніх випробувань. Проведення досвідченої експлуатації.

7.5 Проведення досвідчених випробувань.

7.6 Введення в промислову експлуатацію.

8 Супровід ЕІС.

8.1 Виконання робіт відповідно до гарантійних зобов'язань.

8.2 Після гарантійне обслуговування.

1.6. Технологічні операції проектування

Технологічні операції проектування з використанням стандартів проектування та в умовних позначеннях цих стандартів мають вигляд:

Перший рівень:

П1- вибір вихідних документів.

Д1 - техніко-економічне обстеження.

Д2 опис систем запитів.

Д3 вихідні документи ТПР

Д{3} форми вихідних документів без доробки.

Д4 форми з вихідних масивів.

Д5 опис до програми для отримання вихідних форм

Д6 вихідні форми, що генеруються за запитом.

Д7 опис балка зв'язків реквізитів у вихідних формах

Д8 інструкції по використовуванню вихідних форм

Д9 періодичність отримання вихідних форм

Деталізація першого рівня

Д1Р1Д10 – П2 – Д{10}Д11-14:

П2 - вибір вихідних документів.

Р1 - обмеження на форми вихідних документів.

Д10 форми вхідних документів.

Д{10} форми вхідних документів без доробки

Д11 обмеження на вхідні форми.

Д12 зв'язки реквізитів.

Д13 інструкції по заповненню.

Д14 періодичність надходження.

Д1-2Д{3}Д{10}Д15 – П3 – Д{15}Д16-18:

П3 - вибір модулів ТПР і розробка схем інформаційної ув'язки ТПР і оригінальних рішень.

Д15 - блок схема рішення задачі.

Д{15} вибрані з ТПР модулі рішення задач без доробки.

Д16 оригінальні модулі.

Д17 схеми інформаційної ув'язки оригінальних модулів і ТПР

Д18 постановка задачі

Д1Д{3}Д4Д6-7 – П4 – U1 Д{19}Д19-20Д27:

П4 - вибір системи класифікації.

U1 загальнодержавні класифікатори.

Д19 довідники класифікатори, що застосовуються в ТПР.

Д{19} вибрані класифікатори, пропоновані в ТПР

Д20 локальні класифікатори.

Д27 довідник класифікаторів на підприємстві

Д2Д{3}Д4-5Д21 - П5 – Д{21}Д22-23:

П5 - вибір типових програмних модулів.

Д{21} типові програмні модулі без доробки.

Д22 ---- ті, що вимагають доробку.

Д23 контрольний приклад.

Р2U2Д24-25U3 – П6 – Д{24}Д26:

П6 вибір комплексу технічних засобів.
Р2 комплекс параметрів інформаційні потоки, що описують, на об'єкті
U2 перелік наявних технічних засобів.
Д24 типові рекомендації по вибору і розміщенню технічних засобів.
Д25 обмеження на використовування тих засобів.
U3 номенклатура технічних засобів.
Д{24} вибрані типові рекомендації.
Д26 перелік вибраних засобів, які є.

Після закінчення проектування, потрібно розрахувати основні показники ефективності ЕІС:

річний економічний ефект $EE_p = РПП - E_H K$

термін окупності $T = \frac{1}{E_P}$

де $E_P = \frac{РПП}{K}$ – розрахунковий коефіцієнт ефективності витрат на створення ЕІС, $РПП = \frac{(A_2 - A_1)П_1}{A_1} + \frac{(C_1 - C_2)A_2}{100}$ – річний приріст прибутку, A_1, A_2 – річний об'єм реалізованої продукції до і після упровадження ЕІС, C_1 C_2 – витрати на 1 гривню реалізованої продукції до і після упровадження ЕІС, $П_1$ прибуток від реалізації продукції до упровадження ЕІС, Eн – нормативний коефіцієнт ефективності капітальних вкладень на упровадження ЕІС, K – розмір капіталовкладень у проект.

Приклад

Написати перелік технологічних операції процесу створення проекту для задачі “Облік матеріальних цінностей”.

Д1-4 – П1 – Д5:
П1 формування першої позиції параметричного потоку.
Д1 документація ППП.
Д2 обмеження на форми вхідних документів.
Д3 система бухобліку на об'єкті.
Д4 перелік задач.
Д5 перелік вхідних документів
Д5V1 – П2 – Д6;
П2 опис вхідних документів.
V1 каталог реквізитів (найменування, значність).
Д6 опис форм вхідних документів
Д1Д3Д4Д7 – П3 – Д8:
П3 визначення переліку вихідних документів.
Д7 обмеження за формою вихідних документів.
Д8 перелік форм вихідних документів
Д8V1 – П4 - Д9:
П4 опис вихідних документів.
Д9 формалізований опис вихідних документів
Д10-11 – П5 – Д12V2:

П5 складання каталогу реквізитів системи.
Д10 документація техніко-економічного обстеження об'єкту управління.
Д11 правило ідентифікації реквізитів.
Д12 документація користувачу.
V2 – каталог реквізитів системи
Д10V3 - П6 – Д{3}:
П6 вибір системи бухобліку.
V3 класифікація систем бухобліку.
Д{3} вибрані системи бухобліку.
Д10Р1 – П7 – Д{4}:
П7 прийнятий перелік задач.
Р1 обмеження об'єкту управління.
Д{4} перелік задач підлягаючих рішенню.
Д6Д9Д12Д14 – П8 – Д15-17:
П8 ідентифікація логічних зв'язків між показниками вхідних і вихідних документів.
Д12 методика розрахунків показників.
Д14 деяка додаткова інформація.
Д15 логічні зв'язки між показниками вхідних документів.
Д16 логічні зв'язки між показниками вихідних документів.
Д17 логічні зв'язки між показниками вхідних і вихідних документів.
Р2-3 Д18 – П9 – Р4:
П9 визначення ресурсів обчислювальної системи.
Р2 вимоги до ресурсів обчислювальної техніки.
Р3 характеристики наявних ресурсів обчислювальної системи.
Д18 обмеження на терміни отримання документів після обробки.
Р4 ресурси обчислювальної системи, доступні для пакету.

Контрольні запитання

1. Які бувають типи інформаційних систем?
2. Наведіть основні відміни економічної інформаційної системи від інших.
3. В чому полягають принципи структури економічної інформації?
4. Назвіть основні стадії проектування інформаційної системи?
5. Яка система кодування застосована у стандартах для позначення етапів проектування інформаційної системи?

У розділі визначено класифікацію, поняття економічної інформаційної системи, структуру економічної інформації, вимоги до методології проектування, стадії і етапи та технологічні операції проектування.

2. ОСНОВИ МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ІС

Вивчення матеріалу цього розділу дозволить зрозуміти основні принципи порядку розробки ІС.

2.1. Життєвий цикл інформаційної системи

Одним з базових понять методології проектування ІС є поняття життєвого циклу її програмного забезпечення (ЖЦ ПЗ). ЖЦ ПЗ – це безперервний процес, який починається з моменту ухвалення рішення про необхідність його створення і закінчується у момент його повного вилучення з експлуатації.

Основним нормативним документом, що регламентує ЖЦ ПЗ, є міжнародний стандарт ISO/IEC 12207 (ISO - International Organization of Standardization - Міжнародна організація по стандартизації, IEC - International Electrotechnical Commission - Міжнародна комісія по електротехніці). Він визначає структуру ЖЦ, що містить процеси, дії і задачі, які повинні бути виконані під час створення ПЗ.

Структура ЖЦ ПЗ за стандартом ISO/IEC 12207 базується на трьох групах процесів:

– основні процеси ЖЦ ПЗ (придбання, поставка, розробка, експлуатація, супровід);

– допоміжні процеси, що забезпечують виконання основних процесів (документування, управління конфігурацією, забезпечення якості, верифікація, атестація, оцінка, аудит, рішення проблем);

– організаційні процеси (управління проектами, створення інфраструктури проекту, визначення, оцінка і поліпшення самого ЖЦ ПЗ, навчання).

Розробка включає всі роботи зі створення ПЗ і його компонент відповідно до заданих вимог, включаючи оформлення проектної і експлуатаційної документації, підготовку матеріалів, необхідних для перевірки працездатності і відповідної якості програмних продуктів, матеріалів, необхідних для організації навчання персоналу і т.д. Розробка ПЗ включає, як правило, аналіз, проектування і реалізацію (програмування).

Експлуатація включає роботи по упровадженню компонентів ПЗ в експлуатацію, зокрема конфігурацію бази даних і робочих місць користувачів, забезпечення експлуатаційною документацією, проведення навчання персоналу і т.д., і безпосередньо експлуатацію, зокрема локалізацію проблем і усунення причин їх виникнення, модифікацію ПЗ в рамках встановленого регламенту, підготовку пропозицій по вдосконаленню, розвитку і модернізації системи.

Управління проектом пов'язане з питаннями планування і організації робіт, створення колективів розробників і контролю за термінами і якістю

виконуваних робіт. Технічне і організаційне забезпечення проекту включає вибір методів і інструментальних засобів для реалізації проекту, визначення методів опису проміжних станів розробки, розробку методів і засобів випробувань ПЗ, навчання персоналу і т.п. Забезпечення якості проекту пов'язане з проблемами верифікації, перевірки і тестування ПЗ. Верифікація – це процес визначення того, чи відповідає поточний стан розробки, досягнутий на даному етапі, вимогам цього етапу. Перевірка дозволяє оцінити відповідність параметрів розробки з початковими вимогами.

2.2. Моделі життєвого циклу ПЗ

Стандарт ISO/IEC 12207 не пропонує конкретну модель ЖЦ і методи розробки ПЗ (під моделлю ЖЦ розуміється структура, що визначає послідовність виконання і взаємозв'язку процесів, дій і задач, виконуваних впродовж ЖЦ. Модель ЖЦ залежить від специфіки ІС і специфіки умов, в яких остання створюється і функціонує). Його регламенти є загальними для будь-яких моделей ЖЦ, методологій і технологій розробки. Стандарт ISO/IEC 12207 описує структуру процесів ЖЦ ПЗ, але не конкретизує в деталях, як реалізувати або виконати дії і задачі, включені в ці процеси.

До теперішнього часу найбільше поширення набули наступні дві основні моделі ЖЦ каскадна та спіральна.

Основною характеристикою каскадної моделі є розбиття всієї розробки на етапи, причому перехід з одного етапу на наступний відбувається тільки після того, як буде повністю завершена робота на поточному (рис. 2.1). Кожен етап завершується випуском повного комплекту документації, достатньої для того, щоб розробка могла бути продовжена іншою командою розробників.

Позитивні сторони застосування каскадного підходу полягають у тому що на кожному етапі формується закінчений набір проектної документації, який відповідає критеріям повноти і узгодженості. Окрім того, виконувані в логічній послідовності етапи робіт дозволяють планувати терміни завершення всіх робіт і відповідні витрати.

Каскадний підхід добре зарекомендував себе при побудові ІС, для яких на самому початку розробки можна достатньо точно і повно сформулювати всі вимоги, з тим щоб надати розробникам свободу реалізувати їх якнайкраще з технічної точки зору. У цю категорію потрапляють складні розрахункові системи, системи реального часу і інші подібні задачі. Проте, в процесі використовування цього підходу виявився ряд його недоліків, викликаних перш за все тим, що реальний процес створення ПЗ ніколи повністю не укладався в таку жорстку схему. В процесі створення ПЗ постійно виникала потреба в поверненні до попередніх етапів і уточненні або перегляді раніше ухвалених рішень. В результаті реальний процес створення ПЗ приймав наступний вигляд (рис. 2.2).

Основним недоліком каскадного підходу є істотне запізнення з отриманням результатів. Узгодження результатів з користувачами проводиться тільки в точках, запланованих після завершення кожного етапу робіт, вимоги до

ІС "заморожені" у вигляді технічного завдання на весь час її створення. Таким чином, користувачі можуть внести свої зауваження тільки після того, як робота над системою буде повністю завершена. У разі неточного викладу вимог або їх зміни протягом тривалого періоду створення ПЗ, користувачі одержують систему, що не задовольняє їх потребам.

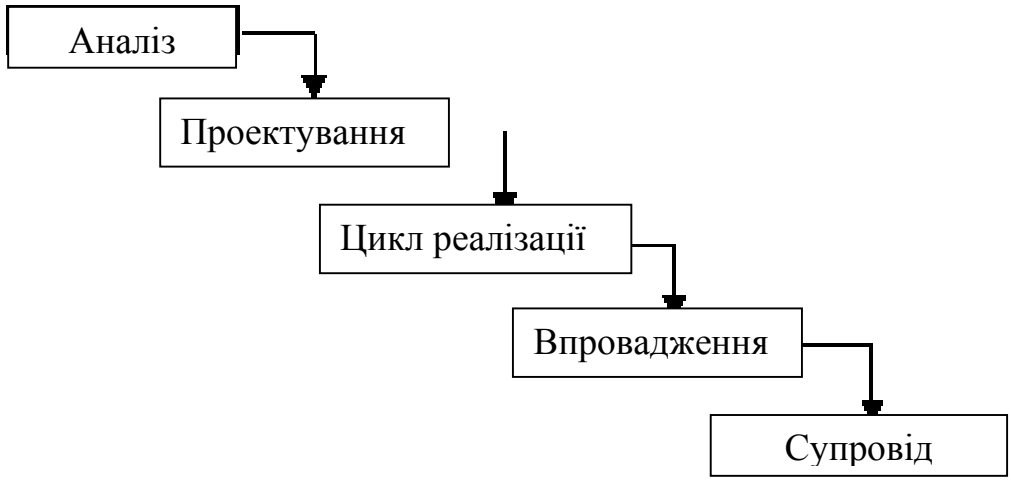


Рис. 2.1. Каскадна схема розробки ПЗ

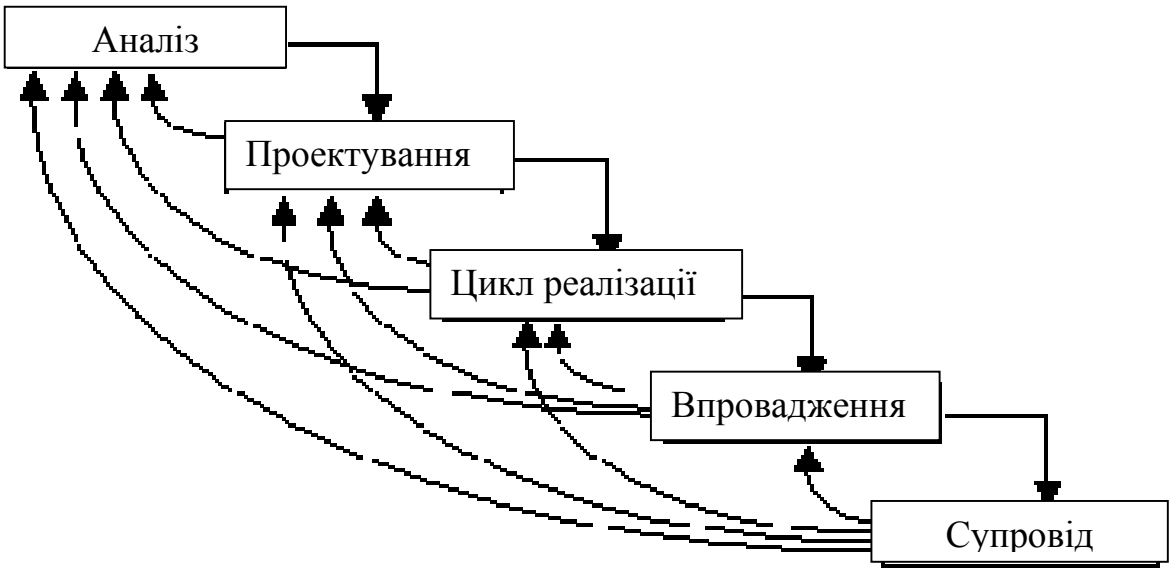


Рис. 2.2. Реальний процес розробки ПЗ за каскадною схемою

Для подолання перерахованих проблем була запропонована спіральна модель ЖЦ (рис. 2.3), яка віддає перевагу початковим етапам ЖЦ: аналізу і проектуванню. На цих етапах реалізація технічних рішень перевіряється шляхом створення прототипів. Кожен виток спіралі відповідає створенню фрагмента або версії ПЗ, на ньому уточнюються мета і характеристики проекту, визначається його якість і плануються роботи наступного витка спіралі. Таким чином, заглиблюються і послідовно конкретизуються деталі проекту, і в результаті вибирається обґрунтований варіант, який доводиться до реалізації.

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на

поточному. При ітеративному способі розробки роботу по вибракуванню розробок можна буде виконати на наступній ітерації. Головна ж задача – щонайшвидше показати користувачам системи працездатний продукт, тим самим, активізуючи процес уточнення і доповнення вимог.

Основна проблема спірального циклу - визначення моменту переходу на наступний етап. Для її вирішення необхідно ввести тимчасові обмеження на кожний з етапів життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота закінчена. План складається на основі статистичних даних, одержаних в попередніх проектах, і особистого досвіду розробників.

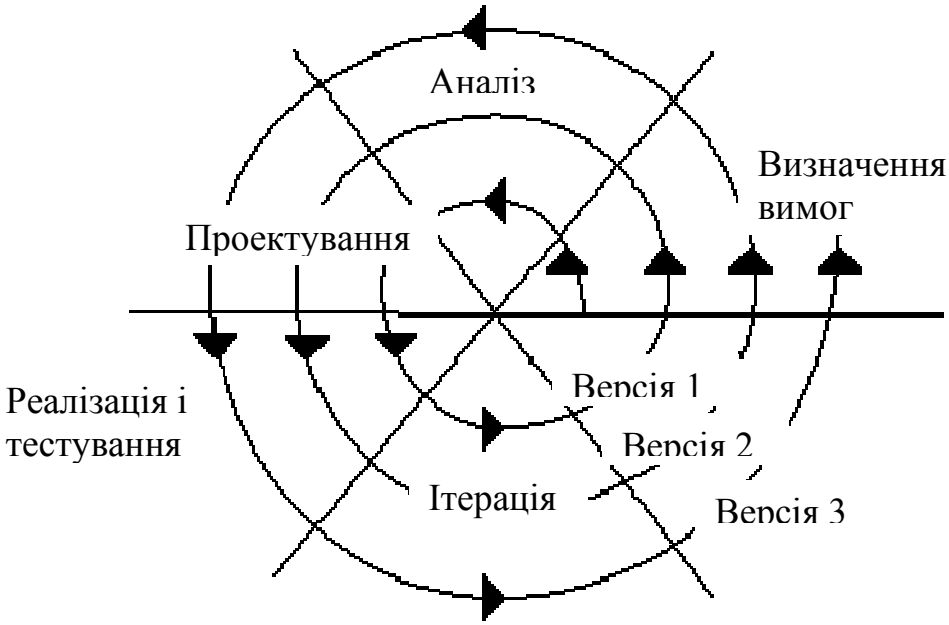


Рис. 2.3. Спіральна модель ЖЦ

2.3. Методології і технології проектування ІС

2.3.1. Загальні вимоги до методології і технології

Методології, технології і інструментальні засоби проектування (CASE-засоби) складають основу проекту будь-який ІС. Методологія реалізується через конкретні технології і стандарти, які підтримують їх, методики і інструментальні засоби, що забезпечують виконання процесів ЖЦ.

Технологія проектування визначається як сукупність трьох складових:

– покрокової процедури, що визначає послідовність технологічних операцій проектування;

– критеріїв і правил, використовуваних для оцінки результатів виконання технологічних операцій;

– нотацій (графічних і текстових засобів), використовуваних для опису проектованої системи.

Технологічні інструкції, що становлять основний зміст технології, повинні складатися з опису послідовності технологічних операцій, умов, залежно від яких виконується та або інша операція, і описів самих операцій.

Технологія проектування, розробки і супроводи ІС повинна задовольняти наступним загальним вимогам:

–підтримувати повний ЖЦ ПЗ;

–забезпечувати гарантоване досягнення цілей розробки ІС із заданою якістю і у встановлений час;

–забезпечувати можливість виконання крупних проектів у вигляді підсистем (тобто можливість декомпозиції проекту на складові частини, які розробляються групами виконавців обмеженої чисельності з подальшою інтеграцією складових частин). Досвід розробки крупних ІС показує, що для підвищення ефективності робіт необхідно розбити проект на окремі слабо зв'язані за даними і функціями підсистеми. Реалізація підсистем повинна виконуватися окремими групами фахівців. При цьому необхідно забезпечити координацію ведення загального проекту і виключити дублювання результатів робіт кожної проектної групи, яке може виникнути через наявність загальних даних і функцій;

– забезпечувати можливість ведення робіт по проектуванню окремих підсистем невеликими групами (3-7 осіб). Це обумовлено принципами керованості колективу і підвищення продуктивності за рахунок мінімізації числа зовнішніх зв'язків;

– забезпечувати мінімальний час отримання працездатної ІС. Йдеться не про терміни готовності всієї ІС, а про терміни реалізації окремих підсистем. Реалізація ІС в цілому в короткі терміни може зажадати залучення великого числа розробників, при цьому ефект може виявитися нижчим, ніж при реалізації в коротші терміни окремих підсистем меншим числом розробників;

– передбачати можливість управління конфігурацією проекту, ведення версій проекту і його складових, можливість автоматичного випуску проектної документації і синхронізацію її версій з версіями проекту;

– забезпечувати незалежність виконуваних проектних рішень від засобів реалізації ІС (систем управління базами даних, операційних систем, мов і систем програмування).

Реальне застосування будь-якої технології проектування, розробки і супроводи ІС в конкретній організації і конкретному проекті неможливе без вироблення ряду стандартів (правив, угод), які повинні дотримуватися всіма учасниками проекту, а саме:

– стандарт проектування;

– стандарт оформлення проектної документації;

– стандарт інтерфейсу, призначеного для користувача.

Стандарт проектування повинен встановлювати:

– набір необхідних моделей (діаграм) на кожній стадії проектування і ступінь їх деталізації;

– правила фіксації проектних рішень на діаграмах, зокрема: правила іменування об'єктів (включаючи угоди по термінології), набір атрибутів для всіх об'єктів і правила їх заповнення на кожній стадії, правила оформлення діаграм, включаючи вимоги до форми і розмірів об'єктів, і т. д.;

– вимоги до конфігурації робочих місць розробників, включаючи настройки операційної системи, настройки CASE-засобів, загальні настройки проекту і т. д.;

– механізм забезпечення спільної роботи над проектом, зокрема: правила інтеграції підсистем проекту, правила підтримки проекту в однаковому для всіх розробників стані (регламент обміну проектною інформацією, механізм фіксації загальних об'єктів і т.д.), правила перевірки проектних рішень на несуперечність і т.д.

Стандарт оформлення проектної документації повинен встановлювати:

– комплектність, склад і структуру документації на кожній стадії проектування;

– вимоги до її оформлення (включаючи вимоги до змісту розділів, підрозділів, пунктів, таблиць і т.д.),

– правила підготовки, розгляди, узгодження і затвердження документації з вказівкою граничних термінів для кожної стадії;

– вимоги до настройки видавничої системи, використовуваної як вбудований засіб підготовки документації;

– вимоги до настройки CASE-засобів для забезпечення підготовки документації відповідно до встановлених вимог.

Стандарт інтерфейсу користувача повинен встановлювати:

– правила оформлення екранів (шрифти і колірна палітра), склад і розташування вікон і елементів управління;

– правила використовування клавіатури і миші;

– правила оформлення текстів допомоги;

– перелік стандартних повідомлень;

– правила обробки реакції користувача

2.3.2. Методологія RAD

Одним з можливих підходів до розробки ПЗ в рамках спіральної моделі ЖЦ є методологія швидкої розробки додатків RAD (Rapid Application Development), що одержала останнім часом широке розповсюдження. Під цим терміном звичайно розуміється процес розробки ПЗ, що містить 3 елементи:

– невелику команду програмістів (від 2 до 10 чоловік);

– короткий, виробничий графік (від 2 до 6 міс.);

– цикл, що повторюється, при якому розробники, у міру того, як додаток починає оформлюватися, реалізують в продукті вимоги, одержані через взаємодію із замовником.

Команда розробників повинна являти собою групу професіоналів, що мають досвід в аналізі, проектуванні, генерації коду і тестуванні ПЗ з використанням CASE-засобів. Члени колективу повинні також уміти трансформувати в робочі прототипи пропозиції кінцевих користувачів.

Життєвий цикл ПЗ за методологією RAD складається з чотирьох фаз: 1) аналізу і планування вимог; 2) проектування; 3) побудови; 4) упровадження.

На фазі аналізу і планування вимог користувачі системи визначають функції, які вона повинна виконувати, виділяють пріоритетні з них, які вимагають опрацьовування в першу чергу, описують інформаційні потреби. Обмежується масштаб проекту, визначаються тимчасові рамки для кожної з подальших фаз. Крім того, визначається сама можливість реалізації даного проекту у встановлених рамках фінансування, на даних апаратних засобах і т.п. Результатом даної фази повинні бути список і пріоритетність функцій майбутньої ІС, попередні функціональні і інформаційні моделі ІС.

На фазі проектування частина користувачів бере участь в технічному проектуванні системи під керівництвом фахівців-розробників. CASE-засоби використовуються для швидкого отримання працюючих прототипів додатків. Користувачі, які безпосередньо взаємодіють з ними, уточнюють і доповнюють вимоги до системи, які не були виявлені на попередній фазі. Детальніше розглядаються процеси системи. Аналізується і, при необхідності, коректується функціональна модель. Кожен процес розглядається детально. При необхідності для кожного елементарного процесу створюється частковий прототип: екрану, діалогу, звіту, що знімає неясності або неоднозначності. Визначаються вимоги розмежування доступу до даних. На цій же фазі відбувається визначення набору необхідної документації.

Після детального визначення складу процесів оцінюється кількість функціональних елементів системи, що розробляється, і ухвалюється рішення про розділення ІС на підсистеми, які будуть реалізовані однією командою розробників за прийнятний для RAD-проектів час, – близько 60 - 90 днів. З використанням CASE-засобів проект розподіляється між різними командами (ділиться функціональна модель). Результатом даної фази повинні бути:

- загальна інформаційна модель системи;
- функціональні моделі системи в цілому і підсистем, реалізованих окремими командами розробників;
- точно визначені за допомогою CASE-засобу інтерфейси між підсистемами, що розробляються автономно;
- побудовані прототипи екранів, звітів, діалогів.

Всі моделі і прототипи повинні бути одержані із застосуванням тих CASE-засобів, які використовуватимуться надалі при побудові системи. Дана вимога викликана тим, що в традиційному підході при передачі інформації про проект з етапу на етап може відбутися фактично неконтрольоване спотворення даних. Застосування єдиного середовища зберігання інформації про проект дозволяє уникнути цієї небезпеки.

На відміну від традиційного підходу, при якому використовувалися специфічні засоби використання прототипів, не призначених для побудови реальних додатків, а прототипи викидалися після того, як виконано задачу. При підході RAD кожен прототип розвивається в частину майбутньої системи. Таким чином, на наступну фазу передається повніша і корисніша інформація.

На фазі побудови виконується безпосередньо сама швидка розробка додатку. На даній фазі розробники проводять ітеративну побудову реальної системи на основі одержаних в попередній фазі моделей, а також вимог не

функціонального характеру. Програмний код частково формується за допомогою автоматичних генераторів, які одержують інформацію безпосередньо з репозиторія CASE-засобів. Кінцеві користувачі на цій фазі оцінюють одержувані результати і вносять корективи, якщо в процесі розробки система перестає задовольняти визначеним раніше вимогам. Тестування системи здійснюється безпосередньо в процесі розробки.

Після закінчення робіт кожної окремої команди розробників проводиться поступова інтеграція даної частини системи з іншими, формується повний програмний код, виконується тестування спільної роботи даної частини додатку з іншими, а потім тестування системи в цілому. Завершується фізичне проектування системи:

- визначається необхідність розподілу даних;
- проводиться аналіз використовування даних;
- проводиться фізичне проектування бази даних;
- визначаються вимоги до апаратних ресурсів;
- визначаються способи збільшення продуктивності;
- завершується розробка документації проекту.

Результатом фази є готова система, що задовольняє всім узгодженим вимогам.

На фазі упровадження проводиться навчання користувачів, організаційні зміни і паралельно з упровадженням нової системи здійснюється робота з існуючою системою (до повного упровадження нової). Оскільки фаза побудови достатньо нетривала, планування і підготовка до упровадження повинні починатися наперед, як правило, на етапі проектування системи. Приведена схема розробки ІС не є абсолютною. Можливі різні варіанти, залежні, наприклад, від початкових умов, в яких ведеться розробка: розробляється абсолютно нова система; вже було проведено обстеження підприємства і існує модель його діяльності; на підприємстві вже існує деяка ІС, яка може бути використана як початковий прототип або повинна бути інтегрована з тією, що розробляється.

Слід, проте, відзначити, що методологія RAD, як і будь-яка інша, не може претендувати на універсальність, вона хороша в першу чергу для відносно невеликих проектів, що розробляються для конкретного замовника. Якщо ж розробляється типова система, яка не є закінченим продуктом, а є комплексом типових компонент, централізований супроводжуваних, адаптуються до програмно-технічних платформ, СУБД, засобів телекомунікації, організаційно-економічним особливостям об'єктів упровадження, які інтегруються з існуючими розробками, на перший план виступають такі показники проекту, як керованість і якість, бо вони можуть увійти до суперечності з простотою і швидкістю розробки. Для таких проектів необхідні високий рівень планування і жорстка дисципліна проектування, строге проходження наперед розробленим протоколам і інтерфейсам, що знижує швидкість розробки.

Методологія RAD непридатна для побудови складних розрахункових програм, операційних систем або програм управління космічними кораблями,

тобто програм, що вимагають написання великого об'єму (сотні тисяч рядків) унікального коду.

Не підходять для розробки за методологією RAD додатків, в яких відсутня яскраво виражена частина інтерфейсу, що наочно визначає логіку роботи системи (наприклад, додатки реального часу) і додатків, від яких залежить безпека людей (наприклад, керування літаком або атомною електростанцією), оскільки ітеративний підхід припускає, що перші декілька версій напевно не будуть повністю працездатні, що в даному випадку виключається.

Оцінка розміру додатків проводиться на основі так званих функціональних елементів (екрани, повідомлення, звіти, файли і т.п.) Подібна метрика не залежить від мови програмування, на якому ведеться розробка. Розмір додатку, який створюється за методологією RAD, для середовища розробки ІС, яке маэ добру відладку, з максимальним повторним використанням програмних компонентів, визначається таким чином:

– менше 1000 функціональних елементів – одна людина;

– 1000-4000 функціональних елементів – одна команда розробників;

– більше 4000 функціональних елементів – по 4000 функціональних елементів на одну команду розробників.

Контрольні запитання

1. Що таке життєвий цикл програмного забезпечення?
2. Які ви знаєте моделі життєвого циклу?
3. Яка з моделей життєвого циклу є найбільш об’єктивною?
4. Що таке стандарт, у поняттях документації на розробку програмного забезпечення?
5. Поясніть сутність методики RAD.

В розділі розглянуто основні поняття миттєвого циклу інформаційної системи, його моделі, загальні вимоги до методології і технології проектування ІС, зокрема, методологія RAD.

3. СТРУКТУРНИЙ ПІДХІД ДО ПРОЕКТУВАННЯ ІС

Розглянуто систему опису елементів інформаційної системи та принципів поєднання цих елементів у термінах CASE-технології.

3.1. Сутність структурного підходу

Сутність структурного підходу до розробки ІС полягає в її декомпозиції (розбитті) на функції, що автоматизуються: система розбивається на функціональні підсистеми, які в свою чергу діляться на підфункції, що підрозділяються на задачі і так далі. Процес розбиття продовжується аж до конкретних процедур. Система, що при цьому автоматизується, зберігає цілісне уявлення, в якому всі компоненти, які складають систему, взаємопов'язані. При розробці системи "знизу-вгору" від окремих задач до всієї системи цілісність втрачається, виникають проблеми при інформаційній стиковці окремих компонентів.

Всі найпоширеніші методології структурного підходу базуються на ряді загальних принципів:

- принцип "розділяй і володарюй" – принцип рішення складних проблем шляхом їх розбиття на множину менших незалежних задач, легких для розуміння і рішення;
- принцип ієрархічного впорядковування – принцип організації складових частин проблеми в ієрархічні деревовидні структури з додаванням нових деталей на кожному рівні;
- принцип абстрагування полягає у виділенні істотних аспектів системи і відвернення від неістотних;
- принцип формалізації полягає в необхідності строгого методичного підходу до рішення проблеми;
- принцип несуперечності полягає в обґрунтованості і узгодженості елементів;
- принцип структуризації даних полягає у тому, що дані повинні бути структуровані і ієрархічно організовані.

У структурному аналізі використовуються в основному дві групи засобів, що ілюструють функції, виконувані системою, і відносини між даними. Кожній групі засобів відповідають певні види моделей (діаграм), найпоширенішими серед яких є наступні:

- SADT (Structured Analysis and Design Technique) моделі і відповідні функціональні діаграми (п 3.2);
- DFD (Data Flow Diagrams) діаграми потоків даних (п. 3.3);
- ERD (Entity-Relationship Diagrams) діаграми "сутність-зв'язок" (п. 3.4).

3.2. Методологія функціонального моделювання SADT

Методологія SADT є сукупністю методів, правил і процедур, призначених для побудови функціональної моделі об'єкту якої-небудь наочної області. Функціональна модель SADT відображає функціональну структуру об'єкту, тобто вироблювані їм дії і зв'язки між цими діями і ґрунтуються на наступних концепціях:

– графіка блоків і дуг SADT-діаграми відображають функцію у вигляді блоку, а інтерфейси входу/виходу представляються дугами, що відповідно входять в блок і виходять з нього. Взаємодія блоків один з одним описуються за допомогою дуг інтерфейсу, що виражають "обмеження", які в свою чергу визначають, коли і яким чином функції виконуються і управляються;

– виконання правил SADT вимагає достатньої строгості і точності, не накладаючи у той же час надмірних обмежень на дії аналітика;

– обмеження кількості блоків на кожному рівні декомпозиції (правило 3-6 блоків);

– зв'язність діаграм (номери блоків);

– унікальність влучних найменувань (відсутність імен, що повторюються);

– синтаксичні правила для графіки (блоків і дуг);

– розділення входів і управлінь (правило визначення ролі даних).

– відділення організації від функції, тобто виключення впливу організаційної структури на функціональну модель.

3.2.1. Склад функціональної моделі

Результатом застосування методології SADT є модель, яка складається з діаграм, фрагментів текстів і глосарію (словнику), які мають посилання одне на одного. Діаграми – це головні компоненти моделі, всі функції ІС і інтерфейси на них представлені як блоки і дуги. Місце з'єднання дуги з блоком визначає тип інтерфейсу. Управляюча інформація входить в блок зверху, тоді як інформація, яка піддається обробці, показана з лівого боку блоку, а результати виходу показані з правого боку. Механізм (людина або автоматизована система), який здійснює операцію, представляється дугою, що входить в блок знизу (рис. 3.1).

Однією з найважливіших особливостей методології SADT є поступове введення все більших рівнів деталізації у міру створення діаграм, що відображають модель.

На рис. 3.2 наведені чотири діаграми і їх взаємозв'язки. Кожен компонент моделі може бути підданий декомпозиції на іншій діаграмі. Кожна діаграма ілюструє "внутрішню будову" блоку на батьківській діаграмі.

3.2.2. Ієрархія діаграм

Побудова SADT-моделі починається з представлення всієї системи у

вигляді простої компоненти – одного блоку і дуг, що зображають інтерфейси з функціями поза системою. Оскільки єдиний блок представляє всю систему як єдине ціле, ім'я, вказане в блоці, є загальним. Це вірно і для дуг інтерфейсу – вони також представляють повний набір зовнішніх інтерфейсів системи в цілому.

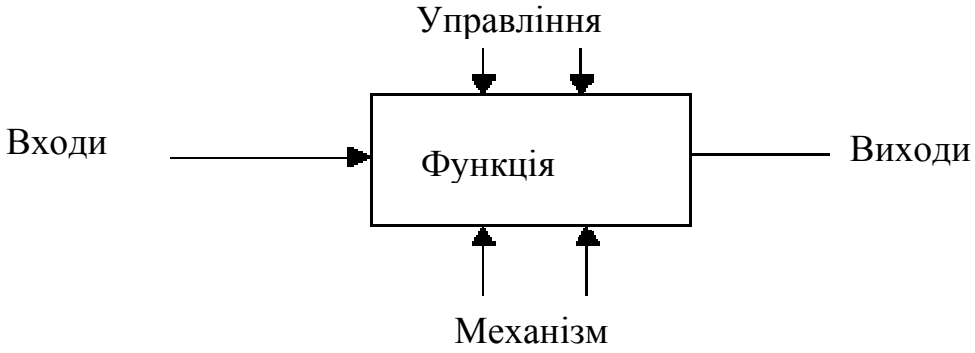


Рис. 3.1. Функціональний блок і дуги інтерфейсу

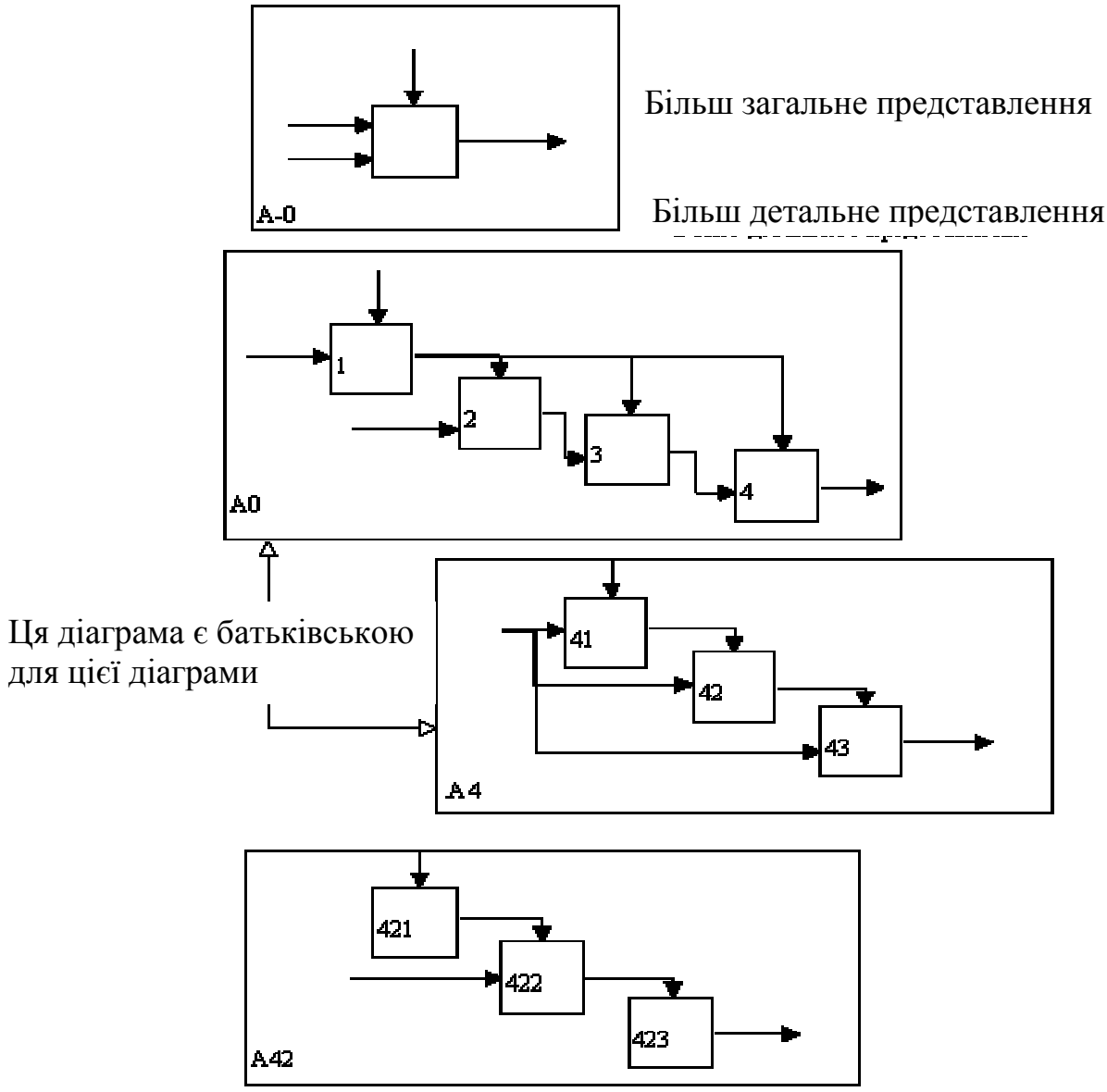


Рис. 3.2. Структура SADT-моделі. Декомпозиція діаграм

Потім блок, який представляє систему як єдиний модуль, деталізується на іншій діаграмі за допомогою декількох блоків, сполучених інтерфейсними дугами. Ці блоки представляють основні підфункції початкової функції. Дана декомпозиція виявляє повний набір підфункцій, кожна з яких представлена як блок, межі якого визначені інтерфейсними дугами. Кожна з цих підфункцій може бути піддана декомпозиції так само для детальнішого уявлення.

У всіх випадках кожна підфункція може містити тільки ті елементи, які входять в початкову функцію. Крім того, модель не може пропустити які-небудь елементи, тобто, як вже наголошувалося, батьківський блок і його інтерфейси забезпечують контекст. До нього не можна нічого додати, і з нього не може бути нічого видалено.

На рис. 3.3 - 3.5 представлені різні варіанти виконання функцій і з'єднання дуг з блоками.

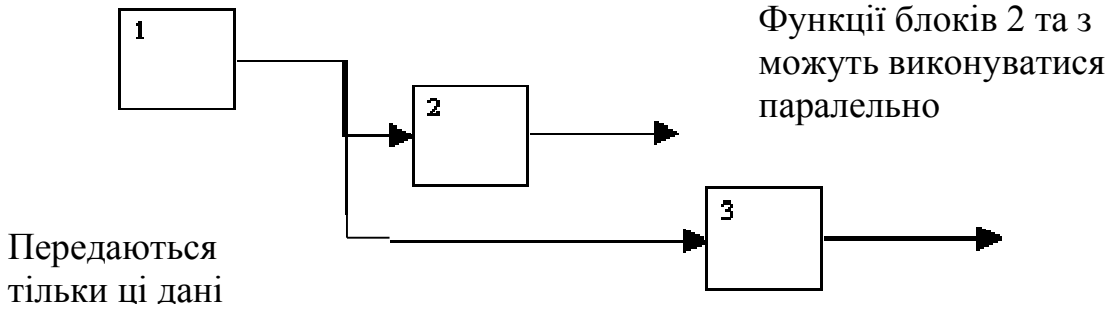


Рис. 3.3. Одночасне виконання

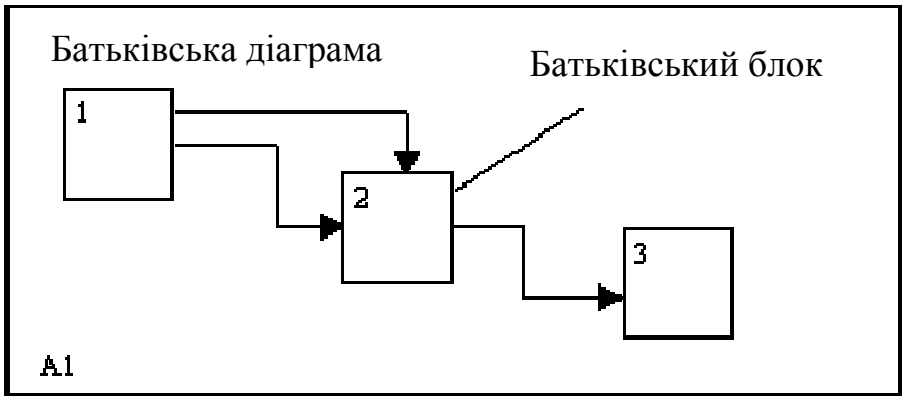


Рис. 3.4. Відповідність повинна бути повною і несуперечливою

Деякі дуги приєднані до блоків діаграми обома кінцями, у інших же один кінець залишається не приєднаним. Не приєднані дуги відповідають входам, управлінням і виходам батьківського блоку. Джерело або одержувач цих прикордонних дуг може бути знайдено тільки на батьківській діаграмі. Не приєднані кінці повинні відповідати дугам на початковій діаграмі. Всі граничні дуги повинні продовжуватися на батьківській діаграмі, щоб вона була повною і несуперечливою.

На SADT-діаграмах не вказані явно ні послідовність, ні час. Зворотні зв'язки, ітерації, процеси, що продовжуються, і функції, що перекриваються (за часом), можуть бути зображені за допомогою дуг. Зворотні зв'язки можуть виступати у вигляді коментарів, зауважень, виправлень і т.д. (рис. 3.5).

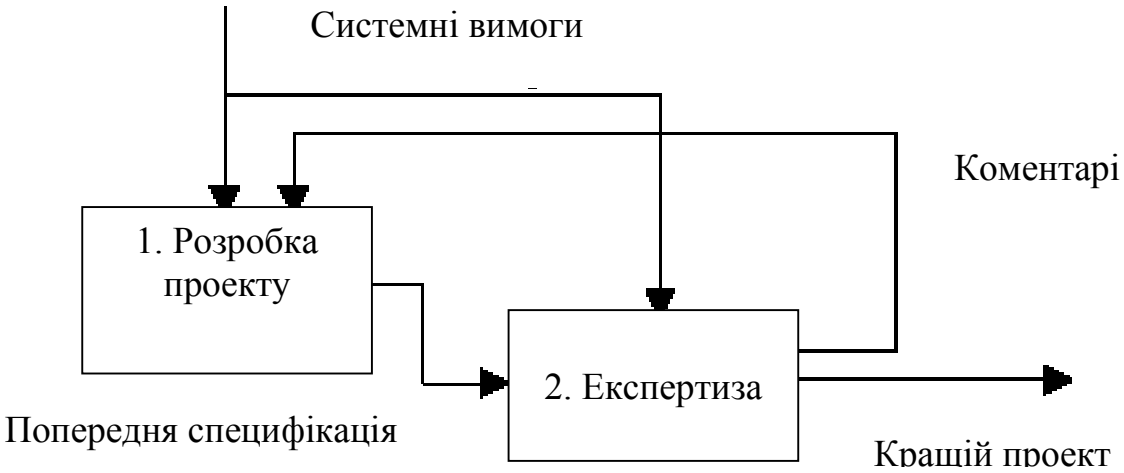


Рис. 3.5. Приклад зворотного зв'язку

Як було відмічено, механізми (дуги з нижньої сторони) показують засоби, за допомогою яких здійснюється виконання функцій. Механізм може бути людиною, комп'ютером або будь-яким іншим пристроєм, який допомагає виконувати дану функцію (рис. 3.6).

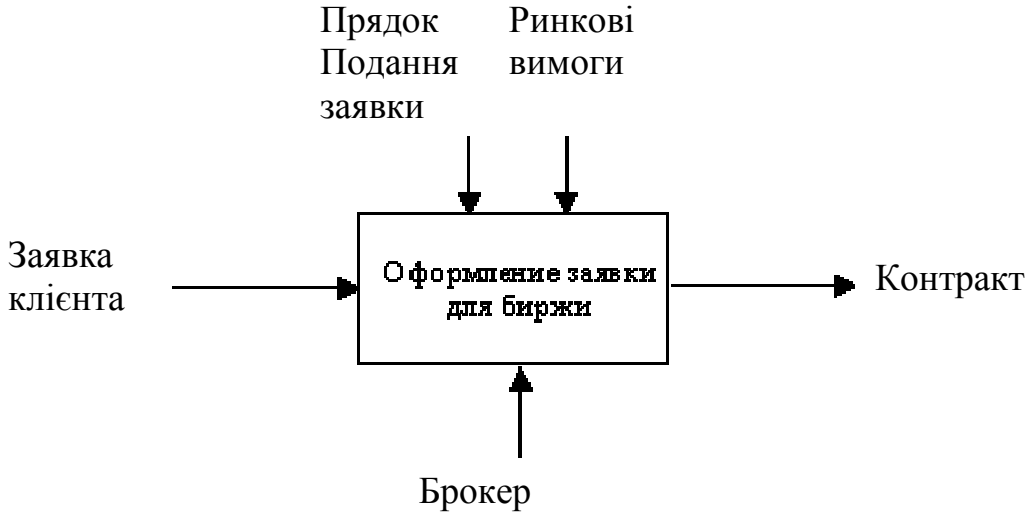


Рис. 3.6. Приклад механізму

Кожен блок на діаграмі має свій номер. Блок будь-якої діаграми може бути далі описаний діаграмою нижнього рівня, яка, у свою чергу, може бути далі деталізована за допомогою необхідного числа діаграм. Таким чином, формується ієрархія діаграм.

Для того, щоб вказати положення будь-якої діаграми або блоку в ієрархії, використовуються номери діаграм. Наприклад, A21 є діаграмою, яка деталізує блок 1 на діаграмі A2. Аналогічно, A2 деталізує блок 2 на діаграмі A0, яка є самою верхньою діаграмою моделі. На рис. 3.7 показане типове дерево діаграм.

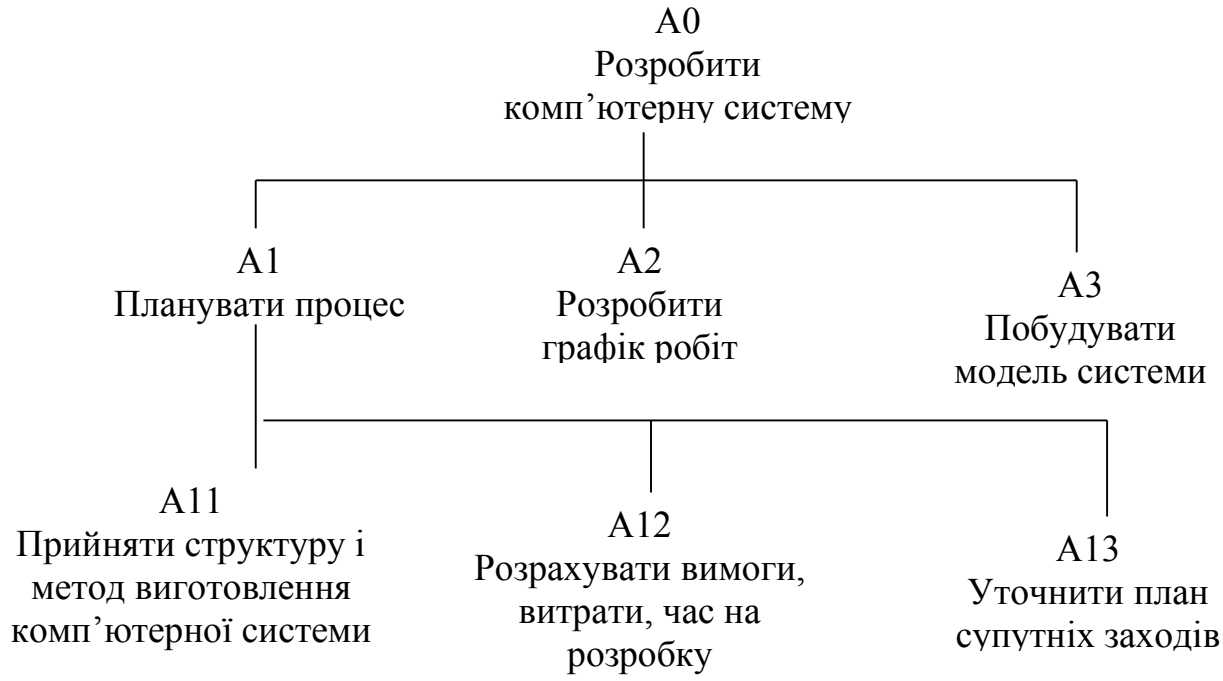


Рис. 3.7. Ієрархія діаграм

3.2.3. Типи зв'язків між функціями

Одним з важливих моментів при проектуванні ІС за допомогою методології SADT є точна узгодженість типів зв'язків між функціями.

Тип зв'язку	Відносна значущість
Випадкова	0
Логічна	1
Тимчасова	2
Процедурна	3
Комунікаційна	4
Послідовна	5
Функціональна	6

Нижче кожен тип зв'язку стисло визначений і проілюстрований за допомогою типового прикладу з SADT.

(0) Тип випадкової зв'язності: якнайменше бажаний.

Випадкова зв'язність виникає, коли конкретний зв'язок між функціями малий або повністю відсутній. Це відноситься до ситуації, коли імена даних на SADT-дугах в одній діаграмі мають малий зв'язок один з одним. Крайній варіант цього випадку показаний на рис. 3.8.

(1) Тип логічної зв'язності. Логічне скріплення відбувається тоді, коли дані і функції збираються разом унаслідок того, що вони потрапляють в загальний клас або набір елементів, але необхідних функціональних відносин між ними не виявляється.

(2) Тип тимчасової зв'язності. Зв'язані за часом елементи виникають унаслідок того, що вони представляють функції, зв'язані в часі, коли дані використовуються одночасно або функції включаються паралельно, а не послідовно.

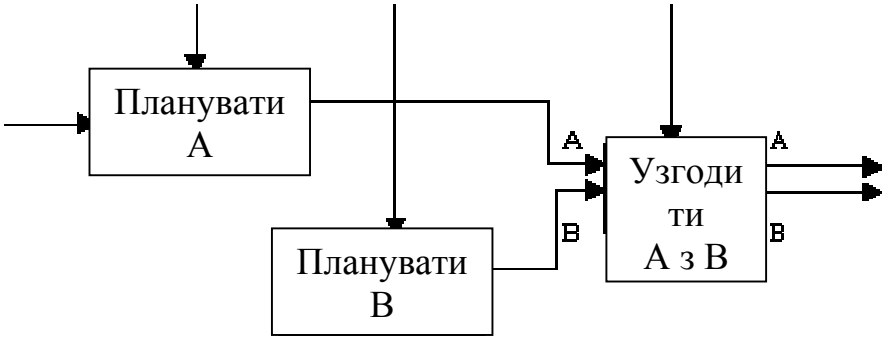


Рис. 3.8. Випадкова зв'язність

(3) Тип процедурної зв'язності. Процедурно зв'язані елементи з'являються згрупованими разом унаслідок того, що вони виконуються протягом однієї і тієї ж частини циклу або процесу. Приклад процедурно-зв'язаної діаграми приведений на рис. 3.9.

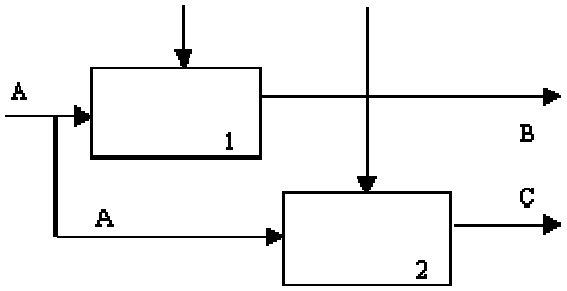


Рис. 3.9. Процедурна зв'язність

(4) Тип комунікаційної зв'язності. Діаграми демонструють комунікаційні зв'язки, коли блоки групуються унаслідок того, що вони використовують одні і ті ж вхідні дані та/або проводять одні і ті ж вихідні дані (рис. 3.10).

(5) Тип послідовної зв'язності. На діаграмах, що мають послідовні зв'язки, вихід однієї функції служить вхідними даними для наступної функції. Зв'язок між елементами на діаграмі є тіснішим, ніж на розглянутих вище рівнях зв'язок, оскільки моделюються причинно-наслідкові залежності (рис. 3.11).

(6) Тип функціональної зв'язності. Діаграма відображає повну функціональну зв'язність, за наявності повної залежності однієї функції від іншої. Діаграма, яка є чисто функціональною, не містить чужорідних елементів, що відносяться до послідовного або слабкішого типу зв'язності. Одним із способів визначення функціонально-зв'язаних діаграм є розгляд двох блоків, зв'язаних через управляючі дуги, як показано на рис. 3.12.

Нижче в таблиці представлені всі типи зв'язків, розглянуті вище. Важливо відзначити, що рівні 4-6 встановлюють типи зв’язностей, які розробники вважають найважливішими для отримання діаграм хорошої якості.

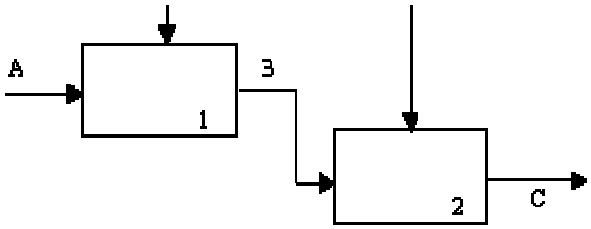


Рис. 3.10. Комунікаційна зв'язність

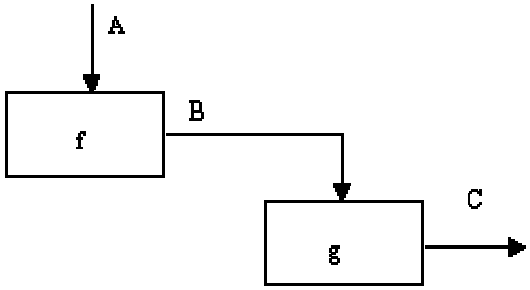


Рис. 3.11. Послідовна зв'язність

Значущість	Тип зв'язності	Для функцій	Для даних
0	Випадкова	Випадкова	Випадкова
1	Логічна	Функції однієї і тієї ж множини або типу (наприклад, "редагувати всі входи")	Дані однієї і тієї ж множини або типу
2	Тимчасова	Функції одного і того ж періоду часу (наприклад, "операції ініціалізації")	Дані, використовувані в якому-небудь тимчасовому інтервалі
3	Процедурна	Функції, що працюють в одній і тій же фазі або ітерації (наприклад, "перший прохід компілятора")	Дані, використовувані під час однієї і тієї ж фази або ітерації
4	Комунікаційна	Функції, що використовують одні і ті ж дані	Дані, на які впливає одна і та ж діяльність
5	Послідовна	Функції, що виконують послідовні перетворення одних і тих же даних	Дані, перетворювані послідовними функціями
6	Функціональна	Функції, об'єднувані для виконання однієї функції	Дані, пов'язані з однією функцією

3.3. Моделювання потоків даних (процесів)

У основі даної методології (методології Gane/Sarson) лежить побудова моделі аналізованої ІС – тієї, що проектується або реально існуючої. Відповідно до методології модель системи визначається як ієрархія діаграм потоків даних (ДПД або DFD), що описують асинхронний процес перетворення інформації від її введення в систему до видачі користувачу. Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси або підсистеми ІС із зовнішніми входами і виходами. Вони деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція продовжується, створюючи багаторівневу

ієрархію діаграм, до тих пір, поки не буде досягнутий такий рівень декомпозиції, на якому процес стає елементарними і деталізувати його далі неможливо.

Джерела інформації (зовнішня сутність) породжують інформаційні потоки (потоки даних), що переносять інформацію до підсистем або процесів. Ті в свою чергу перетворять інформацію і породжують нові потоки, які переносять інформацію до інших процесів або підсистем, накопичувачів даних або зовнішньої сутності – споживачів інформації. Таким чином, основними компонентами діаграм потоків даних є: 1) зовнішня сутність; 2) системи/підсистеми; 3) процеси; 4) накопичувачі даних; 5) потоки даних.

3.3.1. Зовнішня сутність

Зовнішня сутність є матеріальним предметом або фізичною особою, що є джерелом або приймачем інформації, наприклад, замовники, персонал, постачальники, клієнти, склад. Визначення деякого об'єкту або системи як зовнішня сутність указує на те, що вона знаходиться за межами аналізованої ІС. В процесі аналізу деяка зовнішня сутність може бути перенесені всередину діаграми аналізованої ІС, якщо це необхідно, або, навпаки, частина процесів ІС може бути винесена за межі діаграми і представлена як зовнішня сутність.

Зовнішня сутність позначається квадратом (рис. 3.13), розташованим як би "над" діаграмою для того, щоб можна було виділити цей символ серед інших позначень:

Рис. 3.13. Зовнішня сутність

3.3.2. Системи і підсистеми

При побудові моделі складної ІС вона може бути представлена в найзагальнішому вигляді на так званій контекстній діаграмі у вигляді однієї системи як єдиного цілого, або може бути розділена на ряд підсистем.

Підсистема (або система) на контекстній діаграмі зображається таким чином (рис. 3.14).

Номер підсистеми служить для її ідентифікації. У полі імені вводиться найменування підсистеми у вигляді пропозиції з підметом і відповідними визначеннями і доповнення

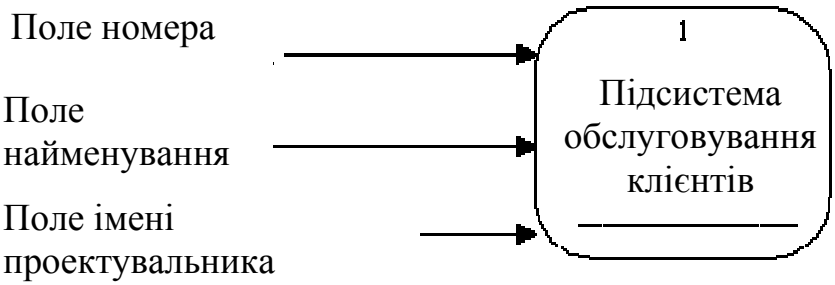


Рис. 3.14. Підсистема

3.3.3. Процеси

Процес є перетворенням вхідних потоків даних у вихідні відповідно до певного алгоритму. Фізично процес може бути реалізований різними способами: це може бути підрозділ організації (відділ), що виконує обробку вхідних документів і випуск звітів, програма, апаратний реалізований логічний пристрій і т.д. Процес на діаграмі потоків даних зображається, як показано на рис. 3.15.

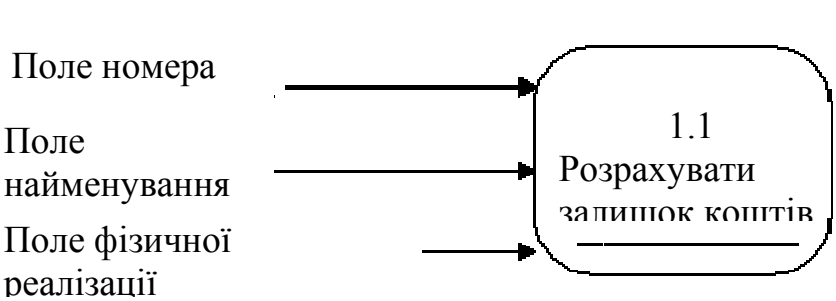


Рис. 3.15. Процес

Номер процесу слугує для його ідентифікації. У полі імені вводиться найменування процесу у вигляді пропозиції з активним недвозначним дієсловом в невизначеній формі (обчислити, розрахувати, перевірити, визначити, створити, одержати), за яким слідують іменники в знахідному відмінку, наприклад:

"Ввести відомості про клієнтів";

"Видати інформацію про поточні витрати";

"Перевірити кредитоспроможність клієнта".

Використання таких дієслів, як "обробити", "модернізувати" або "відредагувати" означає, як правило, недостатньо глибоке розуміння даного процесу і вимагає подальшого аналізу.

Інформація в полі фізичної реалізації показує, який підрозділ організації, програма або апаратний пристрій виконує даний процес.

3.3.4. Накопичувачі даних

Накопичувач даних є абстрактним пристроєм для зберігання інформації, яку можна у будь-який момент помістити в накопичувач і через деякий час витягнути, причому способи вміщення і витягання можуть бути будь-якими.

Накопичувач даних на діаграмі потоків даних зображається, як показано на рис. 3.16.

Накопичувач даних ідентифікується буквою "D" і довільним числом. Ім'я накопичувача вибирається з міркування найбільшої інформативності для проектувальника.

Рис. 3.16. Накопичувач даних

Накопичувач даних в загальному випадку є прообразом майбутньої бази даних і опис даних, що зберігаються в ньому, повинен бути пов'язане з інформаційною моделлю.

3.3.5. Потоки даних

Потік даних визначає інформацію, яка передається через деяке з'єднання від джерела до приймача. Реальний потік даних може бути інформацією, що передається кабелем між двома пристроями, пересланою поштою, магнітними стрічками або дискетами, перенесеною з одного комп'ютера на іншій і т.д.

Потік даних на діаграмі зображається лінією, стрілкою, що закінчується, яка показує напрям потоку (рис. 3.17). Кожен потік даних має ім'я, що відображає його зміст.

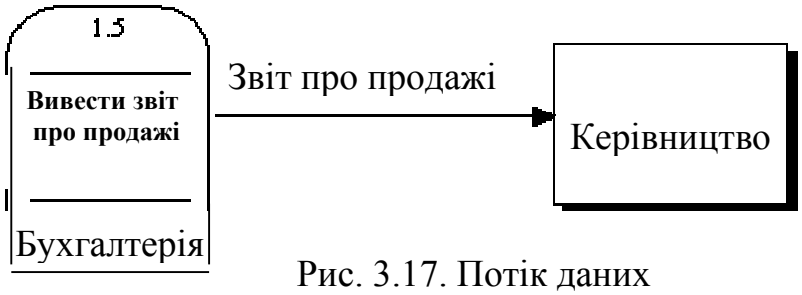


Рис. 3.17. Потік даних

3.3.6. Побудова ієрархії діаграм потоків даних

Першим кроком при побудові ієрархії ДПД є побудова контекстних діаграм. При проектуванні простих ІС будується єдина контекстна діаграма із зіркоподібною топологією, в центрі якої знаходиться так званий головний процес, сполучений з приймачами і джерелами інформації, за допомогою яких із системою взаємодіють користувачі й інші зовнішні системи.

Якщо для складної системи обмежитися єдиною контекстною діаграмою, то вона міститиме дуже велику кількість джерел і приймачів інформації, які важко розташувати на листі паперу нормального формату, і крім того, єдиний головний процес не розкриває структури розподіленої системи. Ознаками складності (у значенні контексту) можуть бути:

- наявність великої кількості зовнішніх сутностей (десять і більше);
- розподілена природа системи;
- багатофункціональність системи з угрупуванням функцій, що вже склалося або виявленою, в окремі підсистеми.

Для складних ІС будується ієрархія контекстних діаграм. При цьому контекстна діаграма верхнього рівня містить не єдиний головний процес, а набір підсистем, сполучених потоками даних. Контекстні діаграми наступного рівня деталізують контекст і структуру підсистем.

Ієрархія контекстних діаграм визначає взаємодію основних функціональних підсистем проектованої ІС як між собою, так і із зовнішніми вхідними і вихідними потоками даних і зовнішніми об'єктами (джерелами і приймачами інформації), з якими взаємодіє ІС.

Розробка контекстних діаграм вирішує проблему строгого визначення функціональної структури ІС на ранній стадії її проектування, що особливо важливо для складних багатофункціональних систем, в розробці яких беруть участь різні організації і колективи розробників.

Після побудови контекстних діаграм одержану модель слід перевірити на повноту початкових даних про об'єкти системи й ізольованість об'єктів (відсутність інформаційних зв'язків із іншими об'єктами).

Для кожної підсистеми, присутньої на контекстних діаграмах, виконується її деталізація за допомогою ДПД. Кожен процес на ДПД, у свою чергу, може бути деталізований за допомогою ДПД або мініспецификації. При деталізації повинні виконуватися наступні правила:

- правило балансування означає, що при деталізації підсистеми або процесу діаграма, яка деталізує зовнішні джерела/приймачі даних, може мати тільки ті компоненти (підсистеми, процеси, зовнішню сутність, накопичувачі даних), з якими має інформаційний зв'язок підсистема, що деталізується, або процес на батьківській діаграмі;
- правило нумерації означає, що при деталізації процесів повинна підтримуватися їх ієрархічна нумерація. Наприклад, процеси, що деталізують процес з номером 12, одержують номери 12.1, 12.2, 12.3 і т.д.

Мініспецификація (опис логіки процесу) повинна формулювати його основні функції так, щоб надалі фахівець, який виконує реалізацію проекту, зміг виконати їх або розробити відповідну програму.

Мініспецификація є кінцевою вершиною ієрархії ДПД. Рішення про завершення деталізації процесу і використання мініспецификації ухвалюється аналітиком виходячи з таких критеріїв:

– наявність у процесу невеликої кількості вхідних і вихідних потоків даних (2-3 потоки);

– можливості опису перетворення даних процесом у вигляді послідовного алгоритму;

– виконання процесом єдиної логічної функції перетворення вхідної інформації у вихідну;

– можливості опису логіки процесу за допомогою мініспецификації невеликого об'єму (не більш 20-30 рядків).

При побудові ієрархії ДПД переходити до деталізації процесів слід тільки після визначення змісту всіх потоків і накопичувачів даних, які описується за допомогою структур даних. Структури даних конструюються з елементів даних і можуть містити альтернативи, умовні входження й ітерації. Умовне входження означає, що даний компонент може бути відсутнім в структурі. Альтернатива означає, що в структуру може входити один з перерахованих елементів. Ітерація означає входження будь-якого числа елементів у вказаному діапазоні. Для кожного елементу даних може указуватися його тип (безперервні або дискретні дані). Для безперервних даних може указуватися одиниця вимірювання (кг, см і т.п.), діапазон значень, точність уявлення і форма фізичного кодування. Для дискретних даних може указуватися таблиця допустимих значень.

Після побудови закінченої моделі системи її необхідно перевірити на повноту і узгодженість (провести верифікацію). У повній моделі всі її об'єкти (підсистеми, процеси, потоки даних) повинні бути детально описані і деталізовані. Виявлені не деталізовані об'єкти слід деталізувати, повертаючись на попередні кроки розробки. В узгодженій моделі для всіх потоків даних і накопичувачів даних повинне виконуватися правило збереження інформації: всі дані, які надходять куди-небудь, повинні бути перелічені, а всі дані, які було прочитано, повинні бути записані.

3.4. Моделювання даних

3.4.1. Case-метод Баркера

Мета моделювання даних полягає в забезпеченні розробника ІС концептуальною схемою бази даних у формі однієї моделі або декількох локальних моделей, які відносно легко можуть бути відображені в будь-яку систему баз даних.

Найпоширенішим засобом моделювання даних є діаграми "сутність-зв'язок" (ERD). З їх допомогою визначаються важливі для наочної області об'єкти (сутність), їх властивості (атрибути) і відносини один з одним (зв'язки). ERD безпосередньо використовуються для проектування реляційних баз даних.

Нотація ERD показано на прикладі моделювання діяльності компанії по торгівлі автомобілями. Нижче приведені витяги з інтерв'ю, проведеного з персоналом компанії.

Головний менеджер: один з основних обов'язків – склад автомобільного майна. Він повинен знати, скільки заплачено за машини і які накладні витрати. Володіючи цією інформацією, він може встановити нижню ціну, за яку міг би продати даний екземпляр. Крім того, він несе відповідальність за продавців і йому потрібно знати, хто що продає і скільки машин продав кожен із них.

Продавець: йому потрібно знати, яку ціну запрошувати і яка нижня ціна, за яку можна виконати операцію. Крім того, йому потрібна основна інформація про машини: рік випуску, марка, модель і т.п. Адміністратор: його задача зводиться до складання контрактів, для чого потрібна інформація про покупця, автомашину і продавця, оскільки саме контракти приносять продавцям винагороди за продажі.

Перший крок моделювання – витягання інформації з інтерв'ю і виділення сутності. Сутність (Entity) – реальний або уявний об'єкт, що має істотне значення для даної наочної області, інформація про яке підлягає зберіганню (рис. 3.18).

Рис. 3.18. Графічне зображення сутності

Кожна сутність повинна володіти унікальним ідентифікатором. Кожен екземпляр сутності повинен однозначно ідентифікуватися і відрізнятися від всіх інших екземплярів даного типу сутності. Кожна сутність повинна володіти деякими властивостями:

- кожна сутність повинна мати унікальне ім'я, і до одного і того ж імені повинна завжди застосовуватися одна і та ж інтерпретація. Одна і та ж інтерпретація не може застосовуватися до різних імен, якщо тільки вони не є псевдонімами;
- сутність володіє одним або декількома атрибутами, які або належать сутності, або успадковуються через зв'язок;
- сутність володіє одним або декількома атрибутами, які однозначно ідентифікують кожен екземпляр сутності;

- кожна сутність може володіти будь-якою кількістю зв'язків з іншою сутністю моделі.

Звертаючись до приведених вище витягів з інтерв'ю, видно, що сутності, які можуть бути ідентифіковані з головним менеджером – це автомашини і продавці. Продавцю важливі автомашини і пов'язані з їх продажем дані. Для адміністратора важливі покупці, автомашини, продавці і контракти. Виходячи з цього, виділяються 4 сутності (автомашина, продавець, покупець, контракт), які зображаються на діаграмі таким чином (рис. 3.19).

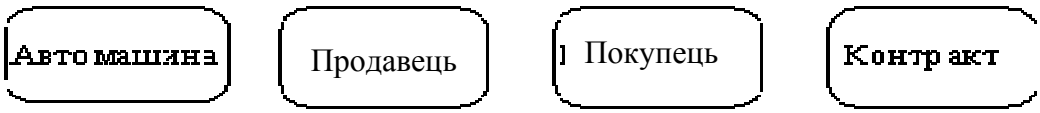


Рис. 3.19. Сутності з прикладу

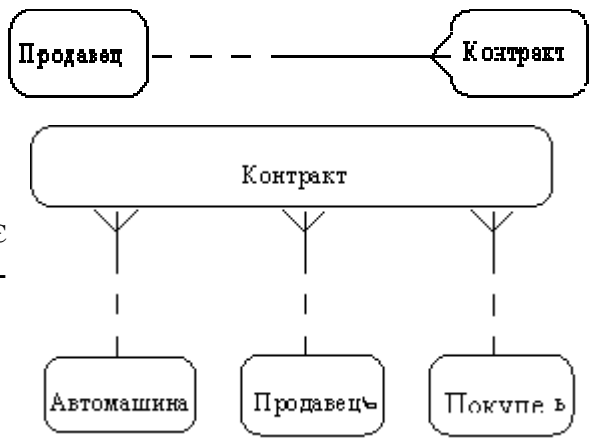
Наступним кроком моделювання є ідентифікація зв'язків.

Зв'язок (Relationship) – пойменована асоціація між двома сутностями, значуща для даної наочної області. Зв'язок – це асоціація між сутністю, при якій, як правило, кожен екземпляр однієї сутності, званої батьківською сутністю, асоційований з довільною (зокрема нульовою) кількістю екземплярів другої сутності, званої сутністю-нащадком, а кожен екземпляр сутності-нащадка асоційований в точності з одним екземпляром сутності-батька. Таким чином, екземпляр сутності-нащадка може існувати тільки при існуванні сутності батька.

Зв'язку може даватися ім'я, що виражатиметься граматичним оборотом дієслова і поміщається біля лінії зв'язку. Ім'я кожного зв'язку між двома сутностями повинен бути унікальним, але імена зв'язків в моделі не зобов'язані бути унікальними. Ім'я зв'язку завжди формується з погляду батька, так що пропозиція може бути утворене з'єднанням імені сутності-батька, імені зв'язку, виразу ступеня і імені сутності-нащадка. Наприклад, зв'язок продавця з контрактом може бути виражена таким чином: продавець може одержати винагороду за 1 або більш контрактів; контракт повинен бути ініційований рівно одним продавцем. Ступінь зв'язку і обов'язковість графічно зображаються таким чином (рис. 3.20).

Рис. 3.20. Ступінь зв'язку і її обов'язковість

Таким чином, 2 пропозиції, що описують зв'язок продавця з контрактом, графічно будуть виражені на рис. 3.21.



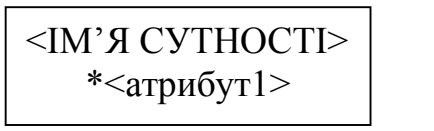
Схему зв'язків решти сутностей, подана на рис. 3.22.

Останнім кроком моделювання є ідентифікація атрибутів. Атрибут – будь-

яка характеристика сутності, значуща для даної наочної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або виразу стану сутності. Атрибут представляє тип характеристик або властивостей, асоційованих з безліччю реальних або абстрактних об'єктів (людей, місць, подій, станів, ідей, пар предметів і т.д.). Екземпляр атрибуту - це певна характеристика окремого елементу множини. Екземпляр атрибуту визначається типом характеристики і її значенням, званим значенням атрибуту. Атрибути асоціюються з конкретною сутністю. Таким чином, екземпляр сутності повинен володіти єдиним певним значенням для асоційованого атрибуту.

Атрибут може бути або обов'язковим, або необов'язковим (рис. 3.23). Обов'язковість означає, що атрибут не може приймати невизначених значень (null values). Атрибут може бути або описовим (тобто звичним дескриптором сутності), або входити до складу унікального ідентифікатора (первинного ключа).

Унікальний ідентифікатор – це атрибут або сукупність атрибутів і/або зв'язків, призначена для унікальної ідентифікації кожного екземпляра даного типу сутності. У разі повної ідентифікації кожен екземпляр даного типу сутності повністю ідентифікується своїми власними ключовими атрибутами, інакше в його ідентифікації беруть участь також атрибути іншої сутності-батька (рис. 3.24).



* – обов'язковий атрибут
o – не обов'язковий атрибут

Рис. 3.23.

Рис. 3.24.
1. - повна ідентифікація. 2 - ідентифікація через іншу сутність

Кожен атрибут ідентифікується унікальним ім'ям, що виражається граматичним оборотом іменника, описує характеристику, що представляється атрибутом. Атрибути зображаються у вигляді списку імен усередині блоку асоційованої сутності, причому кожен атрибут займає окремий рядок. Атрибути, що визначають первинний ключ, розміщуються нагорі списку і виділяються знаком "#".

Кожна сутність повинна володіти хоча б одним можливим ключем. Можливий ключ сутності – це один або декілька атрибутів, чиї значення однозначно визначають кожен екземпляр сутності. При існуванні декількох можливих ключів один з них позначається як первинний ключ, а інші – як альтернативні ключі.

З урахуванням наявної інформації доповнимо побудовану раніше діаграму (рис. 3.25). Визначимо підтипи і супертипи: одна сутність є узагальненим поняттям для групи подібної сутності (рис. 3.26). Зв'язки, що взаємно виключають: кожен екземпляр сутності бере участь тільки в одному зв'язку з групи зв'язків, що взаємно виключають (рис. 3.27).

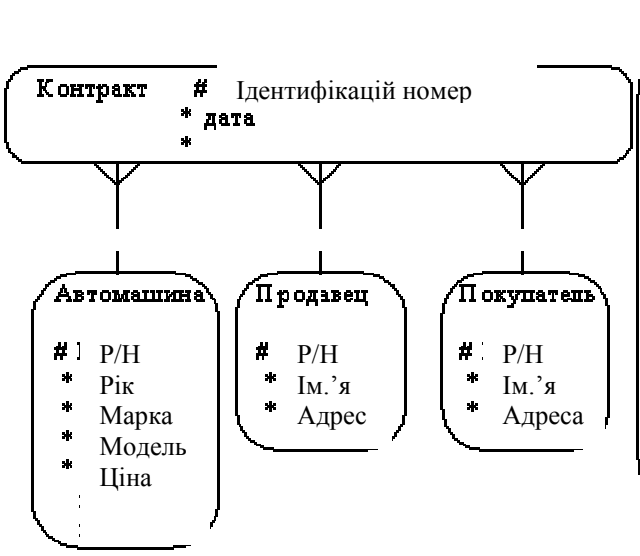


Рис. 3.25.

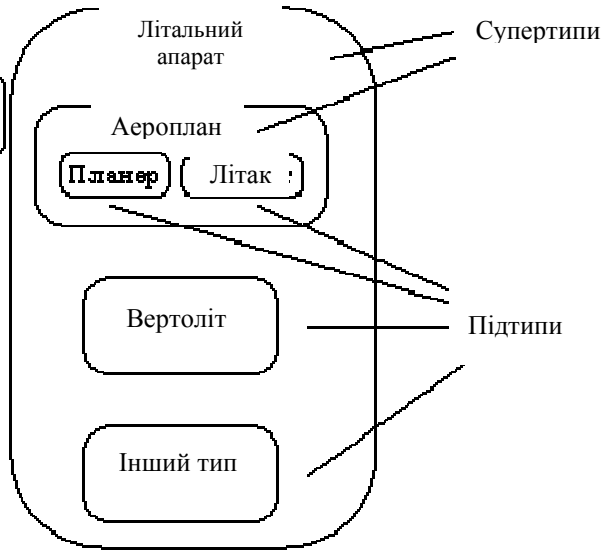


Рис. 3.26. Підтипи і супертипи

Визначимо поняття рекурсивного зв'язку: сутність може бути зв'язана сама з собою (рис. 3.28).

Неперeміщувані (non-transferrable) зв'язки: екземпляр сутності не може бути перенесений з одного екземпляра зв'язку в інший (рис. 3.29).

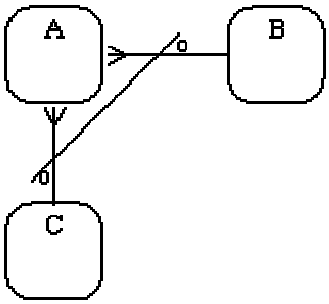


Рис. 3.27. Зв'язки, що взаємно виключають одне одного

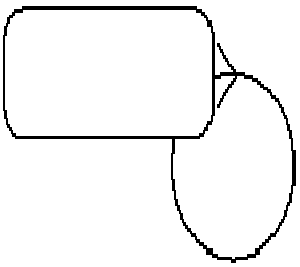


Рис. 3.28. Рекурсивний зв'язок

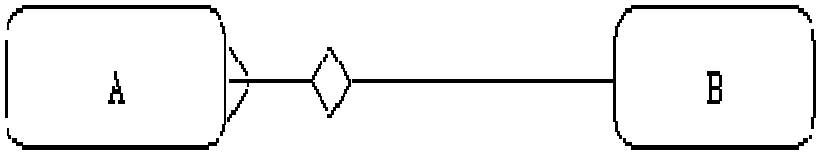


Рис. 3.29. Неперeміщуваний зв'язок

3.4.2. Методологія IDEF1

Метод IDEF1 дозволяє побудувати модель даних, еквівалентну реляційній моделі в третій нормальній формі. В даний час на основі вдосконалення методології IDEF1 створена її нова версія - методологія IDEF1X, яка розроблена з урахуванням таких вимог, як простота вивчення і можливість автоматизації. IDEF1X-діаграми використовуються рядом поширених CASE-засобів (зокрема, ERwin, Design/IDEF).

Сутність в методології IDEF1X є незалежною від ідентифікаторів або просто незалежною, якщо кожен екземпляр сутності може бути однозначно ідентифікований без визначення його відносин з іншою сутністю. Сутність називається залежною від ідентифікаторів або просто залежною, якщо однозначна ідентифікація екземпляра сутності залежить від його відношення до іншої сутності (рис. 3.30).

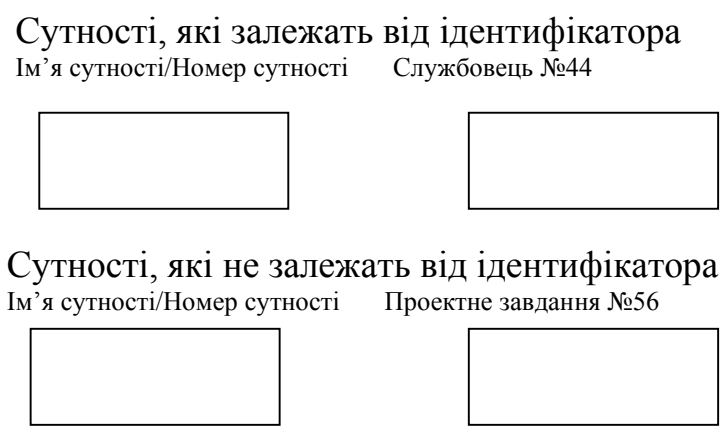


Рис. 3.30. Сутність за методологію IDEF1X

Кожній сутності привласнюється унікальне ім'я і номер, що розділяється косою межею "/" і поміщається над блоком.

Зв'язок може додатково визначатися за допомогою вказівки ступені або потужності (кількості екземплярів сутності-нащадка, яка може існувати для кожного екземпляра сутності-батька). У IDEF1X можуть бути виражені наступні потужності зв'язків:

- кожен екземпляр сутності-батька може мати нуль, один або більше пов'язаних з ним екземплярів сутностей-нащадків;
- кожен екземпляр сутності-батька повинен мати не менше одного пов'язаного з ним екземпляра сутності-нащадка;
- кожен екземпляр сутності-батька повинен мати не більш одного пов'язаного з ним екземпляра сутності-нащадка;
- кожен екземпляр сутності-батька пов'язаний з деяким фіксованим числом екземплярів сутностей-нащадків.

Якщо екземпляр сутності-нащадка однозначно визначається своїм зв'язком з сутністю-батьком, то зв'язок називається ідентифікуючим, інакше – не ідентифікуючим. Зв'язок зображається лінією, що проводиться між сутністю-батьком і сутністю-нащадком з крапкою на кінці лінії у сутності-нащадка. Потужність зв'язку позначається як показано на рис. 3.31 (потужність за умовчанням - N).

Ідентифікуючий зв'язок між сутністю-батьком і сутністю-нащадком зображається суцільною лінією (рис. 3.32).

Сутність-нащадок в ідентифікуючому зв'язку є залежною від ідентифікатора сутністю. Сутність-батько в ідентифікуючому зв'язку може бути як незалежною, так і залежною від ідентифікатора сутністю (це визначається її зв'язками з іншою сутністю).

Пунктирна лінія зображає не ідентифікуючий зв'язок (рис. 3.33). Сутність-нащадок в не ідентифікуючому зв'язку буде незалежною від ідентифікатора, якщо вона не є також сутністю-нащадком в якому-небудь ідентифікуючому зв'язку.

Нуль або один Z	
Нуль або більше P	Нуль, один або більше N

Рис. 3.31. Потужність зв'язку

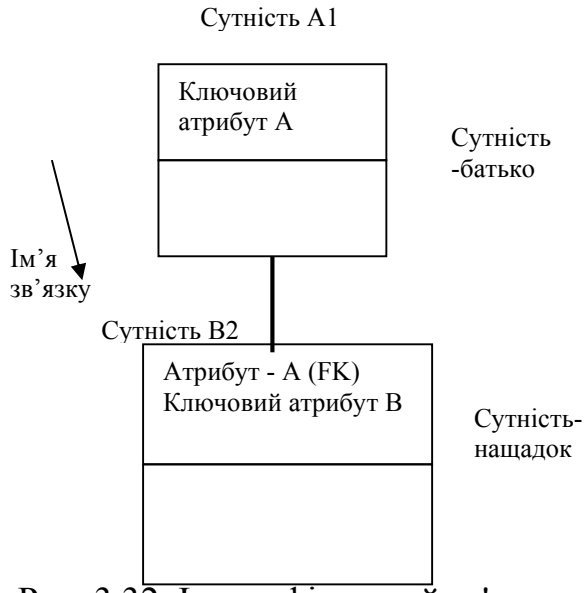


Рис. 3.32. Ідентифікуючий зв'язок

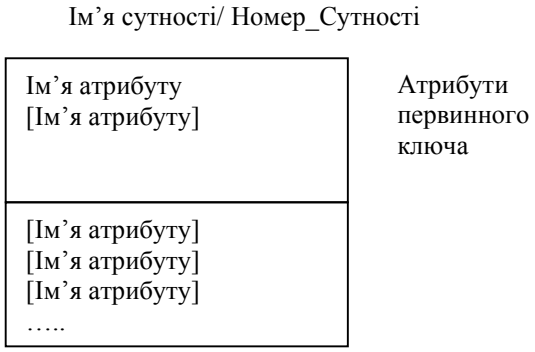


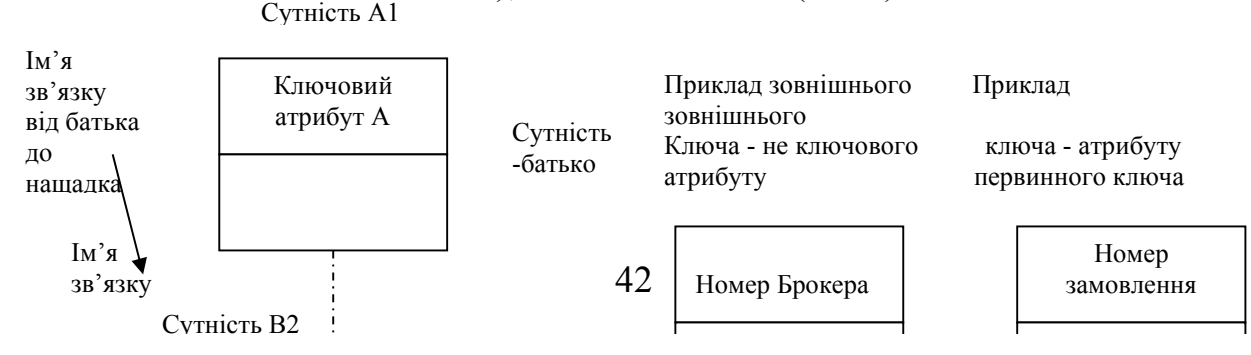
Рис. 3.33. Не ідентифікуючий зв'язок

Атрибути зображаються у вигляді списку імен усередині блоку сутності. Атрибути, що визначають первинний ключ, розміщуються нагорі списку і відділяються від інших атрибутів горизонтальною межею (рис. 3.34).

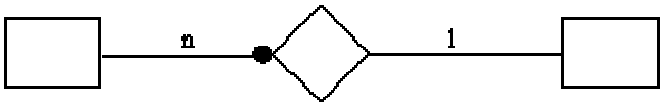
Сутність може мати також зовнішні ключі (Foreign Key), які можуть використовуватися як частина або цілий первинний ключ або не ключовий атрибут. Зовнішній ключ зображається за допомогою приміщення всередину блоку сутності імен атрибутів, після яких слідують букви FK в дужках (рис. 3.35).

3.4.3. Підхід, використовуваний в CASE-засобі Vantage Team Builder

У CASE-засобі Vantage Team Builder на ER-діаграмах сутність позначається прямокутником, що містить ім'я сутності (рис. 3.36), а зв'язок – ромбом, зв'язаним лінією з кожною із взаємодіючих сутностей. Числа над лініями означають ступінь зв'язку. Зв'язки спрямовані до декількох сутностей і можуть мати атрибути (за винятком ключових). Виділяють два види зв'язків: необов'язковий зв'язок (optional); слабкий зв'язок (weak).



У необов'язковому зв'язку (рис. 3.37) можуть брати участь не всі екземпляри сутності.



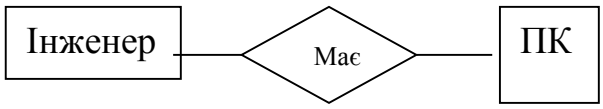


Рис. 3.36. Позначення сутності і зв'язків Рис. 3.37. Необов'язковий зв'язок

На відміну від необов'язкового зв'язку в повному (total) зв'язку беруть участь всі екземпляри хоча б однієї з сутностей. Це означає, що екземпляри такого зв'язку існують тільки за умови існування екземплярів іншої сутності. Повний зв'язок може мати один з 4-х видів: обов'язковий зв'язок, слабкий зв'язок, зв'язок "супертип-підтип" і асоціативний зв'язок.

Обов'язковий (mandatory) зв'язок описує зв'язок між "незалежною" і "залежною" сутністю. Всі екземпляри залежної ("обов'язкової") сутності можуть існувати тільки за наявності екземплярів незалежної ("необов'язкової") сутності, тобто екземпляр "обов'язкової" сутності може існувати тільки за умови існування певного екземпляра "необов'язкової" сутності.

У прикладі (рис. 3.38) мається на увазі, що кожен автомобіль має принаймні одного водія, але не кожен службовець управляє машиною.

У слабкому зв'язку існування однієї з сутностей, яка належить деякій множині ("слабкої") залежить від існування певної сутності, що належить іншій множині ("сильної"), тобто екземпляр "слабкої" сутності може бути ідентифікований тільки за допомогою екземпляра "сильної" сутності. Ключ "сильної" сутності є частиною складового ключа "слабкої" сутності.

Слабкий зв'язок завжди є бінарним і має на увазі обов'язковий зв'язок для "слабкої" сутності. Сутність може бути "слабкою" в одному зв'язку і "сильною" в іншому, але не може бути "слабкою" більш, ніж в одному зв'язку. Слабкий зв'язок може не мати атрибутів.

У прикладі на рис. 3.39 показано, що ключ (номер) рядка в документі може не бути унікальним і повинен бути доповнений ключем документа.

Рис. 3.38. Обов'язковий зв'язок Рис. 3.39. Слабкий зв'язок

Зв'язок "супертип-підтип" зображений на рис. 3.40. Загальні характеристики (атрибути) типу визначаються в сутності-супертипі, сутність-підтип успадковує всі характеристики супертипу. Екземпляр підтипу існує тільки за умови існування певного екземпляра супертипу. Підтип не може мати ключа (він імпортує ключ з супертипу). Сутність, що є супертипом в одному зв'язку, може бути підтипом в іншому зв'язку. Зв'язок супертипу не може мати атрибутів.

У асоціативному зв'язку кожен екземпляр зв'язку (асоціативний об'єкт) може існувати тільки за умови існування певних екземплярів кожної з взаємозв'язаної сутності. Асоціативний об'єкт – об'єкт, що є одночасно сутністю і зв'язком. Асоціативний зв'язок – це зв'язок між декількома "незалежною" сутністю і однією "залежною" сутністю. Зв'язок між незалежною сутністю має атрибути, які визначаються в залежній сутності. Таким чином, залежна сутність визначається в термінах атрибутів зв'язку між рештою сутності.

У прикладі на рис. 3.41 літак виконує посадку на злітну смугу в заданий час при певній швидкості і напрямі вітру. Оскільки ці характеристики застосовні тільки до конкретної посадки, вони є атрибутами посадки, а не літака або злітної смуги. Пілот, що виконує посадку, пов'язаний набагато сильніше з конкретною посадкою, ніж з літаком або злітною смугою.

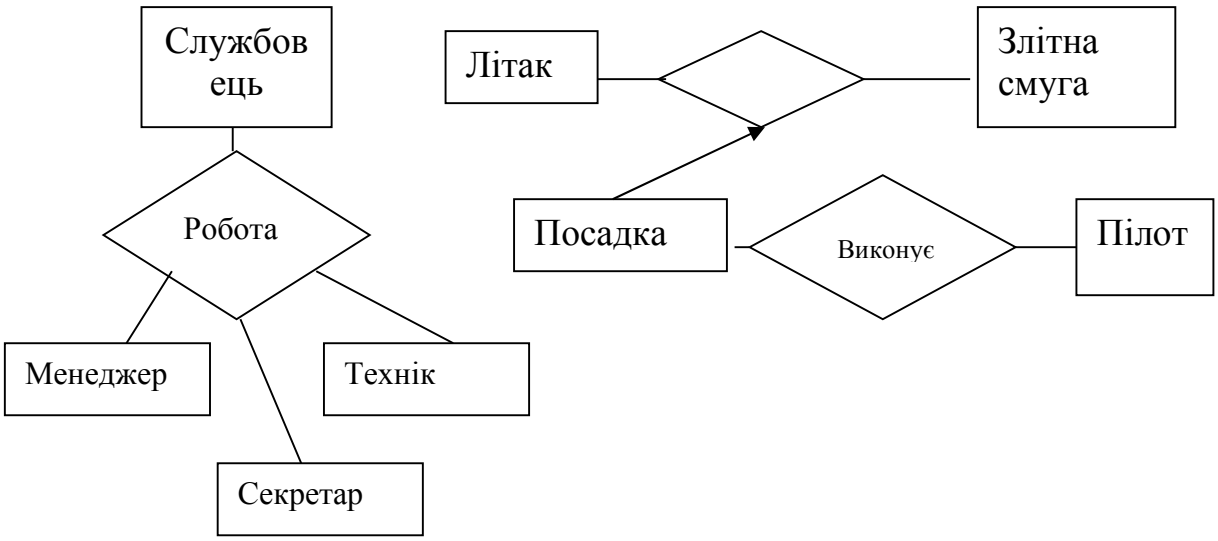


Рис. 3.40. Зв'язок "супертип-підтип" Рис. 3.41. Асоціативний зв'язок

ER-діаграма повинна підкорятися наступним правилам:

- кожна сутність, кожен атрибут і кожен зв'язок повинні мати ім'я (зв'язок супертипу або асоціативний зв'язок може не мати імені);
- ім'я сутності повинне бути унікальне в рамках моделі даних;
- ім'я атрибуту повинне бути унікальне в рамках сутності;

- ім'я зв'язку повинне бути унікальне, якщо для неї генерується таблиця БД;
- кожен атрибут повинен мати визначення типу даних;
- сутність в необов'язковому зв'язку повинна мати ключовий атрибут. Те ж саме відноситься до сильної сутності в слабкому зв'язку, супертипу в зв'язку "супертип-підтип" і необов'язковій сутності в обов'язковому (повному) зв'язку;
- підтип в зв'язку "супертип-підтип" не може мати ключовий атрибут;
- у асоціативному або слабкому зв'язку може бути тільки одна асоціативна (слабка) сутність;
- зв'язок не може бути одночасно обов'язковим, "супертип-підтип" або асоціативним.

3.5. Приклад використання структурного підходу

3.5.1. Опис наочної області

У даному прикладі використовується методологія Yourdon, реалізована в CASE-засобі Vantage Team Builder.

Як наочна область використовується опис роботи відеобібліотеки, яка одержує запити на фільми від клієнтів і стрічки, що повертаються клієнтами. Запити розглядаються адміністрацією відеобібліотеки з використанням інформації про клієнтів, фільми і стрічки. При цьому перевіряється і обновляється список орендованих стрічок, а також перевіряються записи про членство в бібліотеці. Адміністрація контролює також повернення стрічок, використовуючи інформацію про фільми, стрічках і список орендованих стрічок, який обновляється. Обробка запитів на фільми і повернення стрічок включає наступні дії: якщо клієнт не є членом бібліотеки, він не має права на оренду. Якщо необхідний фільм є в наявності, адміністрація інформує клієнта про орендну платню. Проте, якщо клієнт прострочив термін повернення стрічок, що є у нього, йому не дозволяється брати нові фільми. Коли стрічка повертається, адміністрація розраховує орендну платню плюс пені за невчасне повернення. Відеобібліотека одержує нові стрічки від своїх постачальників. Коли нові стрічки поступають в бібліотеку, необхідна інформація про них фіксується. Інформація про членство в бібліотеці міститься окремо від записів про оренду стрічок. Адміністрація бібліотеки регулярно готує звіти за певний період часу про членів бібліотеки, постачальників стрічок, видачу певних стрічок і стрічки, придбані бібліотекою.

3.5.2. Організація проекту

Весь проект розділяється на 4 фази: аналіз, глобальне проектування (проектування архітектури системи), детальне проектування і реалізація (програмування).

На фазі аналізу будується модель середовища (Environmental Model). Побудова моделі середовища включає:

- аналіз поведінки системи (визначення призначення ІС, побудова початкової контекстної діаграми потоків даних (DFD) і формування матриці списку подій (ELM), побудова контекстних діаграм);
- аналіз даних (визначення складу потоків даних і побудова діаграм структур даних (DSD), конструювання глобальної моделі даних у вигляді ER-діаграми).

Призначення ІС визначає угоду між проектувальниками і замовниками щодо призначення майбутньої ІС, загальний опис ІС для самих проектувальників і межі ІС. Призначення фіксується як текстовий коментар в "нульовому" процесі контекстної діаграми.

Наприклад, в даному випадку призначення ІС формулюється таким чином: ведення бази даних про членів бібліотеки, фільми, оренду і постачальників. При цьому керівництво бібліотеки повинне мати нагоду одержувати різні види звітів для виконання своїх задач.

Перед побудовою контекстної DFD необхідно проаналізувати зовнішні події (зовнішні об'єкти), що роблять вплив на функціонування бібліотеки. Ці об'єкти взаємодіють з ІС шляхом інформаційного обміну з нею.

З опису наочної області виходить, що в процесі роботи бібліотеки беруть участь наступні групи людей: клієнти, постачальники і керівництво. Ці групи є зовнішніми об'єктами. Вони не тільки взаємодіють з системою, але також визначають її межі і зображаються на початковій контекстній діаграмі як зовнішня сутність.

Початкова контекстна діаграма зображена на рис. 3.42. На відміну від нотації Gane/Sarson зовнішня сутність позначається звичними прямокутниками, а процеси - колами.

Список подій будується у вигляді матриці (ELM) і описує різні дії зовнішньої сутності і реакцію ІС на них. Ці дії є зовнішніми подіями, що впливають на бібліотеку. Розрізняють наступні типи подій:

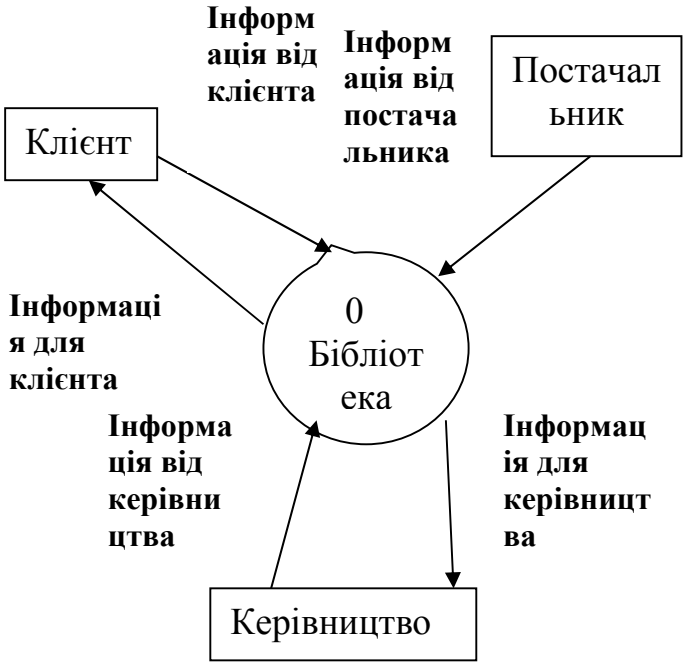


Рис. 3.42. Початкова контекстна діаграма

Абревіатура	Тип
NC	Нормальне управління
ND	Нормальні дані
NCD	Нормальне управління /дані
TC	Тимчасове управління
TD	Тимчасові дані

TCD	Тимчасове управління /дані

Всі дії позначаються як нормальні дані. Ці дані є подіями, які ІС сприймає безпосередньо, наприклад, зміна адреси клієнта повинна бути відразу зареєстрована. Вони з'являються в DFD як вміст потоків даних.

Матриця списку подій має наступний вигляд:

№	Опис	Тип	Реакція
1	Клієнт бажає стати членом бібліотеки	ND	Реєстрація клієнта як члена бібліотеки
2	Клієнт повідомляє про зміну адреси	ND	Реєстрація зміненої адреси клієнта
3	Клієнт запитує оренду фільму	ND	Розгляд запиту
4	Клієнт повертає фільм	ND	Реєстрація повернення
5	Керівництво надає повноваження новому постачальнику	ND	Реєстрація постачальника
6	Постачальник повідомляє про зміну адреси	ND	Реєстрація зміненої адреси постачальника
7	Постачальник направляє фільм в бібліотеку	ND	Отримання нового фільму
8	Керівництво запрошує новий звіт	ND	Формування необхідного звіту для керівництва

Для завершення аналізу функціонального аспекту поведінки системи будується повна контекстна діаграма, що включає діаграму нульового рівня. При цьому провадиться декомпозиція процесу "бібліотека" на 4 процеси, що відображають основні види адміністративної діяльності бібліотеки. Існуючі "абстрактні" потоки даних між термінаторами (зовнішніми подіями) і процесами трансформуються в потоки, що представляють обмін даними на конкретнішому рівні. Список подій показує, які потоки існують на цьому рівні: кожна подія зі списку повинна формувати деякий потік (подія формує вхідний потік, реакція - вихідний потік). Один "абстрактний" потік може бути роздільний на більш ніж один "конкретний" потік.

На фазі аналізу будується глобальна модель даних, що представляється у вигляді діаграми "сутність-зв'язок" (рис. 3.43). Аналіз функціонального аспекту поведінки системи дає уявлення про обмін і перетворення даних в системі. Взаємозв'язок між "абстрактними" потоками даних і "конкретними" потоками даних на діаграмі нульового рівня виражається в діаграмах структур даних (рис. 3.44).

Потоки інформації

Потоки на діаграмі верхнього рівня	Потоки на діаграмі нульового рівня
Інформація від клієнта	Дані про клієнта, Запит про оренду
Інформація для клієнта	Членська картка, Відповідь на запит про оренду
Інформація від керівництва	Запит звіту про нових членів, Новий

	постачальник, Запит звіту про постачальників, Запит звіту про оренду, Запит звіту про фільми
Інформація для керівництва	Звіт про нових членів, Звіт про постачальників, Звіт про оренду, Звіт про фільми
Інформація від постачальника	Дані про постачальника, Нові фільми

Між різними типами діаграм існують наступні взаємозв'язки:

- ELM-DFD: події – вхідні потоки, реакції – вихідні потоки
- DFD-DSD: потоки даних – структури даних верхнього рівня
- DFD-ERD: накопичувачі даних – ER-діаграми
- DSD-ERD: структури даних нижнього рівня - атрибути сутності

На фазі проектування архітектури будується наочна модель. Процес побудови наочної моделі включає:

- детальний опис функціонування системи;
- подальший аналіз використаних даних і побудова логічної моделі даних для подальшого проектування бази даних;
- визначення структури призначеного для користувача інтерфейсу, специфікації форм і порядку їх появи;
- уточнення діаграм потоків даних і списку подій, виділення серед процесів нижнього рівня інтерактивних і не інтерактивних, визначення для них мініспецифікацій.

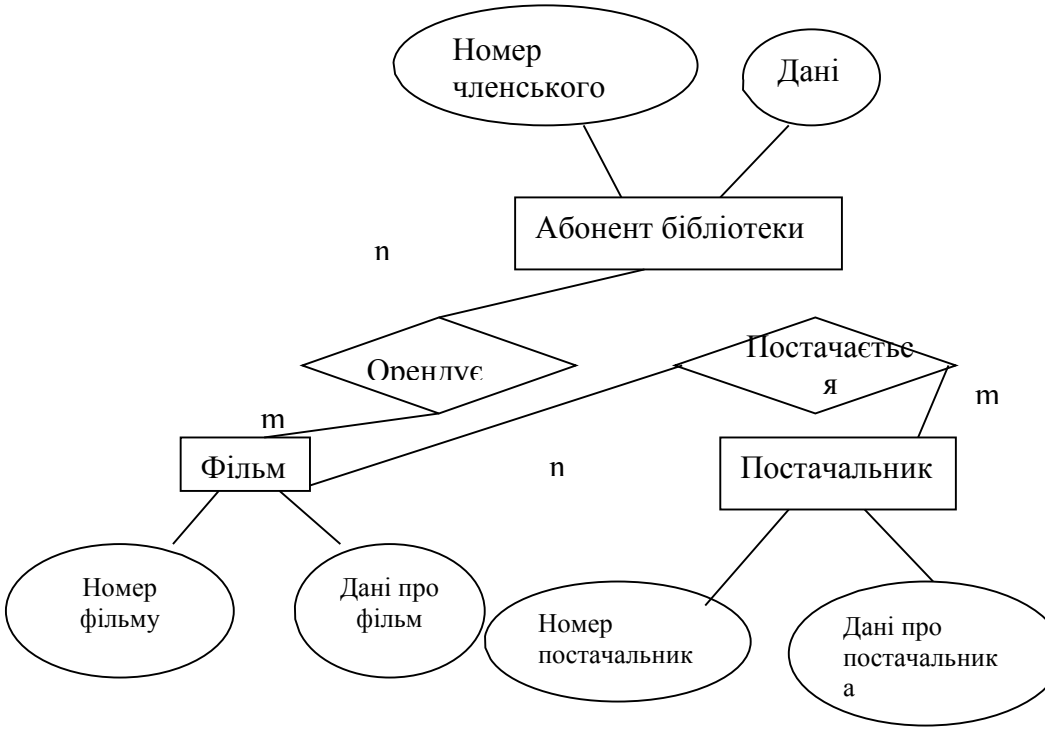


Рис. 3.43. Діаграма "сутність-зв'язок"

Результатами проектування архітектури є:

- модель процесів (діаграми архітектури системи (SAD) і мініспецифікації на структурованій мові);
- модель даних (ERD і підсхеми ERD);
- модель призначеного для користувача інтерфейсу (класифікація процесів на інтерактивні і не інтерактивні функції, діаграма послідовності форм (FSD - Form Sequence Diagram), що показує, які форми з'являються в додатку і в якому порядку. На FSD фіксується набір і структура викликів екранних форм.

Діаграми FSD утворюють ієрархію, на вершині якої знаходиться головна форма додатку, що реалізовує підсистему. На другому рівні знаходяться форми, що реалізовують процеси нижнього рівня функціональної структури, зафіксованої на діаграмах SAD.

На фазі детального проектування будується модульна модель. Під модульною моделлю розуміється реальна модель проектованої прикладної системи. Процес її побудови включає:

- уточнення моделі бази даних для подальшої генерації SQL-пропозицій;
- уточнення структури призначеного для користувача інтерфейсу;
- побудова структурних схем, що відображають логіку роботи призначеного для користувача інтерфейсу і модель бізнес-логіки (Structure Charts Diagram - SCD) і прив'язка їх до форм.

Результатами детального проектування є:

- модель процесів (структурні схеми інтерактивних і не інтерактивних функцій);
- модель даних (визначення в ERD всіх необхідних параметрів для додатків);
- модель інтерфейсу, призначеного для користувача (діаграма послідовності форм (FSD) що показує, які форми з'являються в додатку і в якому порядку, взаємозв'язок між кожною формою і певною структурною схемою, взаємозв'язок між кожною формою і однієї або більш сутністю в ERD).

На фазі реалізації будується реалізаційна модель. Процес її побудови включає:

- генерацію SQL-пропозицій, що визначають структуру цільової БД (таблиці, індекси, обмеження цілісності);
- уточнення структурних схем (SCD) і діаграм послідовності форм (FSD) з подальшою генерацією коду додатків.

На основі аналізу потоків даних і взаємодії процесів зі сховищами даних здійснюється остаточне виділення підсистем (попереднє повинне мало б бути зроблене і зафіксоване на етапі формулювання вимог в технічному завданні). При виділенні підсистем необхідно керуватися принципом функціональної зв'язаності і принципом мінімізації інформаційної залежності. Необхідно враховувати, що на підставі таких елементів підсистеми як процеси і дані на етапі розробки повинен бути створений додаток, здатний функціонувати самостійно. З іншого боку при угрупуванні процесів і даних в підсистеми необхідно враховувати вимоги до конфігурації продукту, якщо вони були сформульовані на етапі аналізу.

На рис. 3.45 представлена контекстна діаграма, побудована на підставі попереднього аналізу. Саме ця діаграма дозволяє почати розробку інформаційної системи у повному обсязі.

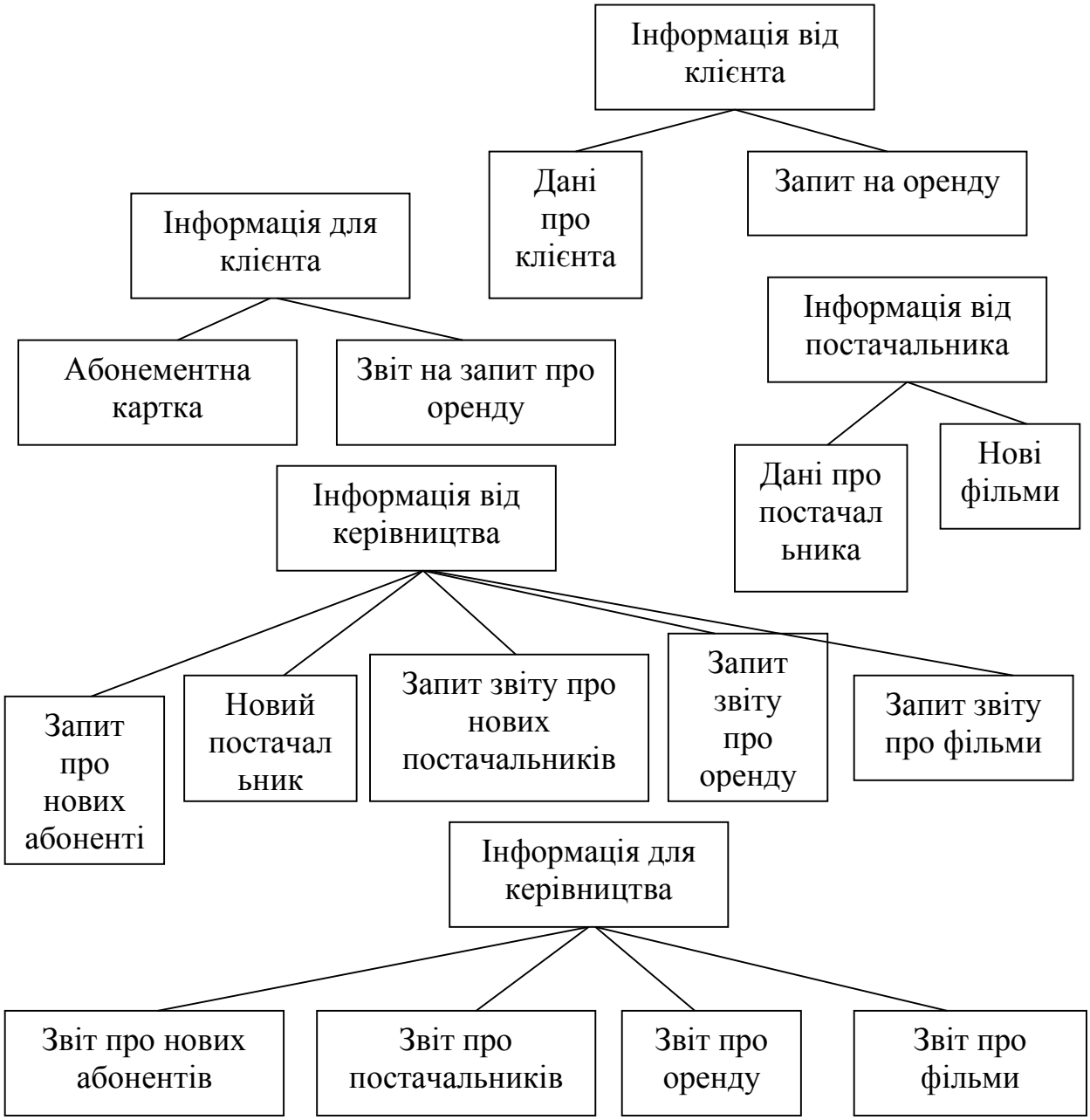


Рис. 3.44. Діаграма структур даних

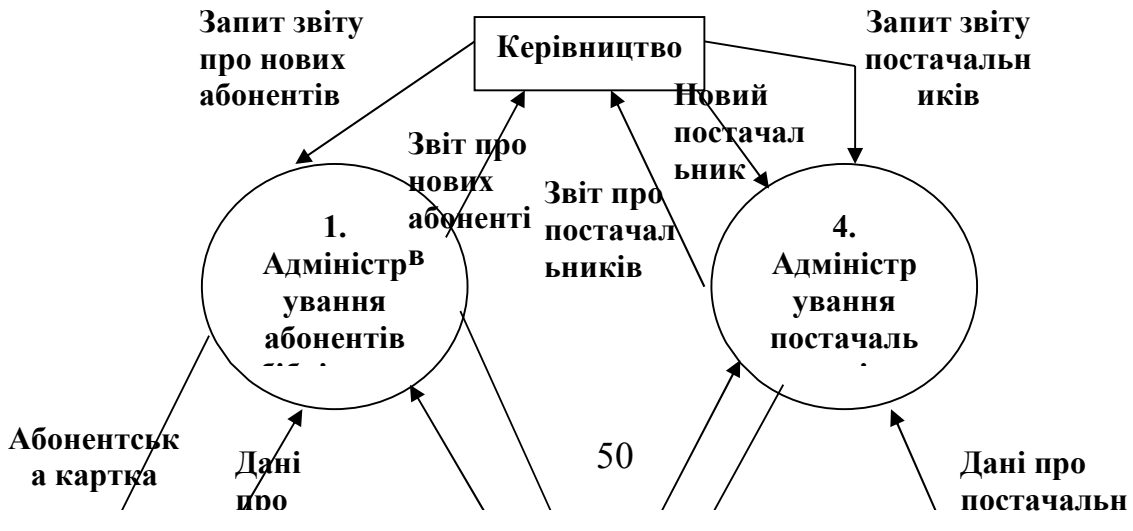


Рис. 3.45. Контекстна діаграма

Контрольні запитання

1. Що таке „ієрархія діаграм”?
2. Як моделювати потоки даних?
3. Чим CASE- метод Баркера відрізняється від методології IDEF1?
4. Що таке „сутність”?
5. Чи контекстна діаграма відрізняється від діаграми структур даних?

В розділі розглянуто три методи графічного представлення потоків інформації, які навчають правилам проектування при CASE-підході. Наведено приклад такого представлення.

4. СТВОРЕННЯ ІС ЗА ПРЕЦЕДЕНТАМИ

У розділі розглядаються принципи побудови прецедентів при проектуванні інформаційної системи.

Прецедент — це набір сценаріїв використання, в якому кожен екземпляр сценарію є послідовністю дій, виконуваних системою для досягнення відчутного результату для конкретного виконавця.

Прецеденти, в основному, це функціональні вимоги, які вказують на те, що повинна робити система. Отже, прецеденти – це вимоги (хоча і не всі вимоги). Деякі вважають вимогами тільки список функцій і властивостей типу "система" прецедентів – це текстові документи, а не діаграми. Моделювання прецедентів – це процес написання тексту, а не малювання.

Прецеденти типу "чорний ящик" (black-box use cases) – це найтиповіший і рекомендований тип прецедентів. Вони не описують внутрішню роботу системи, її компонентів або дизайн. Навпаки, системі ставляться деякі обов'язки (responsibilities). Тобто, програмні елементи мають обов'язки і взаємодіють з іншими елементами зі своїми обов'язками. Визначаючи обов'язки системи через прецеденти типу "чорний ящик", можна вказати, що повинна робити система (функціональні вимоги), не розписуючи, як це робити (не виконуючи проектування).

Прецеденти описуються в різних форматах, залежно від потреб. Крім типів "чорного ящика" і "білого ящика", виділяють декілька ступенів формалізації опису прецедентів.

- **Стислий** – анотація у вигляді одного абзацу. Звичайно вона описує тільки головний успішний сценарій. Приклад такого опису приведений нижче для прецеденту «Оформлення продажу» (Process Sale).
- **Вільний** – неформальний стиль опису. Опис прецеденту займає декілька абзаців і охоплює різні сценарії. Прикладом такого опису є розглянутий нижче прецедент «Повернення товару».
- **Розгорнутий** – найдокладніший стиль опису. При такому підході детально описуються всі кроки і варіанти розвитку сценарію, а також передумови і результати.

4.1. Прецеденти і задачі

Виконавці мають свої задачі (або потреби), для вирішення яких вони використовують систему. Тому прецеденти рівня EBP ще називають прецедентами рівня задач користувача (user goal). Це робиться для того, щоб звернути увагу на реалізацію потреб користувачів системи або основного виконавця,

Звідси слідує алгоритм виділення прецедентів.

1. Виділити задачі (цілі) користувачів.

2. Визначити для кожної з них окремий прецедент.

Як виділити прецедент? Часто визначити правильний (а точніше, корисний) прецедент дуже складно. Кожну задачу можна розглядати на різних рівнях деталізації, починаючи від конкретних простих дій і закінчуючи діяльністю на рівні підприємства.

Часто при реалізації проекту важливо не що зробити, а як. Перелік використовуваних технологій теж приводиться в описі прецеденту. Типовим прикладом такої ситуації є технічні обмеження, що висуваються зацікавленими особами для технологій введення і висновку. Наприклад, замовник може зажадати, щоб POS-система підтримувала введення даних кредитної картки з клавіатури і за допомогою спеціального пристрою.

Приклад списку технологій і типів даних для задачі „Продаж”

3а. Ідентифікатор товару прочитується зі штрих-коду (за наявності останнього) лазерним сканером або вводиться з клавіатури.

3б. Ідентифікатор товару може визначатися за схемами кодування UPC, EAN, JAN або SKU.

7а. Інформація про відкритий кредит вводиться за допомогою спеціального пристрою або з клавіатури.

7б. Підпис при оплаті чеком ставиться на паперовому документі. Проте очікується, що протягом двох років більшість покупців вимагатиме цифрові пристрої прочитання підпису.

Приклад основного сценарію для задачі „Продаж”

Дія виконавця	Відгук системи
1. Покупець підходить до касового апарату POS-системи з вибраними товарами	
2. Касир відкриває новий продаж	
3. Касир вводить ідентифікатор товару	4. Система записує найменування товару і видає його опис, ціну і загальну вартість. Ціна обчислюється на основі набору правил.
Касир повторює дії, описані в пп. 3-4 для кожного найменування товару.	5. Система обчислює загальну вартість покупки з податком
6. Касир повідомляє покупцю загальну вартість і пропонує сплатити покупку.	
7. Покупець оплачує покупку.	8. Система обробляє платіж.
	9. Система реєструє продаж і відправляє інформацію про неї до зовнішньої

Дія виконавця	Відгук системи
	бухгалтерської системи (для оновлення бухгалтерських документів і нарахування комісійних) і системі складського обліку (для оновлення даних). Система видає товарний чек.

4.2. Допоміжні задачі і прецеденти

Не дивлячись на те, що задача "представитися" системі і виконати аутентифікацію (або задачу реєстрації) не була віднесена до рівня задач користувача, вона все ж таки є метою, але на нижчому рівні. Такі задачі називають **допоміжними** (subfunctional goal), оскільки вони покликані забезпечувати виконання задач користувача. Для допоміжних задач окремі прецеденти створюються дуже рідко, хоча фахівці з написання прецедентів часто рекомендують покращувати (зазвичай спрощувати) набір прецедентів.

Для допоміжних задач писати прецеденти не забороняється, проте це не завжди потрібно, оскільки при цьому ускладнюється модель прецедентів. Кількість допоміжних задач для системи може обчислюватися сотнями, але не варто створювати багато прецедентів.

Важливо розуміти, що зі збільшенням числа прецедентів зростає складність задачі формулювання і управління вимогами, а значить, збільшується також час рішення цієї задачі,

Основним мотивом написання прецеденту для допоміжної задачі повинна служити повторюваність цієї задачі або її важливе значення як передумови для множини інших прецедентів. Цьому принципу задовольняє задача авторизації, яка забезпечує передумову для більшості, якщо не всієї решти прецедентів рівня задач користувача.

Задачі можуть належати до різних рівнів складності і бути складовими, починаючи від рівня підприємства ("бути прибутковим"), до задач середнього рівня ("реєстрація торгових операцій") і допоміжних задач в рамках додатку ("перевірка правильності введення").

Аналогічно, і прецеденти можуть відноситися до різних рівнів складності і складатися з прецедентів нижчого рівня.

Наявність декількох рівнів складності задач і прецедентів може ввести в оману при визначенні відповідного рівня для основних прецедентів. Для відсівання дуже детальної інформації слід використовувати принцип знаходження елементарних бізнес-процесів.

4.3. Визначення основних виконавців, задач і прецедентів

Прецеденти призначені для задоволення потреб основних виконавців. Тому для виділення прецедентів використовується наступна процедура.

1. Визначте рамки системи: чи є вона програмним додатком, апаратно-програмним комплексом, чи включає своїх користувачів або всю організацію?

2. Ідентифікуйте основних виконавців, потреби (цілі) яких задовольняються за допомогою системи.

3. Для кожного виконавця визначите його задачі. Складіть ієрархію відповідно до рекомендацій по виділенню EBP.

4. Визначте прецеденти, що задовольняють потреби кожного виконавця, і привласніть їм імена відповідно до задач. Звичайно основні прецеденти відповідають задачам користувачів, за одним виключенням, про яке мова піде нижче.

Крок 1. Визначення рамок системи

Для визначення рамок системи слід, в першу чергу, вказати, що до неї не відноситься, тобто визначити зовнішніх основних і допоміжних виконавців. Після ідентифікації зовнішніх виконавців рамки системи обкреслюються чіткіше. Наприклад, чи покладається на систему повна відповідальність за авторизацію платежів? Ні, цю задачу виконує зовнішній виконавець – служба авторизації платежів.

Кроки 2 і 3. Визначення основних виконавців і задач

Не можна однозначно вказати послідовність визначення виконавців і задач. Звичайно на семінарі за визначенням вимог методом мозкового штурму ідентифікуються і ті й інші артефакти. Іноді виконавці визначаються після формулювання задач, а іноді навпаки.

В процесі мозкового штурму основну варто приділити визначенню основних виконавців, оскільки це розширить можливості для подальшого дослідження.

При визначенні основних виконавців і задач користувачів слід відповісти на наступні питання, щоб не упустити з вигляду деякі неочевидні моменти.

- Хто запускає і вимикає систему?
- Хто є системним адміністратором?
- Хто здійснює управління користувачами і безпекою?
- Чи відноситься час до числа виконавців, іншими словами, чи повинна система виконувати які-небудь дії у відповідь на події часу?
- Чи існує процес моніторингу, завдяки якому система перезапускається у разі збою?
- Хто контролює діяльність і продуктивність системи?
- Як виконується оновлення програмного забезпечення?
- Хто аналізує журнали реєстрації? Чи можна забезпечити видалений

доступ до них?

4.4. Основні та допоміжні виконавці

Нагадаємо, що основні виконавці – це ті, чиї потреби задовольняються за допомогою системи. Для вирішення своїх задач вони використовують систему. На відміну від них, допоміжні виконавці (supporting actor) займаються обслуговуванням системи. Поки зосередимося на ідентифікації основних виконавців.

Нагадаємо також, що основними виконавцями, серед іншого, можуть бути інші комп'ютерні системи.

Складіть список основних виконавців і їх задач В термінах артефактів уніфікованого процесу цей список повинен бути розділом артефакту «Бачення». Розглянемо наступну таблицю.

Виконавець	**Задачі**	**Виконавець**	**Задачі**
Касир	Оформляє продажі Оформляє кредити Виконує повернення товару Реєструє виручку	Системний адміністратор	Додає користувачів. Змінює параметри користувачів. Видаляє користувачів. Управляє безпекою. Управляє системними таблицями.
Менеджер	Включає систему Вимикає систему	Система аналізу торгової діяльності	Аналізує інформацію про продажі і оцінює продуктивність

Чому основним виконавцем для прецеденту **Оформлення продажу** є касир, а не покупець? Чому покупець не включений в список виконавців?

Відповідь визначається рамками системи. Якщо підприємство або торгову організацію розглядати як **агрегатну систему**, то для неї основним виконавцем повинен бути покупець, задача якого – придбання товарів або послуг. Проте з погляду самої POS-системи (яка визначає рамки системи для даного прецеденту), основним виконавцем є касир, задача якого — обслуговування продажів.

Для визначення виконавців, їх задач і прецедентів можна також використовувати зовнішні події. Що це означає? Часто до одного і того ж рівня ЕВР або прецеденту, наприклад, відноситься ціла група подій.

Зовнішня подія	Ініціатор	Задача
Введення інформації про найменування товару	Касир	Оформити продаж
Введення інформації про платіж	Касир або покупець	Оформити продаж

Крок 4. Визначення прецедентів

Як правило, кожній задачі користувача відповідає один прецедент рівня ЕВР. Його ім'я повинне відповідати назві задачі, наприклад, задачі оформлення продажу повинен відповідати прецедент **Оформлення продажу**.

Як правило, ім'я прецеденту починається з іменника, що описує дію.

Типовим виключенням з правила відповідності задач і прецедентів є прецедент, що вирішає чотири задачі – створення, відновлення, оновлення і видалення. Звичайно такий прецедент називається «Управління чим-небудь». Наприклад, задачі "Зміна інформації про користувачів", "Видалення користувачів" і т.д. розв'язуються в рамках прецеденту «Управління користувачами».

Визначення прецедентів виконується у декілька етапів, одні з яких займають декілька хвилин (наприклад, привласнення імен прецедентам), а інші – по декілька днів або тижнів (розгорнений опис). У подальших розділах цього розділу, присвячених уніфікованому процесу, ці етапи будуть розглянуті в контексті ітеративної розробки.

Для опису прецедентів для декількох ітерацій потрібно скликати семінари для визначення вимог.

Специфікація вимог створює лише ілюзію коректності. Можна з упевненістю стверджувати, що прецеденти і інші вимоги в специфікації вимог не коректні. Може не вистачати важливої інформації і містяться невірні твердження. Для вирішення цієї проблеми використовується ітеративність процесу розробки, але на додаток до нього іноді потрібне постійне особисте спілкування – щоденне обговорення між всіма розробниками за участю фахівця з наочної області, який уповноважений ухвалювати рішення про вимоги до системи. Потрібен хтось, до кого програмісти можуть підійти і за декілька секунд зняти виниклі у них питання. Наприклад, в рамках підходу XP існує відмінний принцип: користувач повинен постійно знаходитися серед учасників проекту, в тому ж приміщенні.

4.5. Опис прецедентів, що відносяться до інтерфейсу користувача, у вільному стилі

Дослідження цілей, а не обов'язків і процедур дозволяє зосередити увагу на основних вимогах. Наприклад, на одному з семінарів касир може сказати, що однією з його цілей є реєстрація в системі. При цьому він може мати на увазі елементи інтерфейсу користувача, відповідні діалогові вікна, введення

ідентифікатора і пароля. Проте все це – механізми досягнення мети, а не сама мета. Вивчаючи ієрархію цілей (відповідаючи на питання "яка мета цієї задачі?"), системний аналітик приходить до формулювання, незалежного від механізму реалізації опису алгоритму "ідентифікувати себе і виконати аутентифікацію", або на вищому рівні – "запобігти просочуванню інформації".

Такий процес дослідження дозволяє одержати нові і ефективніші рішення. Наприклад, у наш час достатньо поширені і недорого коштують клавіатура і миша з пристроями зчитування біометричної інформації, зокрема відбитків пальців. Якщо метою є ідентифікація й аутентифікація, то чом би для її досягнення не використовувати ефективний і швидкий засіб зчитування біометричних даних з клавіатури? Проте відповідаючи на це питання, слід брати до уваги зручність користування. У даному випадку доведеться встановити профілі типових користувачів. А якщо їх пальці чимось забруднені? А якщо вони травмовані?

Базовий стиль опису припускає виклад на рівні намірів користувача і обов'язків системи, а не на рівні їх конкретних дій. При такому стилі опису не потрібно заглиблюватися в деталі технології і механізму реалізації, особливо при розгляді питань, пов'язаних з інтерфейсом користувача.

Помітимо, що поняття мета і намір є синонімами. Цим підтверджується взаємозв'язок між базовим стилем опису і орієнтацією на меті (задачі). Дійсно, багато намірів виконавців можна трактувати як допоміжні цілі.

4.6. Виконавці

Виконавець (actor) – це сутність, що має поведінку. До числа виконавців може відноситися і сама комп’ютерна система, якщо вона викликає служби інших систем. У прецеденті можуть брати участь основні і допоміжні (другорядні) виконавці. Виконавцями є не тільки люди, але і організації, машини і програми. Існує три типи зовнішніх по відношенню до системи, що розробляється, виконавців.

- Основний виконавець (primary actor) – його задачі виконуються з використанням системи. Прикладом основного виконавця є касир.

Навіщо його ідентифікувати? Щоб визначити цілі користувача, на основі яких формулюються прецеденти.

- Допоміжний виконавець (supporting actor) – обслуговує систему (наприклад, надає інформацію). Прикладом допоміжного виконавця є служба авторизації платежів.

Навіщо його ідентифікувати? Щоб визначити зовнішні інтерфейси і протоколи.

- Закулісний виконавець (offstage actor) – зацікавлений в реалізації прецеденту, але не є основним або допоміжним виконавцем. Прикладом закулісного виконавця є податкова служба.

Навіщо його ідентифікувати? Щоб упевнитися, що всі інтереси визначені і задоволені. Інтереси закулісних виконавців звичайно не очевидні і їх легко упустити з вигляду, якщо не ідентифікувати

4.7. Вимоги і списки низькорівневих властивостей

Основним мотивом опису прецедентів є визначення вимог в контексті задач (цілей) і сценаріїв використовування системи. Це дуже добре, оскільки дозволяє поглибити розуміння системи. Проте прецеденти – це не єдині артефакти формулювання вимог. Деякі не функціональні вимоги, правила і інші елементи краще описати в додатковій специфікації, розгляду якої буде присвячений наступний розділ.

Основна ідея – замінити детальні списки низькорівневих властивостей (які звичайно складалися при використовуванні традиційного методу формулювання вимог) прецедентами (за деякими виключеннями). Ці списки були згруповані по функціональному призначенню і виглядали приблизно таким чином.

Ідентифікатор	**Властивість**
Властивість 1.9	Система дозволятиме вводити нові ідентифікатори
Властивість 2.4	Система реєструватиме платежі по кредитній картці в бухгалтерській системі

Такі списки теж можуть виявитися корисними, проте один подібний список займає не пів сторінки, а десятки або сотні сторінок, Це викликає деякі проблеми, вирішити які дозволяють прецеденти. До числа проблем відносяться наступні,

- Довгі, докладні списки функцій не відображають вимог в загальному контексті. Різні функції і властивості організовані в звичний список розрізнених елементів. А при описі прецедентів вимоги розглядаються в контексті задач і цілей системи.
- Якщо використовуються і описи прецедентів і списки функцій, то відбувається дублювання інформації. На це витрачається більше праці, потрібно більше зусиль для їх осмислення і узгодження.

Прагніть замінити докладні списки властивостей низького рівня описом прецедентів.

Списки властивостей високого рівня системи цілком допустимі

Достатньо часто функціональні властивості системи описуються в коротких списках властивостей високого рівня в рамках документа "Бачення". Це цілком виправдано. На відміну від списку низькорівневих властивостей, перерахованих на 100 сторінках, список основних властивостей системи повинен включати не більш декілька десятків елементів. У ньому перераховані лише функціональні властивості системи, не згадані при описі прецедентів.

Список властивостей системи

- Реєстрація продажів.
- Авторизація платежів (кредитних, дебетних, чекових).
- Системне адміністрування для управління користувачами, безпекою, таблицями кодів і констант і т.п.
- Автоматична обробка інформації про продажі у фоновому режимі у

разі відмови зовнішніх компонентів.

- Взаємодія у реальному часі із зовнішніми системами (на основі промислових стандартів), включаючи систему складського обліку, обчислення податкових відрахувань, бухгалтерську систему, систему управління людськими ресурсами і служби авторизації платежів.
- Визначення і виконання бізнес-правил, що настроюються автоматично, у фіксованих точках сценарію.

4.8. Прецеденти в рамках уніфікованого процесу

Прецеденти виконують життєво важливу роль при реалізації уніфікованого процесу, оскільки вся розробка в рамках цього підходу здійснюється під управлінням прецедентів (use-case driven development). Це означає наступне.

- Вимоги в основному формулюються при описі прецедентів (у моделі прецедентів). Решта вимог (якщо такі існують) є або технічними (наприклад, список функцій), або другорядними.
- Прецеденти – важливий етап ітеративного планування. На кожній ітерації реалізуються деякі сценарії або цілі прецеденти. Тому описи прецедентів вносять істотний внесок в оцінювання результату.
- Розробка додатку полягає в реалізації прецедентів. Тобто група розробників продумує способи взаємодії об'єктів або архітектуру підсистем для реалізації прецедентів.

В рамках UP виділяють два види прецедентів: системні і бізнес-прецеденти. **Системні прецеденти** (system use-case) – це такі, що розглядалися в цьому розділі, наприклад Оформлення продажу. Вони створюються в рамках дисципліни "Вимоги" і є частиною моделі прецедентів.

Бізнес-прецеденти (business use-case) використовуються набагато рідше. При необхідності вони створюються в рамках дисципліни "Бізнес-моделювання" як частина великомасштабного бізнес-процесу або для полегшення розуміння контексту нової системи. Вони описують послідовність дій в цілому, виконуваних **бізнес-виконавцем** (business actor) (виконавцем в бізнес-середовищі, наприклад, споживачем). Зокрема, для ресторану можна виділити бізнес-прецедент **Приготування блюда**.

Приклад: прецеденти початкової фази проекту „Торгівля”

На початковій стадії не всі прецеденти описуються детально. Модель прецедентів на цьому етапі може бути розроблена на наступному рівні деталізації.

У розгорненому форматі	У вільному форматі	Короткий опис
Оформлення продажу	Обчислення орендної платні	Реєстрація виручки Управління користувачами

Повернення товару	Аналіз торгової діяльності	Запуск системи Завершення роботи Управління системними таблицями

Приклад розгорненого опису прецеденту при продажі товарів

Для розгорненого опису прецедентів існують різні шаблони форматування. Проте найчастіше використовується шаблон, приведений на Web-вузлі www.usecases.org. Цей стиль проілюстрований в наступному прикладі.

1. Покупець підходить до касового апарату PQS-системи з вибраними товарами.
2. Касир відкриває новий продаж.
3. Касир вводить ідентифікатор товару,
4. Система записує найменування товару і видає його опис, ціну і загальну вартість. Ціна обчислюється на основі набору правил. Касир повторює дії, описані в пп. 3-4, для кожного найменування товару.
5. Система обчислює загальну вартість покупки з податком.
6. Касир повідомляє покупцю загальну вартість і пропонує сплатити покупку.
7. Покупець оплачує покупку, система обробляє платіж,
8. Система реєструє продаж і відправляє інформацію про неї зовнішній бухгалтерській системі (для оновлення бухгалтерських документів і нарахування комісійних) і системі складського обліку (для оновлення даних).
9. Система видає товарний чек.
10. Покупець покидає магазин з чеком і товарами (якщо він щось купив).

Контрольні запитання

1. Що таке прецедент?
2. Для чого потрібна така система проектування інформаційних систем?
3. Як відрізнити основні і допоміжні прецеденти?
4. Що таке „виконавці” в системі опису прецедентів?
5. Як побудувати вимоги властивостей різного рівня?

Розглянуто систему створення описів проекту інформаційної системи за допомогою принципу „прецедентів”, в яких виділяються основні та допоміжні задачі, виконавці і користувачі.

5. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

1. Індивідуальна робота виконується на комп`ютері в лабораторії або по місцю роботи студента, або в іншому місці, де є доступ до комп'ютерів з операційною системою LINUX.
2. Індивідуальна робота виконується із застосуванням пакету Open Office.
3. Індивідуальна робота складається з друкованого тексту з титульним листом (див. зразок), та дискети, яка містить цю роботу. З іншої сторони титульного листа наклеюється паперовий клапан, в який і вкладається дискета.
4. Електронна частина звіту по лабораторній роботі, яка містить технічне завдання на проектування інформаційної системи згідно з варіантом студента, записується у форматі текстового редактора Writer. Шрифт – Times New Roman, кегль 14, 1,5 інтервали. Береги листа – всі по 2 см. Нумерація сторінок – всередині низу листа.
5. Структура роботи:
    - Завдання на розробку проекту;
    - Анотація
    - Титульний лист;
    - Зміст, де вказані розділи та номера сторінок;
    - Тексту:
    - Списку використаних джерел, включаючи адреси сайтів, де була знайдена потрібна інформація.

1. Завдання на розробку проекту

Завдання на містить *вказівку* наочної області, функціональні і кількісні вимоги до системи, а також, інші вимоги, що визначають індивідуальність системи.

До функціональних вимог відноситься основні функції, виконувані системою, конфігурація і розподілені властивості системи, наявність локальних або видалених робочих місць, забезпечення безпеки і цілісності інформації, використовуваної в системі, наявність інтерфейсів зв'язку з іншими інформаційними системами.

До кількісних параметрів системи можуть бути віднесені кількість призначених для користувача робочих місць, об'єм даних (кількість записів в найкритичніших таблицях), що зберігаються, вимоги до продуктивності (час виконання запитів, час реакції на команди користувача і ін., швидкість обробки потоків даних), додаткові технічні вимоги.

2. Анотація

У анотації гранично стисло висловлюється зміст виконаного проекту:

- відомості про об'єм проекту, кількість сторінок і ілюстрацій до кожного документа;
- мета проекту;
- основні результати, одержані в проекті;
- суть виконаного проекту, проведені дослідження і розрахунки;
- основні характеристики розробленого об'єкту;
- оригінальні авторські рішення, використані в проекті. Об'єм анотації не повинен перевищувати одного листу.

3. Записка пояснення

У записці пояснення студент в короткій і чіткій формі повинен висловити творчий задум проекту, застосовані методи і технічні рішення, результати розрахунків і досліджень.

Всі ухвалені розробником технічні рішення мають бути обґрунтовані. Як обґрунтування можуть використовуватися порівняння запропонованих варіантів, рекомендації літературних джерел (зі вказівкою конкретних умов їх застосовності), результати розрахунків.

Текстовий і ілюстративний матеріал, узятий з літератури і ін. джерел, допускається приводити лише у виняткових випадках, коли без цього неможливо виконати розрахунок, зробити висновки і т.д., тобто коли просто посилання на відповідне джерело недостатнє.

Зміст повинен бути конкретним і відноситися тільки безпосередньо до теми розробки. Не допускається переписування з літератури в проектне завдання визначень, виведень співвідношень, загальних положень і ін. Достатньо послатися на кінцевий результат.

Рекомендований зміст може виглядати таким чином:

Зміст

Вступ

1. Розробка і аналіз технічного завдання
   - 1.1. Опис наочної області
   - 1.2. Розробка технічного завдання
   - 1.3. Аналіз технічного завдання
   - 1.4. Вибір способів і засобів рішення виконання технічного завдання
2. Розробка моделі процесів об'єкту професійної діяльності
3. Розробка моделі даних об'єкту професійної діяльності
4. Розрахунки і оцінки
   - 4.1. Розрахунок необхідних ресурсів обчислювальних засобів
   - 4.2. Оцінка завантаження обчислювальних засобів, оцінка продуктивності
5. Висновок
6. Бібліографічний список
7. Додатки

Проект починається з титульного листу. Далі слідує зміст, перший лист якого оформляється як заголовний лист за формою 5 ГОСТ 2.104-68.

У ***вступі*** стисло розглядається сучасний стан інженерної або наукової задачі, рішенню якої сприяє виконання проекту. Указується місце і значення (в даний час або в перспективі) проектованого об'єкту в загальній системі, конструкції або виробництві. Наголошується доцільність розробки з погляду потреб виробництва або учбового процесу. Указується ступінь новизни (нова розробка або модернізація існуючої).

У розділі ***Розробка і аналіз технічного завдання*** описується наочна область, об'єкту професійної діяльності, що розробляється, і чітко формулюються задачі, які розв'язуються в проекті. Цей розділ може складатися з наступних частин.

1. Дослідження (опис) наочної області, в якій формулюються основні вимоги і особливості наочної області, що впливають на розробку проекту.

2. Формулювання технічного завдання. У технічному завданні приводиться конкретне формулювання вимог, яким повинна задовольняти проектована система. Серед них:

- вимоги до конфігурації системи (розподілені властивості системи, наявність локальних або видалених робочих місць);
- функціональні вимоги до системи (основні функції, виконувані системою);
- кількісні вимоги до системи (кількість робочих місць, кількість і об'єм записів, об'єм файлів даних, необхідний час реакції системи, час виконання запитів, швидкість обробки потоків даних і др.);
- вимоги по безпеці і цілісності інформації (категорії доступу користувачів до тієї або іншої інформації) ;
- вимоги по сумісності (наявність інтерфейсів зв'язку з іншими інформаційними системами, сумісність з колишніми форматами даних.

3. Аналіз завдання. У цьому пункті приводиться змістовна постановка задачі, в якій аналізуються можливі способи реалізації функціональних, кількісних і інших вимог до новостворюваної системи або особливості роботи вже функціонуючої системи.

4. Вибір способів і засобів, в якій здійснюється вибір методології (структурна, об'єктно-орієнтована, інша) і інструментальних засобів (CASE-засоби, сервер Бази Даних, мова програмування, методи захисту інформації і т.д.) рішення поставлених задач. попереднє техніко-економічне обґрунтування вибраних рішень.

У розділі ***Розробка моделі процесів об'єкту професійної діяльності*** виконується проектування архітектури системи і побудова моделей процесів.

Розробка програмного забезпечення може виконуватися або з використанням структурного підходу (у даному курсі), або об'єктно-орієнтованого.

При використанні структурного підходу розробляються моделі процесів (або функціональні моделі), в яких описується функціонування обробки потоків даних в методології IDEF0, DFD або IDEF3.

Один з основних процесів в системі описується у формі моделі прецедентів. Далі по моделі прецедентів будується модель процесів.

Описується загальна архітектура системи (файл-сервер, клієнт-сервер, "товстий" або "тонкий" клієнт, розподіл робочих місць, розподілене або локальне зберігання даних і ін.). Описується розподіл функцій системи по підсистемах, указуються протоколи обміну даними.

Кількість і склад моделей і діаграм визначається особливостями обробки потоків даних і повинна забезпечувати чітке і несуперечливе розуміння його механізмів функціонування. При цьому, якщо обробка потоку даних має ряд типових компонентів, то достатньо опис функціонування одного з них.

У розділі ***Розробка моделі даних об'єкту професійної діяльності*** виконується проектування моделей даних. Модель розробляється в середовищі ERWin в стандарті IDEF1X. Дається словесне обґрунтування моделі, вибору суті і типів зв'язків. Приводяться фізична і логічна моделі у вигляді схем. Указується, як модель даних пов'язана з моделлю процесів.

У розділі ***Розрахунки і оцінки*** проводяться наступні розрахунки, що відображають специфіку розробляється обробка потоку даних.

– Розрахунок (оцінка) необхідних ресурсів обчислювальних засобів. Служить для кількісного обґрунтування вибору технічних засобів;

– Оцінка завантаження обчислювальних засобів, оцінка продуктивності. Служить для обґрунтування виконання вимог технічного завдання за часом реакції системи на запити і інші дії користувачів, оцінки частки часу або частки використовування ресурсів технічних засобів;

У підсумках приводяться висновки про ступінь відповідності виконаного проекту технічному завданню і оцінка одержаних техніко-економічних показників.

Після висновку дається список літератури, на яку робляться посилання в проекті.

У додатках поміщаються структурні схеми, схеми даних і діаграми, що не увійшли до основного тексту курсового проекту. Нумерація сторінок в додатках окрема - в кожному з них з 1-й сторінки.

5. Дискета має бути підписана так:

Індивідуальна робота з ПІС
ст.гр.ОА-96-1
Косач Лариси Петрівни
10.10.2008 р.

Дата вказує на момент передачі курсової роботи на перевірку.

6. Всі документи повинні бути виконані українською мовою.

7. Студентам заочного відділення здавати контрольні роботи треба в аудиторію 3/32 секретарю кафедри т.373-07-84. Студенти денного відділення здають свої роботи викладачам, які є керівниками індивідуальної роботи.

8. Кожен студент виконує роботу на тему, яка визначається наступною таблицею, згідно номера (m) за списком групи та загальної кількості студентів в групі (n) за формулою.

$$N\text{вар} = Z^*k - (n+m),$$

де k – кількість тем у таблиці з переліком тем, Z= 1, 2, 3, ….. – ціле число натурального ряду, яке вибирається так: щоб $Zk > (n+m)$.

Розробити технічне завдання на проектування інформаційної системи:

1.	Облік кадрів середнього підприємства
2.	Ведення бібліотеки стандартних вузлів для САПР
3.	Інформаційна підтримка управління якістю підприємства
4.	Облік талонів на обслуговування поліклініки
5.	Облік простою вагонів на залізниці
6.	Облік і розміщення номенклатури на складі
7.	Автоматична заміна товарів, яких немає в наявності на складі, на інші альтернативні товари
8.	Ідентифікація історії походження номенклатури: серійний номер і номер партії
9.	Ведення юридичної інформації про договори з клієнтами і постачальниками, умовах оплати, контактах і відповідальних
10.	Прив'язка накладних і оплат до конкретного договору (реєстрація договору в рядках журналів обліку, замовленнях, закупівлях, накладних і оплатах з подальшим перенесенням в проводку по клієнту/постачальнику)
11.	Автоматичне/періодичне зіставлення проводок по контрагентах і договорах
12.	Перевірка наявності у клієнта ліцензії на замовлені медикаменти..
13.	Реєстр незадоволеного попиту
14.	Відстеження номенклатур по серійному номеру і номеру партії.
15.	Управління карантином. Переглядання номенклатури на карантинному складі на будь-якому етапі контролю якості.
16.	Розрахунок потреби в матеріалах і потужностях.
17.	Прогнози закупівель і продажів. Можливість огляду довгострокових потреб по закупівлі, виробництву і ресурсам.
18.	Відстеження пропозиції товарів на продаж.
19.	Обробка не поставлених товарів.
20.	Ідентифікація фізичного розміщення товару: склад, осередок і палети.

Кількість робочих місць користувачів і адміністраторів знаходиться за формулами:

$$\text{Кількість робочих місць} = [n/2 + m/10]$$

$$\text{Кількість адміністраторів} = [n/15 + m/10],$$

Квадратні дужки означають округлення до найближчого більшого цілого , але результат не може бути не менше одиниці.

Кількість файлів у базі даних = [k*(n/20 + m/10)/10],
але не менше 3.

Приклад визначення теми: нехай студент має номер за списком групи 21, а всього у групі 29 студентів.

Визначимо число (n+m) = 21+29 = 50.

Тоді основна його тема буде 3*20 – 50 = 10. Тут k = 20, z = 3 «Прив'язка накладних і оплат до конкретного договору (вказівка договору в рядках журналів ГК, замовленнях, закупівлях, накладних і оплатах з подальшим перенесенням в проводку по клієнту/постачальнику».

Кількість робочих місць = [29/2 + 21/10] = 16.

Кількість адміністраторів = [29/15 + 21/10] = 4.

Кількість файлів у базі даних = [20*(29/20 + 21/10)/10] = 7.

ЗРАЗОК ОФОРМЛЕННЯ ТИТУЛЬНОГО ЛИСТА

Міністерство освіти і науки України
Національний гірничий університет
Кафедра економічної кібернетики та інформаційних технологій

НАВЧАЛЬНИЙ ПРОЕКТ
з дисципліни **«ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»**
Осінній (чи весняний) семестр 2008/2009 року
НА ТЕМУ " Прив'язка накладних і оплат до конкретного договору (вказівка договору в рядках журналів ГК, замовленнях, закупівлях, накладних і оплатах з подальшим перенесенням в проводку по клієнту/постачальнику"

Розробила: ст.гр.ЕК-06-1
Косач Лариса Петрівна
Варіант m=26, n=21
Прийняв: проф. І.М.Пістунов
Каф.ЕКІТ

Дніпропетровськ
2008

ПІДСУМКИ

Сучасна технологія проектування інформаційних систем неможлива без засобів автоматичного проектування. Ці засоби (програми) характеризуються вимогами формалізації постановки задачі у вигляді спеціальних графіків.

Графічне представлення процедури створення проекту інформаційної системи вимагає ієрархічного підходу до розгляду об’єкта автоматизації. Кожен графік вищого рівня може бути підданий декомпозиції і представлений більш складним графіком нижчого рівня. В термінологію введено поняття сутностей, як окремих елементів інформаційної системи.

Для зручнішого засвоєння матеріалу викладення в посібнику було розбито на чотири розділи.

В першому було подано основні принципи створення проекту інформаційних систем, їх класифікація. Розглянуто поняття економічної інформаційної системи та її структуру. Визначено вимоги до методології проектування, стадії і етапи проектування, а також технологічні операції проектування.

Другий розділ присвячено основам методології проектування, таким як життєвий цикл та його моделі життєвого циклу програмного забезпечення, загальні вимоги до методології і технології, зокрема, розглянуто методологію RAD.

У третьому розділі викладається структурний підхід до проектування ІС. У тому числі методологія функціонального моделювання SADT, ієрархія діаграм, типи зв'язків між функціями, Case-метод Баркера, методологія IDEF1, підхід, використовуваний в CASE-засобі Vantage Team Builder. Подано приклад використання структурного підходу.

Четвертий розділ розглядає принципи створення ІС за прецедентами. Визначення основних виконавців, задач і прецедентів, опис прецедентів, що відносяться до інтерфейсу користувача, у вільному стилі, Вимоги і списки низькорівневих властивостей та прецеденти в рамках уніфікованого процесу.

У п’ятий розділ включено індивідуальні завдання на проектування простих інформаційних систем за допомогою CASE- методів, що дозволить поглибити знання у цих областях знань.

Безумовно, застосування таких прийомів та програм дозволить прискорити процес проектування інформаційних систем при загальному зменшенні числа помилок в ній.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. ANSI/IEEE 1008 – 1986. Тестирование программных модулей и компонент ПС.
2. ANSI/IEEE 1012-986. Планирование проверки (оценки) (verification) и подтверждения достоверности (validation) программных средств.
3. ANSI/IEEE 829 – 1983. Документация при тестировании программ.
4. ISO 09000-3:1991. Общее руководство качеством и стандарты по обеспечению качества.
5. ISO 09126:1991. ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.
6. ISO 09646 – 1-6: 1991. ИТ. ВОС. Методология и основы аттестационного тестирования ВОС.
7. ISO 12207:1995. Процессы жизненного цикла программных средств.
8. Бойко В.В. Савиников В.М. Проэктирование баз данных информационных систем. – М.: Финансы и статистика, 1989.
9. Вендров А.М. CASE-технологии. современные методы и средства проектирования информационных систем. – 2004. – http://www.ariadna.net.ru/#1.html
10. Герасименко В.А. Защита информации в автоматизированных системах обработки данных. Книги 1 и 2 – М.: Энергоатомиздат, 1994.
11. Горчинська О.Ю. Designer/2000 - нове покоління CASE-продуктів фірми ORACLE. /"СУБД", 1995, №3.
12. Горін С.В., Тандоєв А.Ю. Застосування CASE-засобу Erwin 2.0 для інформаційного моделювання в системах обробки даних./ "СУБД", 1995, №3.
13. Горін С.В., Тандоєв А.Ю. CASE-засіб S-Designor 4.2 для розробки структури бази даних. /"СУБД", 1996, №1.
14. DATARUN Concepts. Computer Systems Advisers Research Ltd., 1994.
15. SE Companion Installation and Administration Manual. SECA Inc., 1995.
16. Новоженов ю.В. Объектно-ориентированные технологии разработки сложных программных систем. – М.: Энергия, 1996.
17. Панащук С.А. Розробка інформаційних систем з використанням CASE-системи Silverrun. /"СУБД", 1995, №3.
18. Шлєєр З., Меллор С. Об'ектно-орієтованний аналіз: моделювання миру в станах. – Київ: "Діалектика", 1993.
19. Зиндер е.З. Бизнес-реынжиниринг и технологии системного проектирования. Учебной пособие. – М., Центр Информационных Технологий, 1996.
20. Калянов г.Н. CASE. Структурный системный анализ (автоматизация и приложение). – М., "Лори", 1996.

ПРЕДМЕТНИЙ ПОКАЖЧИК

ДОДАТОК. Список програм, які реалізують CASE-метод

Silverrun 5.1.2. JAM.
Vantage Team Builder (Westmount I-CASE) + Uniface.
Designer/2000 + Developer/2000.
Локальні засоби – ERwin, BPwin, S-Designor, CASE.Аналитик.
Об'єктно-орієнтовані CASE-засоби – Rational Rose.

Комплекс технологій і інструментальних засобів створення ІС, заснований на методології і технології DATARUN. До складу комплексу входять наступні інструментальні засоби:

CASE-засіб Silverrun;
засіб розробки додатків JAM;
міст Silverrun-RDM <-> JAM;
комплекс засобів тестування QA;
менеджер транзакцій Tuxedo;
комплекс засобів планування і управління проектом SE Companion;
комплекс засобів конфігураційного управління PVCS;
об'єктно-орієнтований CASE-засіб Rational Rose;
засіб документування SoDA.

Прикладами інших подібних комплексів є:

Vantage Team Builder for Uniface + Uniface (фірми "DataX/Florin" і "ЩОК");

комплекс засобів, що поставляються і використовуваних фірмою "ФОРС":

CASE-засоби Designer/2000 (основне), ERwin, Bpwin і Oowin (альтернатів-ниє);

засоби розробки додатків Developer/2000, ORACLE Power Objects (основниє) і Usoft Developer (альтернативне);

засіб настройки і оптимізації ExplainSQL (Platinum);

засоби адміністрування і супроводу SQLWatch, DBVision, SQL Spy, TSReorg і ін. (Platinum);

засіб документування ORACLE Book.

комплекс засобів на основі продуктів фірми CENTURA:

CASE-засоби ERwin, Bpwin і Oowin (об'єктно-орієнтований аналіз);

засоби розробки додатків SQLWindows і TeamWindows;

засіб тестування і оптимізації додатків "клієнт-сервер" SQLBench (ARC);

засоби експлуатації і супроводу Quest і Crystal Reports.

Навчальне видання

Пістунов Ігор Миколайович

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Навчальний посібник

У редакції автора
Комп'ютерна верстка І.М. Пістунова

Підписано до друку 11.03.2008. Формат 30 х 42/4.
Папір офсетний. Ризографія. Умовн. друк. арк. 3,72.
Обліково-видавн. арк. 3,73. Тираж 150 прим. Зам. №

Підготовлено до друку та надруковано
в Національному гірничому університеті.
Свідоцтво про внесення до державного реєстру ДК №1842.
49005, м. Дніпропетровськ, просп. К. Маркса, 19.